Formal alignment of high-level architecture models with requirements models

***Racem Bougacha*[1]**, *Régine Laleau[2], Simon Collart-Dutilleul[1]*

[1] COSYS-ESTAS, IFSTTAR, Université Gustave Eiffel, campus de Lille, France
[2]LACL, Université Paris-Est Créteil, Créteil, France

Journée GT IE & INFORSID
Mars 2024

(presentation from ICECCS 2023)

Outline of the presentation

- Context of the work
- HLA modelling and verification approach
- SysML/KAOS approach
- Graphical alignment links
- Formalization of graphical alignment links
- Conclusion

Context

- **Autonomous freight train project**: a project of the French IRT Railenium (Technological Research Institute), involving industrial and academic partners for developing Rail Research and Innovation.

- In the project, complex systems are seen as an interplay of heterogeneous sub-systems, generally critical, in particular their development process is most often challenging since it could be difficult to verify that stakeholders needs are satisfied.

- High-level architecture (HLA) of these systems are represented as an interconnected hierarchy of their sub-systems

- Requirements traceability is a crucial element of any especially for the design of critical complex systems

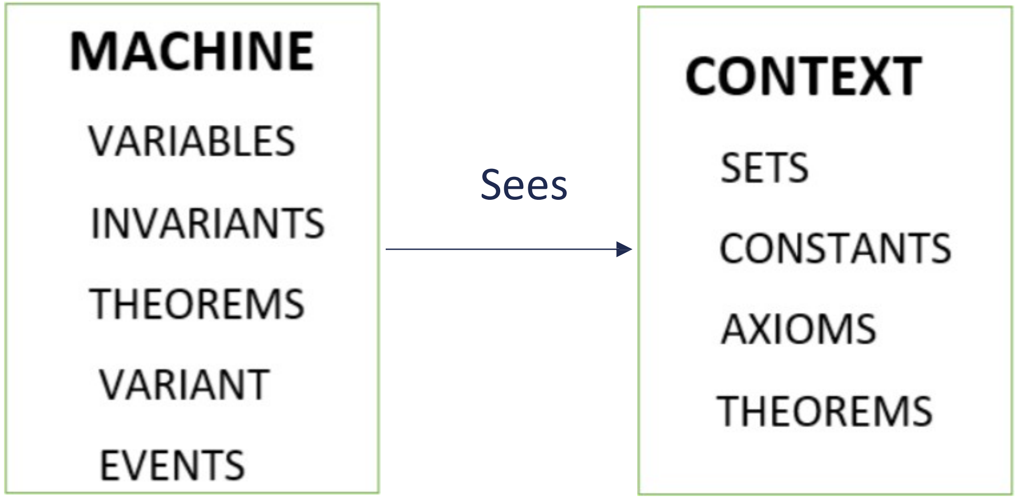
Problem statement and motivations

- High-level architecture (HLA) models must be aligned with requirements models.
  - ➢ The need of a graphical alignment links between HLA models and requirements models for critical systems.
- Critical complex systems require formal and rigorous reasoning.
  - ➢ The need of a formalization of these alignment links.

Our objective: Defining graphical and formal alignments to be validated by experts of various domains (railways systems)

Event-B

- A formal method based on set theory and first order logic
- An Event-B model is composed of a set of contexts and machines

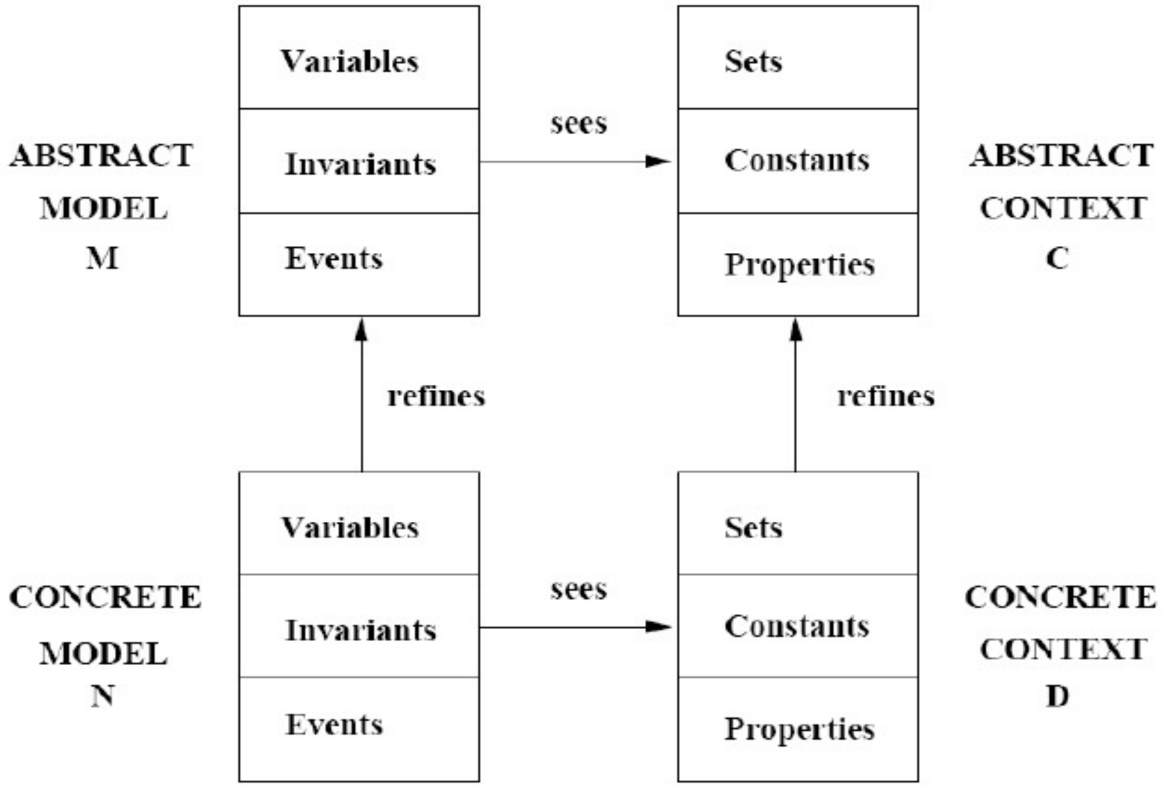


Event E = SELECT G(v) THEN S(v) END

with

- ***G(v)*** **: guard**
- ***S(v)*** **: substitution**
- ***v*** **: state variables and local variables**

