Modeling the Hybrid ERTMS/ETCS Level 3 Standard Using a Formal Requirements Engineering Approach

Steve Tueno[1,2], Marc Frappier[1], Régine Laleau[2], Amel Mammar[3]

[1]GRIL – Université de Sherbrooke, Canada
[2]LACL – Université Paris Est Créteil Val de Marne, France
[3]SAMOVAR-CNRS – Télécom SudParis, France

ABZ, Southampton, UK *(June 5th-8th, 2018)*

Outline

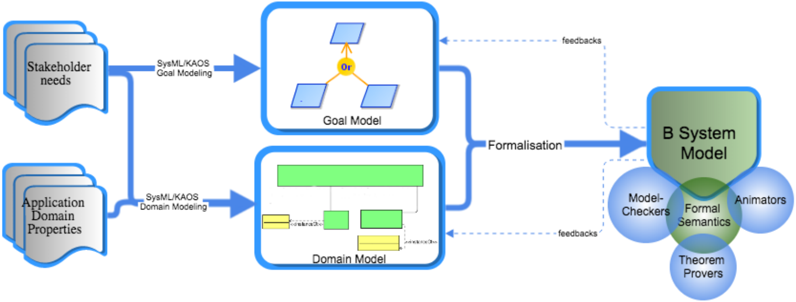
SysML/KAOS Requirements Engineering Method

FORMOSE project (ANR-14-CE28-0009)

FORMOSE: Formal Requirements Modeling for Critical Complex Systems, Method and Toolkit

Mission

Build methods and toolsets for the formal requirements modeling of critical and complex systems



SysML/KAOS Goal Modeling Language

The SysML/KAOS goal modeling language combines the traceability features provided by ***SysML*** with goal expressiveness provided by ***KAOS***

Requirements models are goal hierarchies built through a succession of refinements using different operators:

- the ***AND* operator**: all the subgoals must be achieved to realise the parent goal
- the ***OR* operator**: the achievement of only one subgoal is sufficient to realise the parent goal
- the ***MILESTONE* operator**: all the subgoals must be achieved, following an achievement order, to realise the parent goal

SysML/KAOS Goal Modeling Language

The SysML/KAOS goal modeling language combines the traceability features provided by ***SysML*** with goal expressiveness provided by ***KAOS***

Requirements models are goal hierarchies built through a succession of refinements using different operators:

- the ***AND*** **operator**: all the subgoals must be achieved to realise the parent goal
- the ***OR*** **operator**: the achievement of only one subgoal is sufficient to realise the parent goal
- the ***MILESTONE*** **operator**: all the subgoals must be achieved, following an achievement order, to realise the parent goal

SysML/KAOS Goal Modeling Language

The SysML/KAOS goal modeling language combines the traceability features provided by ***SysML*** with goal expressiveness provided by ***KAOS***

Requirements models are goal hierarchies built through a succession of refinements using different operators:

- the ***AND*** **operator**: all the subgoals must be achieved to realise the parent goal
- the ***OR*** **operator**: the achievement of only one subgoal is sufficient to realise the parent goal
- the ***MILESTONE*** **operator**: all the subgoals must be achieved, following an achievement order, to realise the parent goal

SysML/KAOS Domain Modeling Language

The SysML/KAOS domain modeling language combines

- the expressivity of the ***Ontology Web Language (OWL)***
- the constraints provided by the standard ***Part Library (PLIB)***
- and the **extensions** needed to guarantee some relevant properties

- Each **domain model** corresponds to a **refinement level** in the SysML/KAOS **goal model**.
- **Domain models can be linked together** to form a hierarchy.

Formalisation of a *SysML/KAOS* Goal Diagram

- Each goal diagram **refinement level** gives a *B System* **component**.
- Each **goal** gives an **"event"**.
- Refinement **links between goals** give refinement **links between** *B system* **components** and **new proof obligations**.