Event-B refinement process



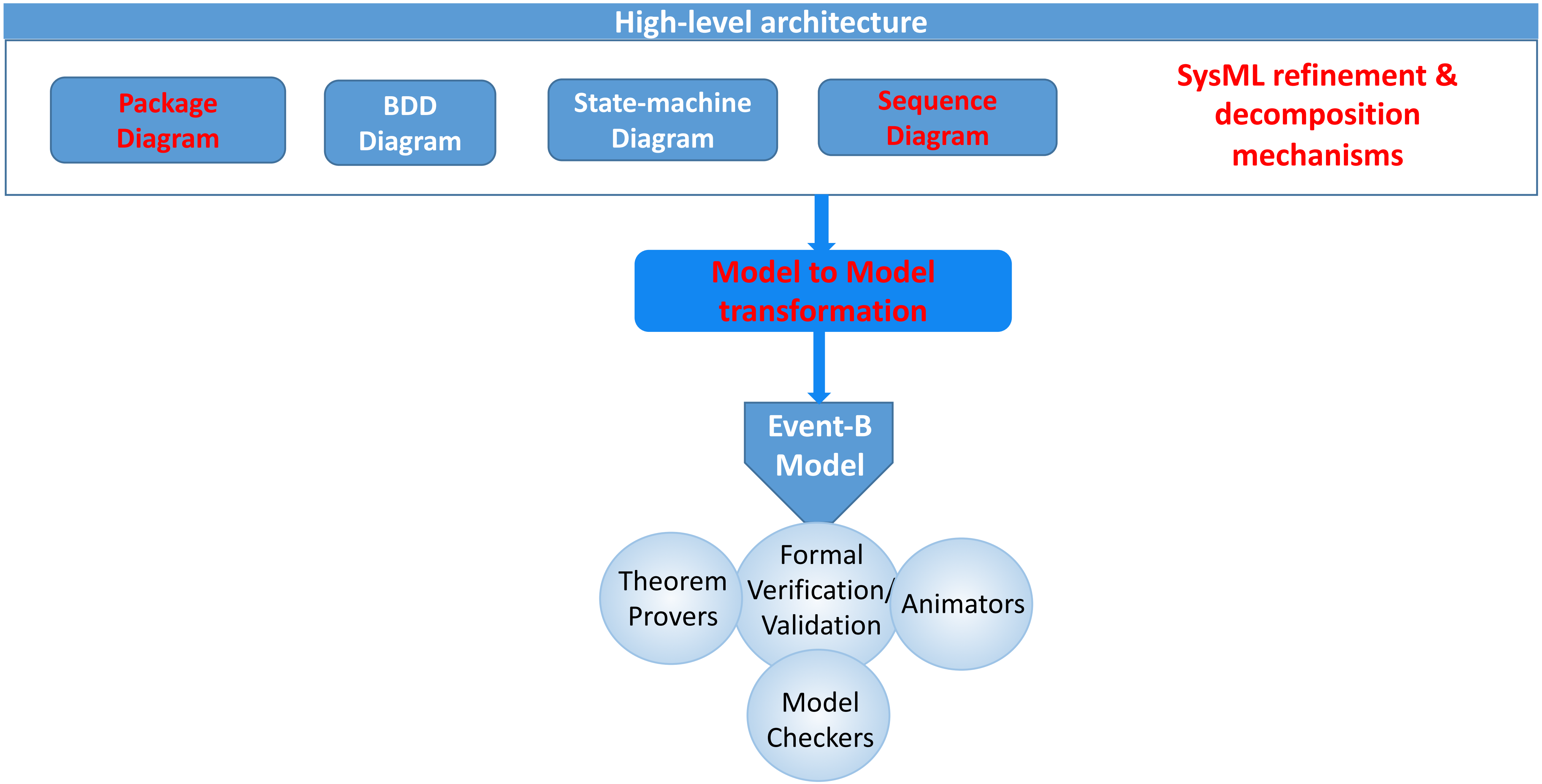
- **Semantics of models and refinements given by proof obligations**
- **Supported by industrial tools (AtelierB, ProB, Rodin platform …)**

HLA modelling

- Providing an automatic translation from SysML diagrams to Event-B specifications

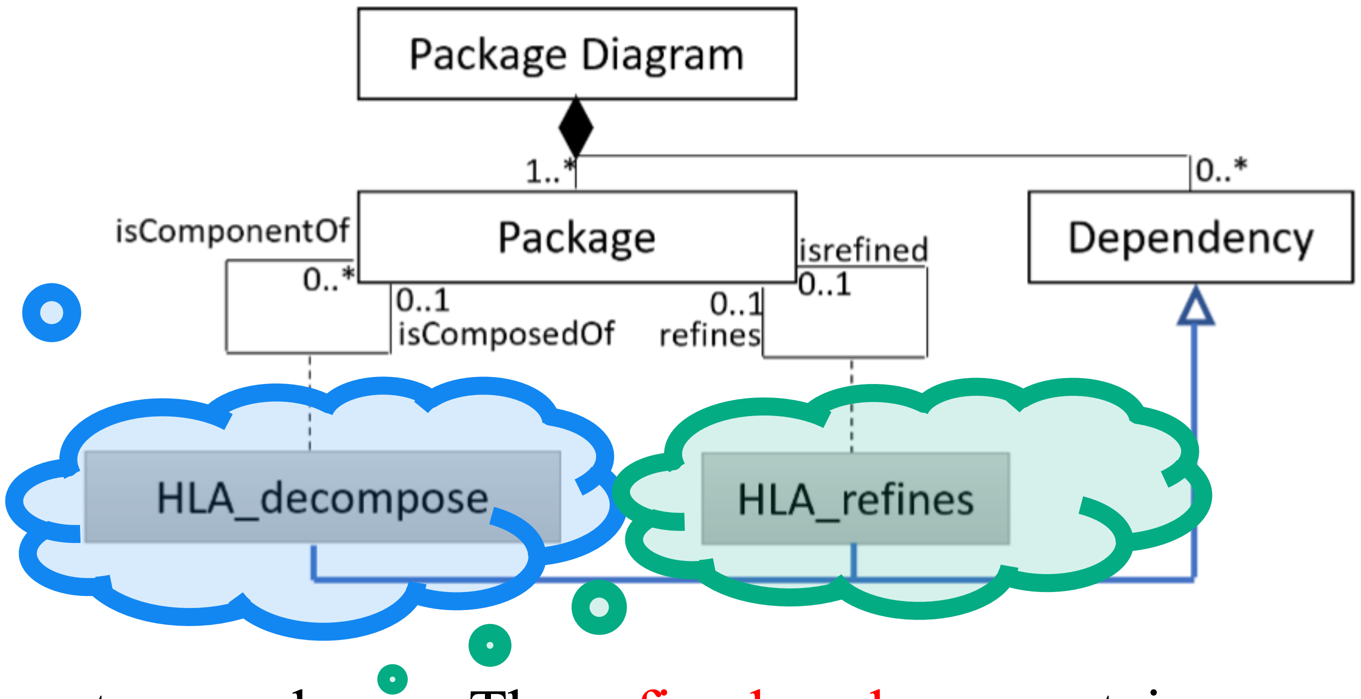
- Extending SysML with the refinement and decomposition mechanisms of Event-B to facilitate a step-by-step design for mastering complexity

The methodology for HLA modelling



SysML package diagram extensions with refinement and decomposition mechanisms

HLA_decompose: allows to decomposed a package modeling a system into a set of packages modeling each of its sub-systems



HLA_refines: defined between two packages. The refined package contains the modeling elements of a system. The refining package contains the modeling elements of its sub-systems to detail the behavior of the parent package

SysML sequence diagram extension with refinement

SysML sequence diagram extension:

A message of a refining package is a refinement of a message of the refined package.