For instance[1]:

MILESTONE operator

- $G_1\_Guard \Rightarrow G\_Guard$
- $G_2\_Post \Rightarrow G\_Post$
- $\Box(G1\_Post \Rightarrow \Diamond G2\_Guard)$: each **state**, corresponding to $G1\_Post$, must be **followed**, **at least once in the future**, by a **state** enabling G_2

[1] Goal ***G*** **refined** into ***G1*** and ***G2***

Formalisation of a *SysML/KAOS* Goal Diagram

- Each goal diagram **refinement level** gives a *B System* **component**.
- Each **goal** gives an **"event"**.
- Refinement **links between goals** give refinement **links between** *B system* **components** and **new proof obligations**.

For instance[1]:

MILESTONE **operator**

- $G_1\_Guard \Rightarrow G\_Guard$
- $G_2\_Post \Rightarrow G\_Post$
- $\square(G1\_Post \Rightarrow \lozenge G2\_Guard)$: each **state**, corresponding to $G1\_Post$, must be **followed**, **at least once in the future**, by a **state** enabling G_2

[1] Goal ***G*** **refined** into ***G1*** and ***G2***

Formalisation of a *SysML/KAOS* Goal Diagram

- Each goal diagram **refinement level** gives a *B System* **component**.
- Each **goal** gives an **"event"**.
- Refinement **links between goals** give refinement **links between *B system* components** and **new proof obligations**.

For instance[1]:

MILESTONE operator

- $G_1\_Guard \Rightarrow G\_Guard$
- $G_2\_Post \Rightarrow G\_Post$
- $\Box(G1\_Post \Rightarrow \Diamond G2\_Guard)$: each **state**, corresponding to $G1\_Post$, must be **followed**, **at least once in the future**, by a **state** enabling G_2

[1] Goal ***G*** **refined** into ***G1*** and ***G2***

Formalisation of *SysML/KAOS* Domain Models

The translation rules have been defined and formally verified. For instance:

- **Abstract concepts** give *B System* **sets**.
- **Concrete concepts** give *B System* **subsets**.
- **Attributes** give *B System* **relations**.
- **Individual** and **data values** give *B System* **set items**.

Outline

Main Characteristics of hybrid ERTMS/ETCS Level 3

The Aim

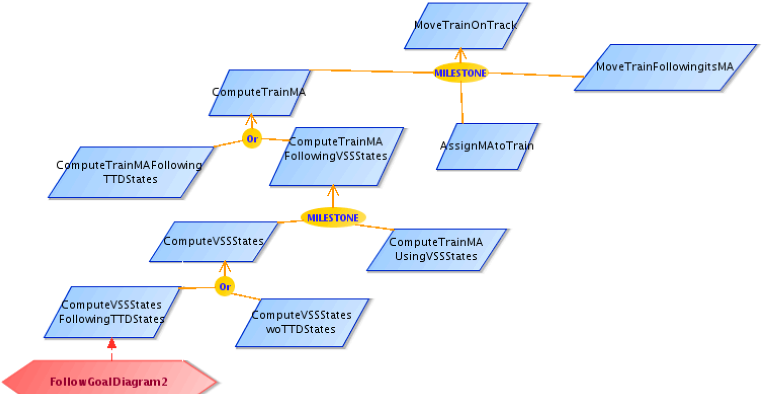
Optimize the use and occupation of railways.

- A **TTD** can be ***free*** or ***occupied***.
- A **VSS** can additionnally be in the ***unknown*** state (0..1) or in the ***ambiguous*** state (1..).
- **Each train can be assigned a Movement Authority (MA)** which is a portion of the track on which it is guaranteed to move safely.