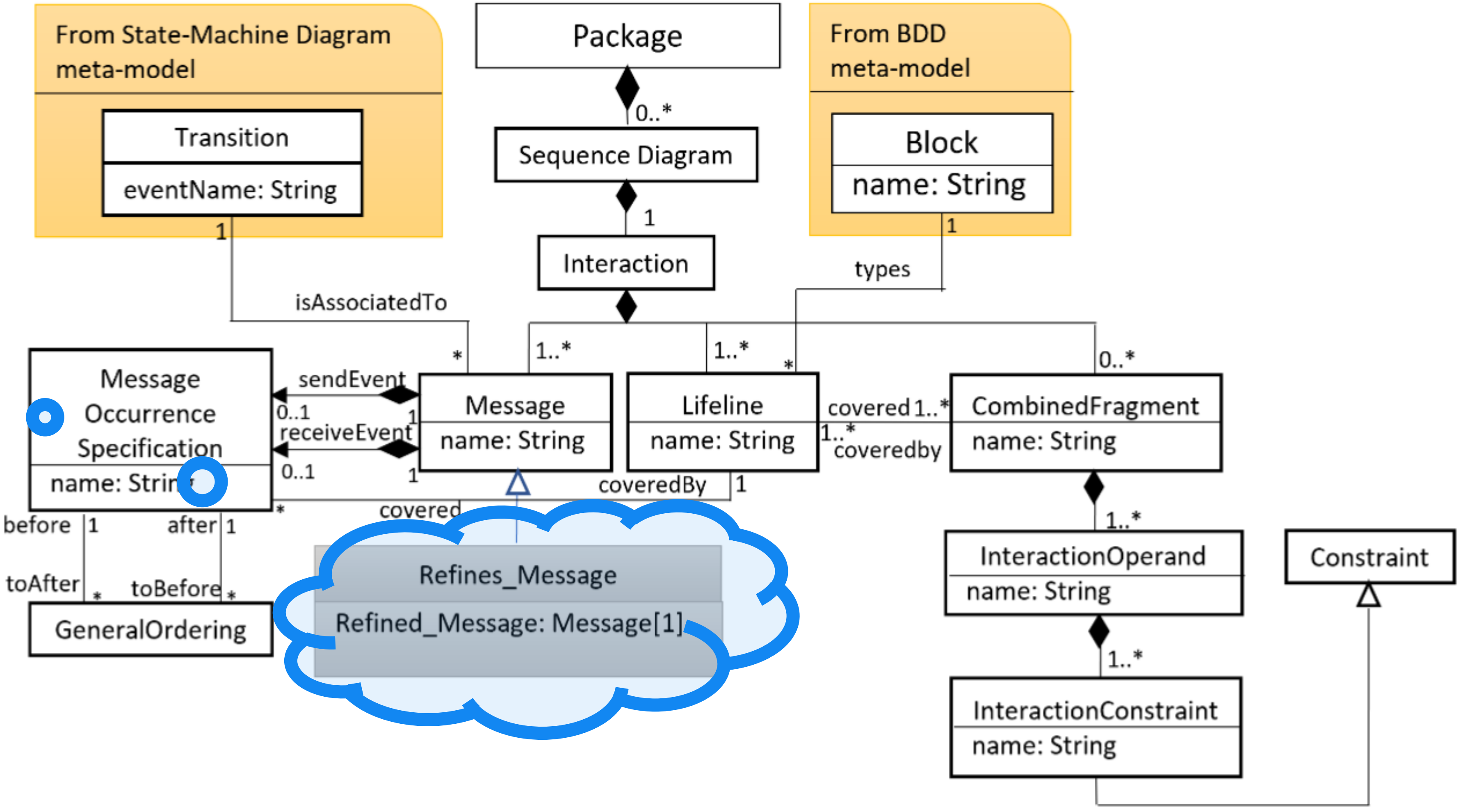
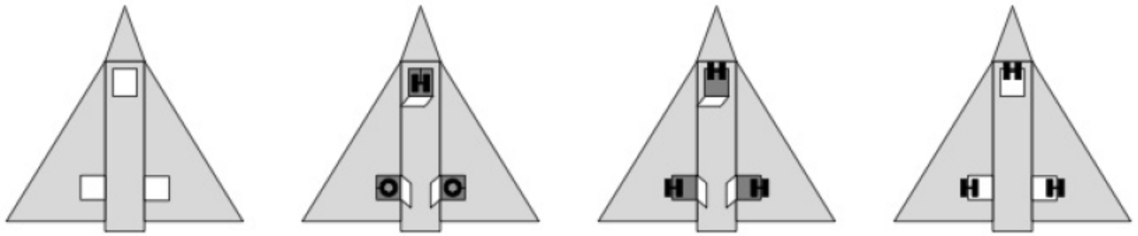


Illustration of the SysML extensions

Landing Gear System case study



a) System in initial state

b) Open doors

c) Extend the "Landing Gear"