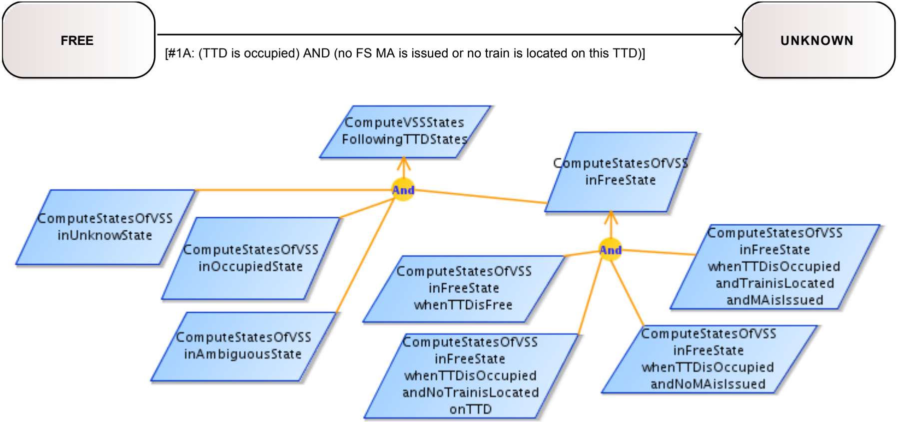
The Main SysML/KAOS Goal Model

A very first contribution of our work

Attempt to **place standard requirements** in view with more **abstract/high-level goals** in a **methodological way**: Trains move and have assigned MA that they must respect.



Goals coming from the requirements of transition *#1A*



Domain Model of the Root Level

ASM 1 (ref. case study description)

- **Track** is a **straight line** : $a < b \land TRACK = a\,..\,b$

```
custom data set TRACK data value a,b type: NATURAL
```

- (*REQ 11* .. *REQ 14*) Trains travel in the **same direction** and can be **connected**: $\forall\, tr.(tr \in dom(rear) \Rightarrow rear(tr) < front(tr))$
  - (*REQ 11* .. *REQ 13*) Each **connected** train broadcasts its **front**
  - (*REQ 12*) **Connected TIMS** trains broadcast their **rear**

Domain Model of the Root Level

ASM 1 (ref. case study description)

- **Track** is a **straight line** : $a < b \wedge TRACK = a\,..\,b$

  `custom data set TRACK data value a,b type: NATURAL`

- (***REQ 11*** .. ***REQ 14***) Trains travel in the **same direction** and can be **connected**: $\forall\, tr.(tr \in dom(rear) \Rightarrow rear(tr) < front(tr))$
  - (***REQ 11*** .. ***REQ 13***) Each **connected** train broadcasts its **front**
  - (***REQ 12***) **Connected TIMS** trains broadcast their **rear**

B System Specification of the Root Level (1/2)

The domain model gives the structural part of the formal specification

SETS TRAIN
CONSTANTS a b TRACK
PROPERTIES

axm1: $a \in \mathbb{N}$ axm2: $b \in \mathbb{N}$ p0.1: $a < b$ p0.2: $TRACK = a\,..\,b$

VARIABLES connectedTrain front rear
INVARIANT

inv1: $connectedTrain \in TRAIN \rightarrowtail\!\!\!\!+ BOOL$
inv2: $front \in dom(connectedTrain) \rightarrow TRACK$
inv3: $rear \in dom(connectedTrain) \rightarrowtail\!\!\!\!+ TRACK$
p0.3: $\forall\, tr \cdot (tr \in dom(rear) \Rightarrow rear(tr) < front(tr))$

B System Specification of the Root Level (2/2)

The goal model gives the behaviour of the formal specification

Event MoveTrainOnTrack $\widehat{=}$
any tr len
where
grd1: $tr \in connectedTrain^{-1}[\{TRUE\}]$
grd2: $len \in \mathbb{N}_1$
grd3: $front(tr) + len \in TRACK$
then
act1: $front(tr) := front(tr) + len$
act2: $rear := (\{TRUE \mapsto rear \lhd\!\!- \{tr \mapsto rear(tr) + len\}, FALSE \mapsto rear\})(bool(tr \in dom(rear)))$[a]
END

[a] The system only observe the update of the rear of TIMS trains

Domain Model of the First Refinement Level

- The **Movement Authority** (MA) of a train is a **contiguous part** of the track:
  $\forall tr \cdot (tr \in dom(MA) \Rightarrow (\exists p, q \cdot (p \,..\, q \subseteq TRACK \wedge p \leq q \wedge MA(tr) = p \,..\, q)))$

```
attribute MA domain: dom(connectedTrain) range: POW(TRACK) {
 is variable: true is functional: true is total: false
}
```