d) Close the doors

Abstract Model

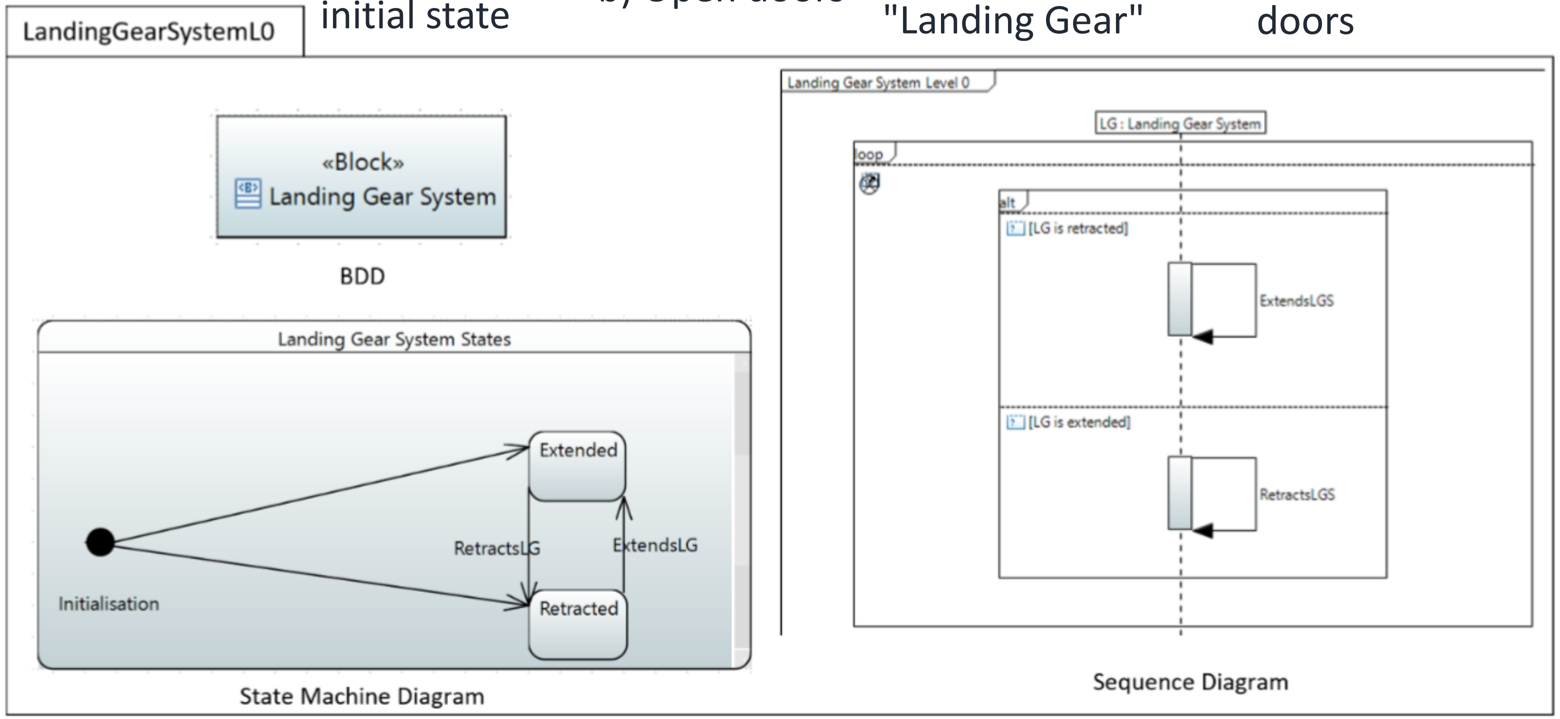


Illustration of the SysML extensions

First Refinement

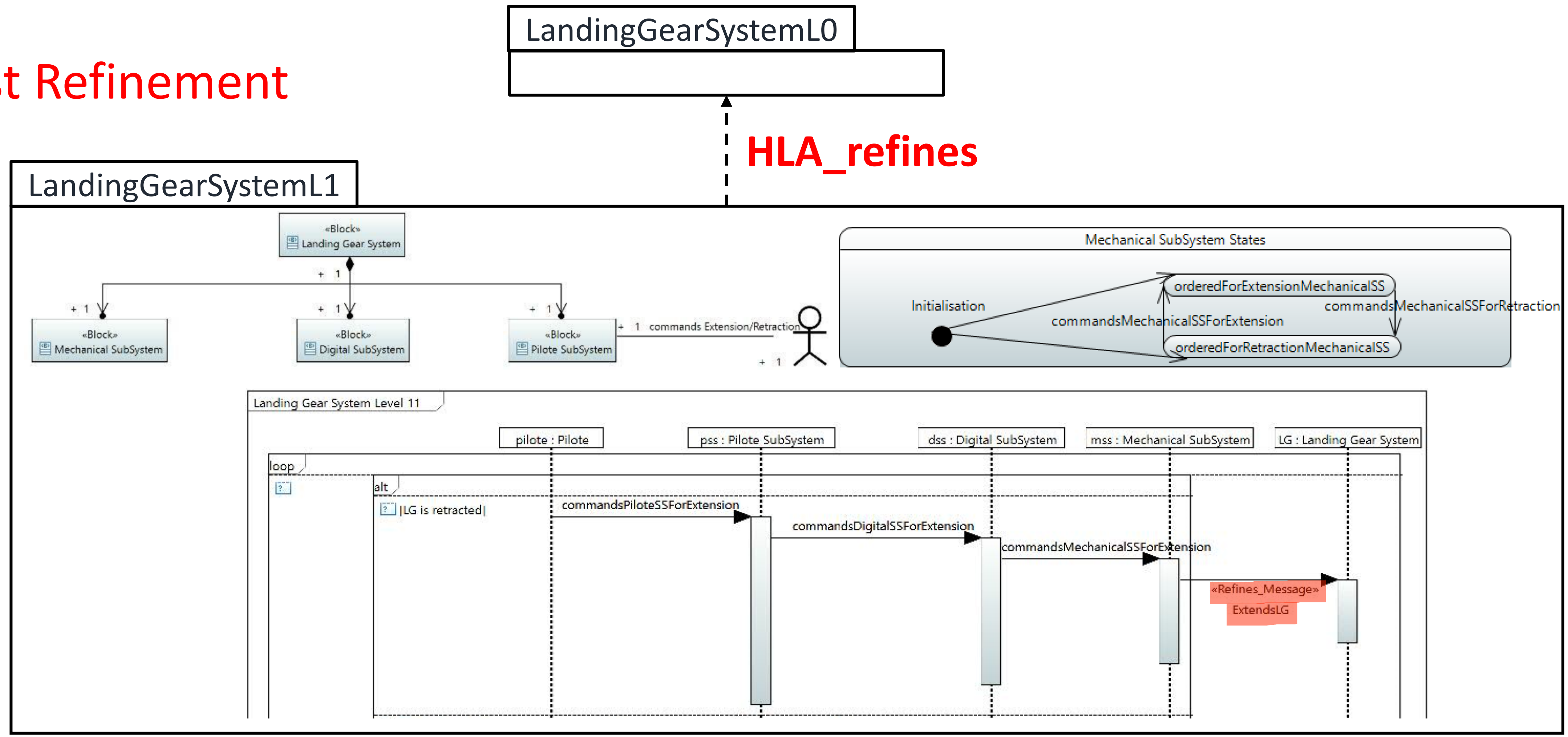
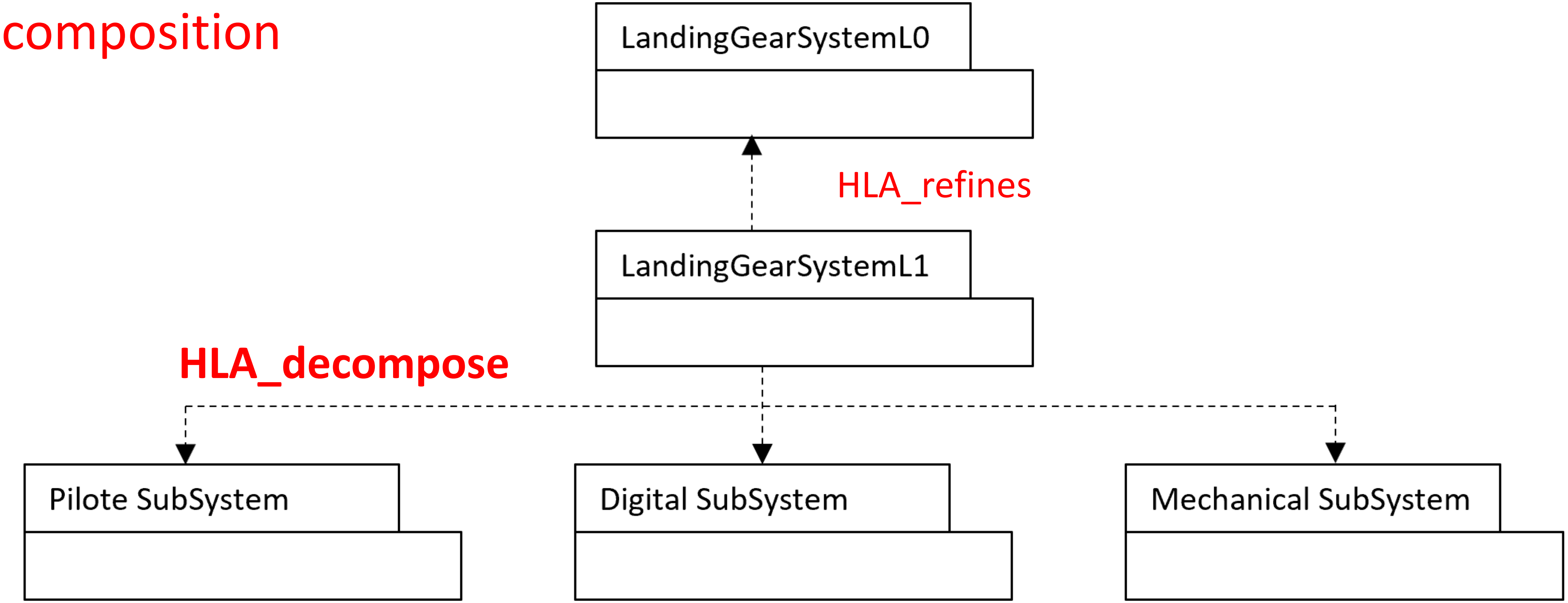
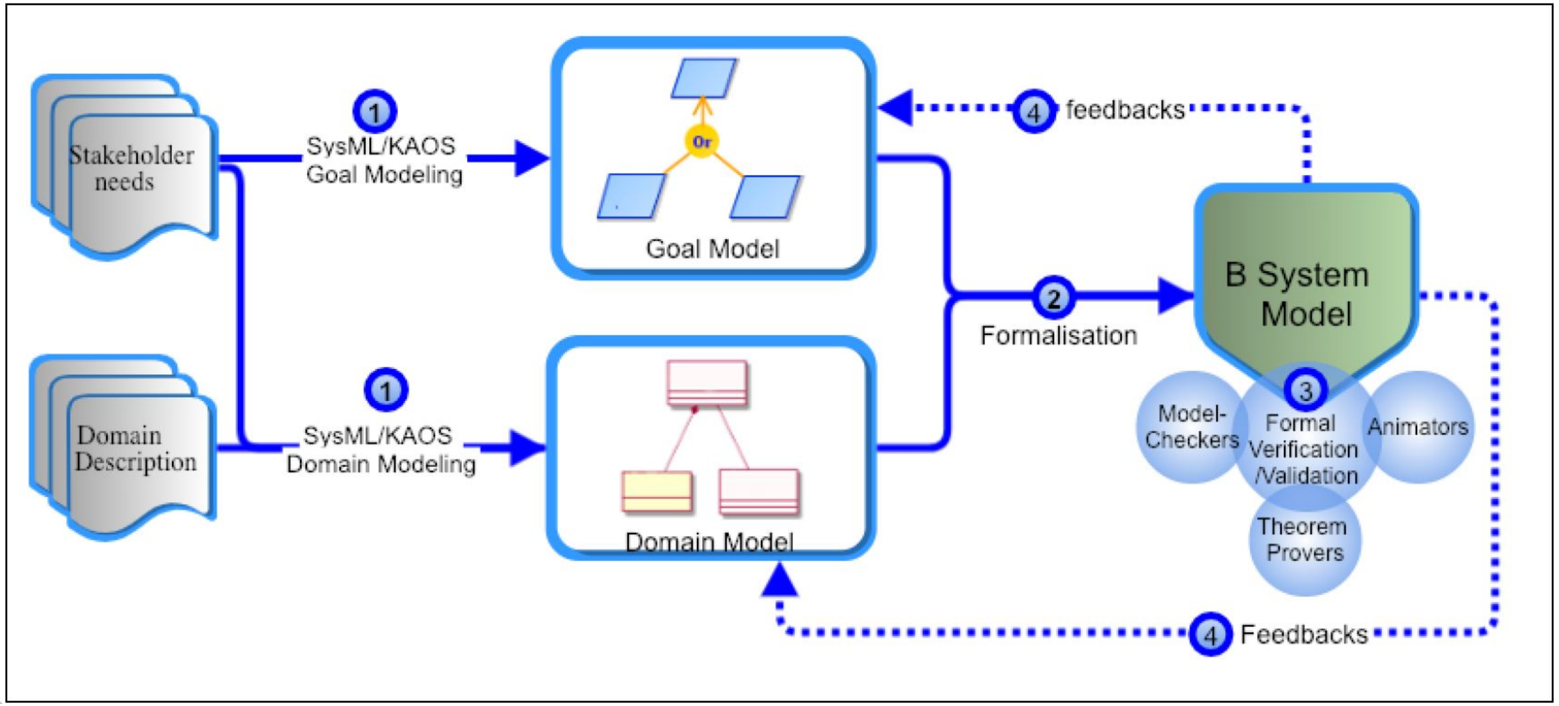


Illustration of the SysML extensions

First Decomposition

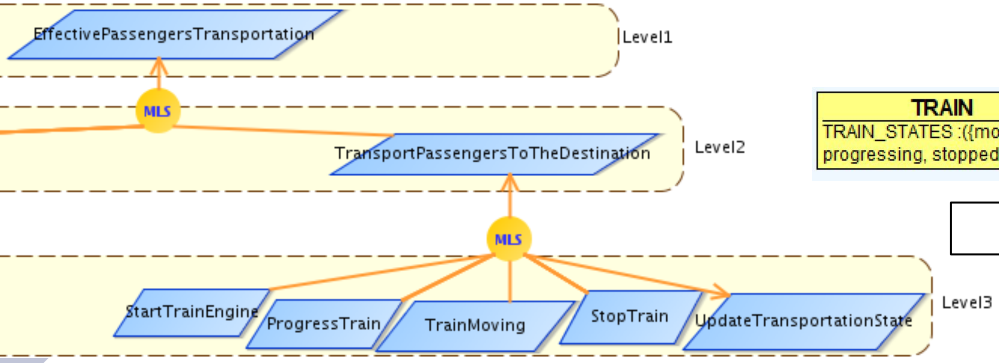


Modelling requirements: the SysML/KAOS approach

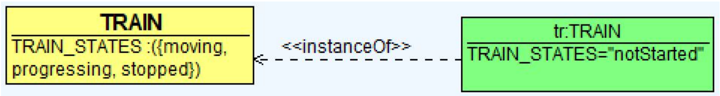



```
SYSTEM TrainControllerL3_CONT
SETS DESTINATIONS; TRAIN_ENGINE_STATES; TRAIN_STATES;
CONSTANTS progressing, moving, stopped, ...
PROPERTIES
…
TRAIN_STATES = {progressing, moving, stopped}
END
```

EXTRACT FROM THE EVENT-B SPECIFICATION OF SYSML/KAOS REQUIREMENTS MODEL Level 1



Goal Model



Domain Model

```
REFINEMENT TrainControllerL3
REFINES TrainControllerL2
SEES TrainControllerL1 CONT, TrainControllerL2 CONT, TrainControllerL3 CONT
VARIABLES trainState, ...
INVARIANTS trainState ∈ TRAIN → TRAIN STATES ∧ ...
INITIALISATION trainState :∈ TRAIN → {stopped} ‖ ...
EVENTS
...
ProgressTrain ref milestone TransportPassengersToTheDestination =
SELECT trainState(tr) = stopped THEN trainState(tr) := progressing
END;
...
StopTrain ref milestone TransportPassengersToTheDestination =
SELECT trainState(tr) = moving THEN
trainState(tr) := stopped END
END
```

EXTRACT FROM THE EVENT-B SPECIFICATION OF SYSML/KAOS REQUIREMENTS MODEL Level 2

- FOTSO, Steve Jeffrey Tueno, FRAPPIER, Marc, LALEAU, Regine, *et al.* Back propagating B system updates on SysML/KAOS domain models. In : *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2018. p. 160-169.

Aligning HLA with requirements

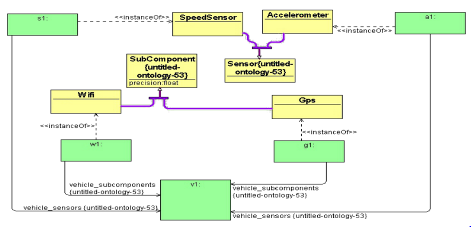
- Providing graphical alignment links between requirements models and HLA models
- Providing an automatic translation from graphical alignment links to Event-B specifications

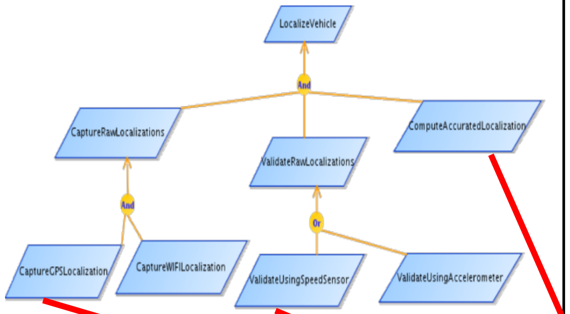
Aligning HLA with requirements

1. SysML/KAOS Modeling

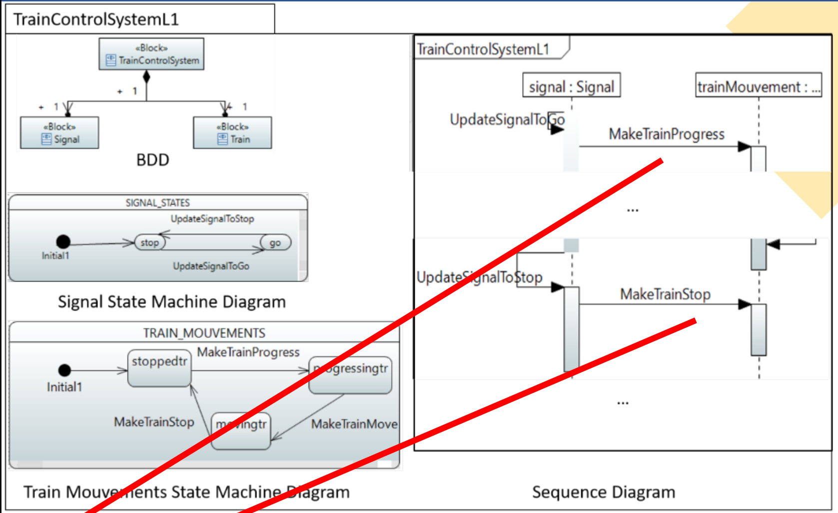
Domain Model

Goal Model





2. SysML HLA Modeling



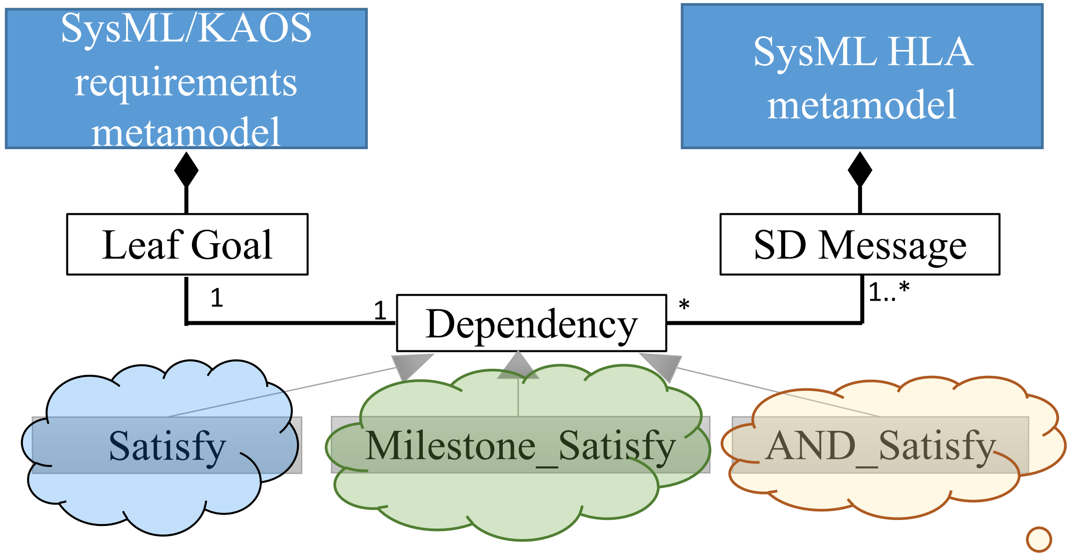
Graphical alignment links between requirements models and HLA models

3. Alignment

Automatic translation from graphical alignment links to Event-B specifications

Event-B Specification

Alignment links



Satisfy: allows to define an alignment link when a message can satisfy a goal.

Milestone_Satisfy: is defined when a sequence execution of a set of messages in a specific order is necessary to satisfy a goal.

AND_Satisfy: is defined when a goal is satisfied by a set of messages, i.e. the execution of all of them, in any order, is necessary to satisfy the goal.

Illustration with the Train control system case study: HLA modelling

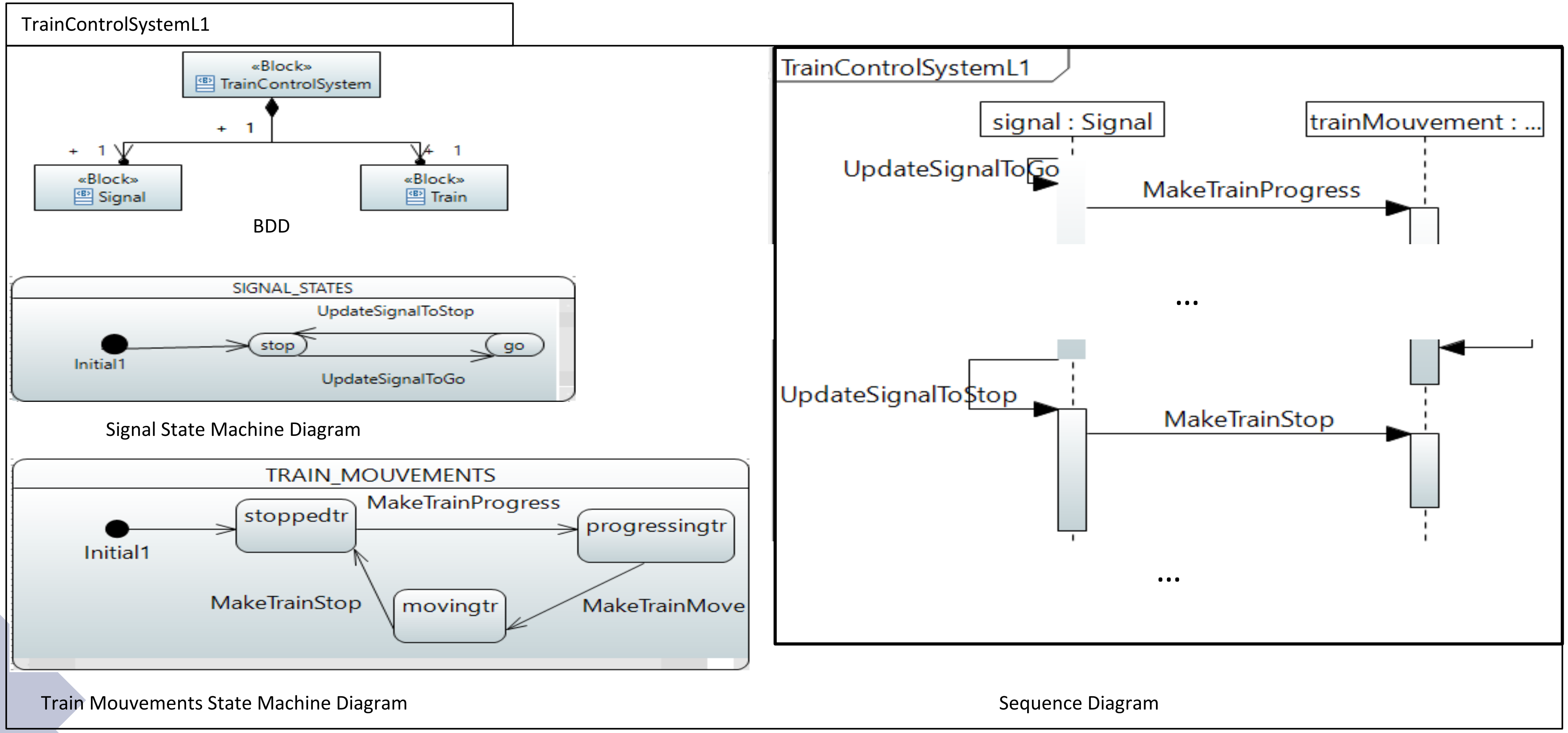
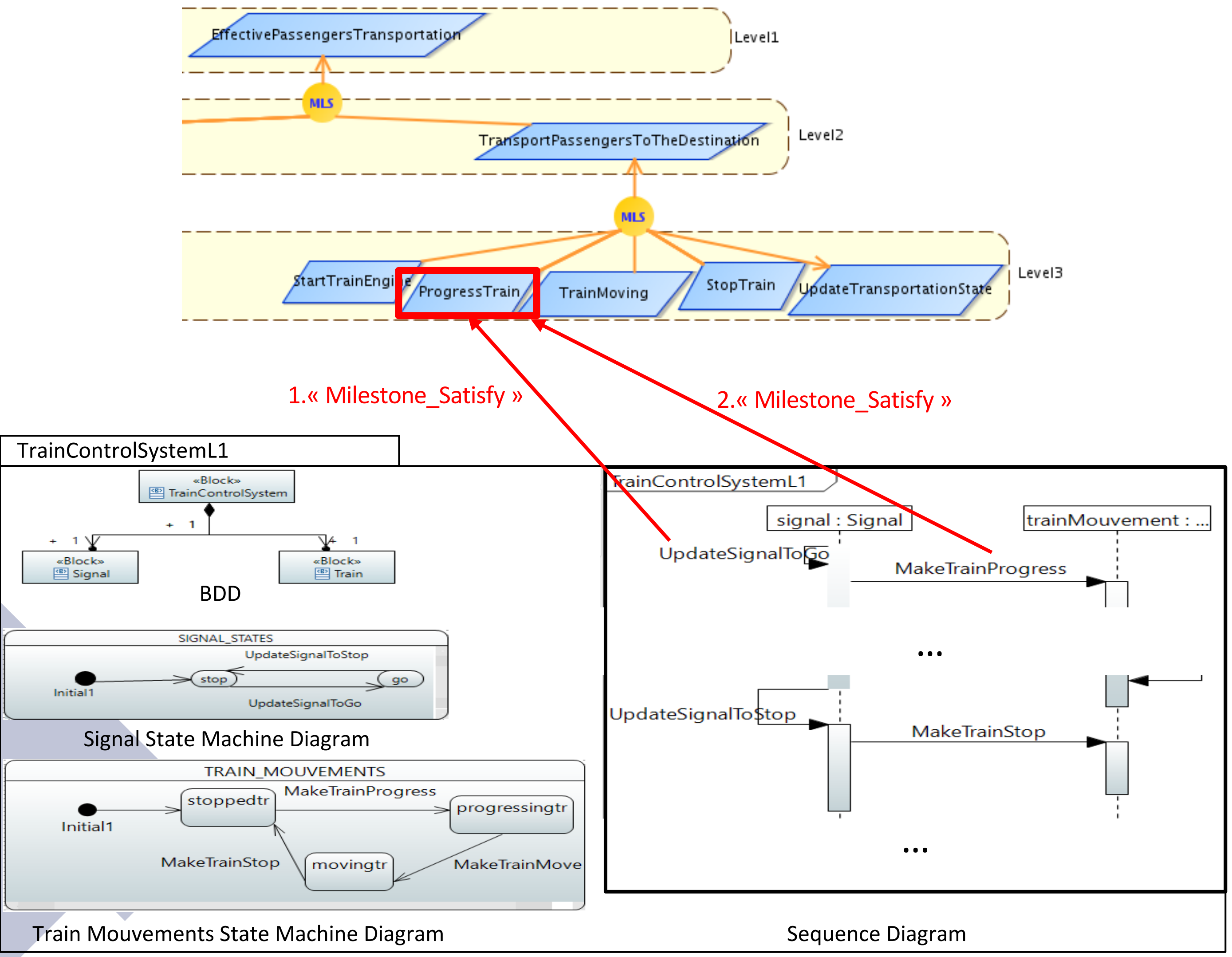
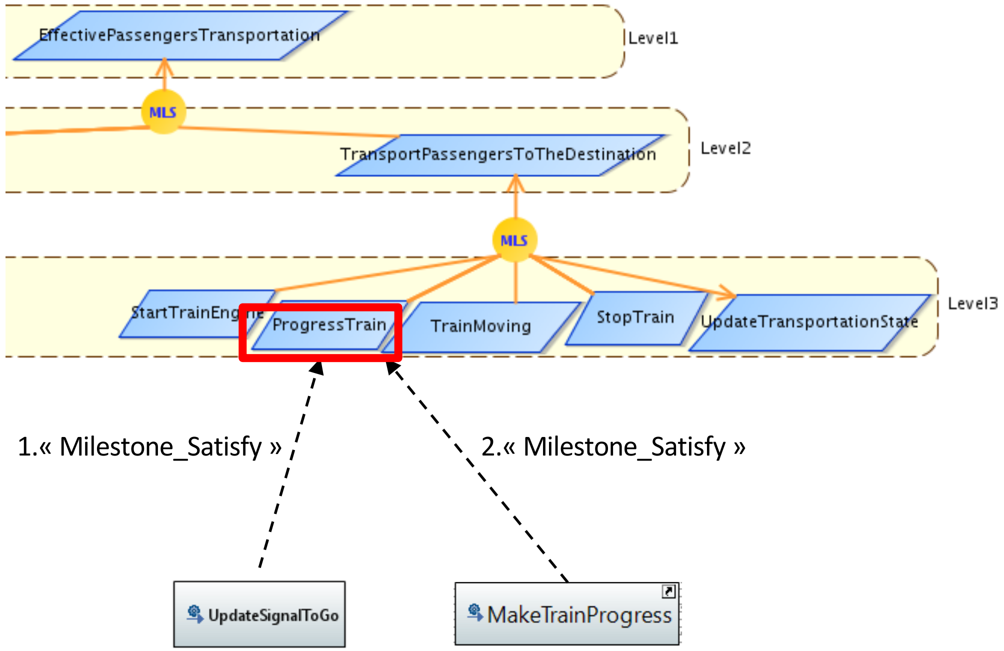


Illustration of the alignment links





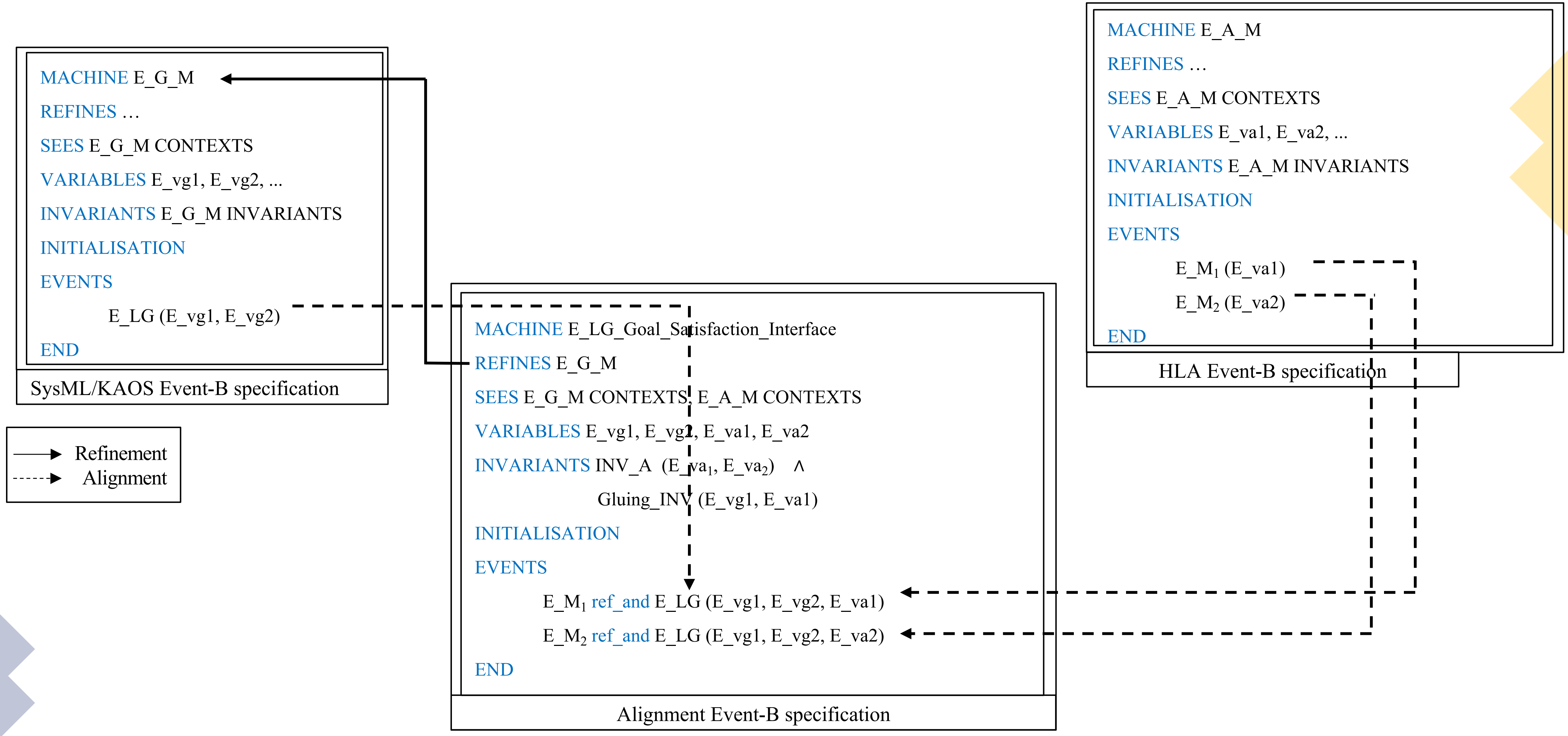
Formalization of graphical alignment links

- A refinement relationship between Event-B events from message and leaf goal,
  - × The Event-B refinement semantics do not match our alignment semantics,
  - × Messages that satisfy a leaf goal can belong to distinct Event-B machines,
- The formalization of SysML/KAOS models and SysML HLA models is performed in Event-B,

➢ A new Event-B machine is built for each alignment link,

➢ New sets of refinement proof obligations are specified, one for each type of alignment.

EVENT-B architecture of the proposed alignment



Formalization of graphical alignment links: some rules

		Source concepts		Target concepts	
Rule	Translation of	Element	Constraint	Element	Constraint
1	Leaf goal to satisfy	LG E_LG	LG is the leaf goal E_LG is the event related to LG E_G_M is the Event-B machine that contains E_LG	E_LG_Goal_Satisfaction_Interface	E_LG_Goal_Satisfaction_Interface ∈ MACHINE; E_LG_Goal_Satisfaction_Interface REFINES E_G_M; E_LG_Goal_Satisfaction_Interface SEES E_G_M CONTEXTS;
2	Variables involved in E_LG	E_vg_i, $i \in [1..n]$	E_vg_i are the variables of E_G_M involved in E_LG	E_vg_i, $i \in [1..n]$	E_vg_i, ∈ VARIABLES
3	Messages responsible for the satisfaction of the leaf goal LG	M_j E_M_j $j \in [1..p]$	M_1, ... M_p are messages E_M_j is the event related to message M_j ; $E\_A\_M_j$ is the Event-B machine that contains E_M_j	E_M_j $j \in [1..p]$	E_M_j ∈ EVENTS E_LG_Goal_Satisfaction_Interface SEES $E\_A\_M_j$ CONTEXTS
4	Variables involved in E_M_j events	$E\_va_{j,k}$ $j \in [1..p]$ $k \in [1..q]$	$E\_va_{j,k}$ are the variables involved in E_M_j $INV\_A\_M_j(E\_va_{j,1}, ...,E\_va_{j,q})$ is the part of $E\_A\_M_j$ invariant related to the $E\_va_{j,k}$ variables	$E\_va_{j,k}$, $j \in [1..p]$ $k \in [1..q]$	$E\_va_{j,k}$ ∈ VARIABLES $INV\_A\_M_j(E\_va_{j,1}, ..., E\_va_{j,q})$ ∈ INVARIANTS

Formalization of graphical alignment links: Proof obligations

- Gluing invariant are constructed in the machine E_LG_Goal_Satisfaction_Interface to links variables of the HLA EVENT-B machines $E\_A\_M_j$ to variables of the SYSML/KAOS EVENT-B machine E_G_M.
- **Alignment proof obligations:**

Let LG be a leaf goal and M1, M2 two messages. Let E_LG be the event associated to LG and E_M_1, E_M_2 the two events associated to M1 and M2. Each event E is of the form:

E = **SELECT** E_Guard **THEN** E_Post.

- Satisfy relationship. Assume that LG is satisfied by M1, then:

E_M_1 ref E_LG

where ref is the standard EVENT-B refinement.

This proof obligation ensures that the execution of E_M_1 implies the satisfaction of E_LG.

Formalization of graphical alignment links: And_Satisfy alignment

- And_Satisfy relationship. Assume that LG is satisfied by M1 and M2, then:

$$E\_M_1 \text{ ref\_and } E\_LG$$

$$E\_M_2 \text{ ref\_and } E\_LG$$

- New proof obligations are generated:

$$E\_M_1\_Guard \Rightarrow E\_LG\_Guard$$

$$E\_M_2\_Guard \Rightarrow E\_LG\_Guard$$

$$(E\_M_1\_Post \land E\_M_2\_Post) \Rightarrow E\_LG\_Post$$

  - These proof obligations ensure firstly that $E\_M_1\_Guard$ and $E\_M_2\_Guard$ should never contradict E_LG_Guard and secondly the execution of E_M_1 and E_M_2 without any specific order implies the satisfaction of E_LG.

Formalization of graphical alignment links: Milestone_Satisfy alignment

- Milestone_Satisfy relationship. Assume that LG is satisfied by the sequential execution of M1 and M2, then:

$$E\_M_1 \text{ ref\_milestone } E\_LG$$

$$E\_M_2 \text{ ref\_milestone } E\_LG$$

- New proof obligations are generated:

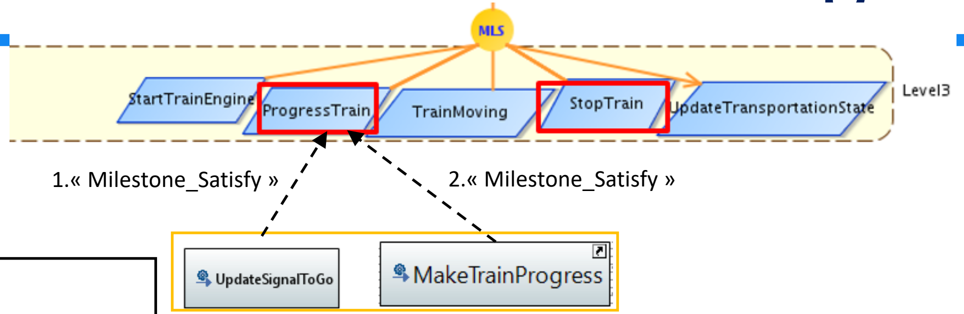
$E\_M_1\_Guard \Rightarrow E\_LG\_Guard$

$E\_M_1\_Post \Rightarrow E\ M_2\_Guard$

$E\_M_2\_Post \Rightarrow E\_LG\_Post$

➢ These proof obligations ensure firstly that $E\_M_1\_Guard$ should never contradict E_LG_Guard. Secondly the scheduling constraint should be respected with $E\_M_1\_Post$ implies $E\_M_2\_Guard$ and finally the execution of E_M_1 followed by the execution of E_M_2 implies the satisfaction of E_LG

Illustration of the formalization the alignment links



REFINEMENT TrainControllerL3
Refines TrainControllerL2
SEES TrainControllerL1_CONT, TrainControllerL2_CONT, TrainControllerL3_CONT
VARIABLES trainState , ...
...
EVENTS
ProgressTrain ref_milestone TransportPassengersToTheDestination =
SELECT trainState (tr) = stopped THEN trainState (tr) := progressing
END ;
...
END

EXTRACT FROM THE EVENT-B SPECIFICATION OF SYSML/KAOS REQUIREMENTS MODEL Level 2

REFINEMENT TrainControlSystemL1
REFINES TrainControlSystemL0
SEES TrainControlSystemL1_CONT , TrainControlSystemL0_CONT
VARIABLES signalState , trainMouvementState
...
EVENTS
UpdateSignalToGo =
SELECT signalState = stop ∧ trainMouvementState (train)= stoppedtr THEN signalState := go END ;

MakeTrainProgress =
SELECT signalState = go ∧ trainMouvementState (train)= stoppedtr THEN
trainMouvementState (train):= progressingtr END ;
...
END

EXTRACT FROM THE EVENT-B SPECIFICATION OF HLA

REFINEMENT ProgressTrain Goal Satisfaction Interface
REFINES TrainControllerL3
SEES TrainControllerL1 CONT, TrainControllerL2 CONT, TrainControllerL3 CONT, TrainControlSystemL1 CONT
VARIABLES ..., trainState, signalState, trainMouvementState
INVARIANTS
signalState ∈ SIGNAL STATES ∧ trainMouvementState ∈ TRAINS → TRAIN MOUVEMENTS
∧ (trainMouvementState[TRAINS] ={stoppedtr} ⇒ trainState[TRAIN]={stopped}) \\gluing invariant
∧ (trainMouvementState[TRAINS] ={progressingtr} ⇒ trainState[TRAIN]={progressing}) \\gluing invariant
∧ (trainMouvementState[TRAINS] ={movingtr} ⇒ trainState[TRAIN]={moving}) \\gluing invariant
INITIALISATION
...
signalState :∈ {stop} ||
trainState, trainMouvementState : ((trainState ∈ TRAIN → {stopped}) ∧ (trainMouvementState :∈ TRAINS → {stoppedtr})
∧ (trainMouvementState[TRAINS] ={stoppedtr} ⇒ trainState[TRAIN]={stopped})
∧ (trainMouvementState[TRAINS] ={progressingtr} ⇒ trainState[TRAIN]={progressing})
∧ (trainMouvementState[TRAINS] ={movingtr} ⇒ trainState[TRAIN]={moving}))
EVENTS
UpdateSignalToGo ref_milestone ProgressTrain =
SELECT signalState = stop ∧ trainMouvementState(train)=stoppedtr ∧ trainState(tr)=stopped THEN
signalState := go END;
MakeTrainProgress ref_milestone ProgressTrain =
SELECT signalState = go ∧ trainMouvementState(train)=stoppedtr THEN
trainMouvementState(train):=progressingtr || trainState(tr):=progressing END
END

EVENT-B FORMALIZATION OF THE GOAL PROGRESSTRAIN ALIGNMENT LINK

Conclusion

- A model-based approach to align complex systems HLA models with SYSML/KAOS requirements models:
  - Graphical to specify the alignment of a leaf goal with HLA elements responsible for its satisfaction,
  - Formal to verify the alignment links in Event-B.
- A set of transformation rules to translate graphical alignment links into Event-B specifications
- Event-B specifications are formally analysed and proved using AtelierB
- A tool has been developed using QVT

Future work

- Propagating the impact of updates on requirements models and/or HLA models on other models and on established alignment links.
- Integrating security and safety properties specification and traceability down to Event-B models (*new on-going PhD*)
- Formal definition of our metamodels and verification of the translation rules. This could be carried out by using EB4EB.

Thanks for your attention

Questions ???