- containing the train:
  - (*REQ 13*) **ERTMS train**:
    $\forall tr \cdot (tr \in dom(MA) \Rightarrow front(tr) \in MA(tr))$
  - (*REQ 12*) **TIMS train**:
    $\forall tr \cdot (tr \in dom(rear) \cap dom(MA) \Rightarrow rear(tr) \in MA(tr))$
- (*ASM 17*) **MA assigned** to two **different trains** must be **disjoint**: $\forall tr1, tr2 \cdot ((\{tr1, tr2\} \subseteq dom(MA) \wedge tr1 \neq tr2)$
  $\Rightarrow MA(tr1) \cap MA(tr2) = \varnothing)$

Domain Model of the First Refinement Level

- The **Movement Authority** (MA) of a train is a **contiguous part** of the track:
  $\forall tr \cdot (tr \in dom(MA) \Rightarrow (\exists p, q \cdot (p \,..\, q \subseteq TRACK \land p \leq q \land MA(tr) = p \,..\, q)))$

```
attribute MA domain: dom(connectedTrain) range: POW(TRACK) {
 is variable: true is functional: true is total: false
}
```

- containing the train:
  - **(*REQ 13*) ERTMS train**:
    $\forall tr \cdot (tr \in dom(MA) \Rightarrow front(tr) \in MA(tr))$
  - **(*REQ 12*) TIMS train**:
    $\forall tr \cdot (tr \in dom(rear) \cap dom(MA) \Rightarrow rear(tr) \in MA(tr))$
- (*ASM 17*) **MA assigned** to two **different trains** must be **disjoint**: $\forall tr1, tr2 \cdot ((\{tr1, tr2\} \subseteq dom(MA) \land tr1 \neq tr2)$
  $\Rightarrow MA(tr1) \cap MA(tr2) = \varnothing)$

Domain Model of the First Refinement Level

- The **Movement Authority** (MA) of a train is a **contiguous part** of the track:
  $\forall tr \cdot (tr \in dom(MA) \Rightarrow (\exists p, q \cdot (p \,..\, q \subseteq TRACK \wedge p \leq q \wedge MA(tr) = p \,..\, q)))$

```
attribute MA domain: dom(connectedTrain) range: POW(TRACK) {
 is variable: true is functional: true is total: false
}
```

- containing the train:
  - (*REQ 13*) **ERTMS train**:
    $\forall tr \cdot (tr \in dom(MA) \Rightarrow front(tr) \in MA(tr))$
  - (*REQ 12*) **TIMS train**:
    $\forall tr \cdot (tr \in dom(rear) \cap dom(MA) \Rightarrow rear(tr) \in MA(tr))$
- (*ASM 17*) **MA assigned** to two **different trains** must be **disjoint**: $\forall tr1, tr2 \cdot ((\{tr1, tr2\} \subseteq dom(MA) \wedge tr1 \neq tr2)$
  $\Rightarrow MA(tr1) \cap MA(tr2) = \varnothing)$

B System Specification of the First Refinement Level (1/2)

- It contains the **result** of the **translation** of the **domain model**
- It defines theorems representing the **proof obligations related to the *MILESTONE* operator**: to ensure the sequencing of events

theorem s1: $ComputeTrainMA\_Guard \Rightarrow MoveTrainOnTrack\_Guard$
theorem s2: $ComputeTrainMA\_Post \Rightarrow AssignMAtoTrain\_Guard$
theorem s3: $AssignMAtoTrain\_Post \Rightarrow MoveTrainFollowingItsMA\_Guard$
theorem s4: $MoveTrainFollowingItsMA\_Post \Rightarrow MoveTrainOnTrack\_Post$

- $(\Box(G1\_Post \Rightarrow \Diamond G2\_Guard))$ **replaced by** $(G1\_Post \Rightarrow G2\_Guard)$

B System Specification of the First Refinement Level (1/2)

- It contains the **result** of the **translation** of the **domain model**
- It defines theorems representing the **proof obligations related to the *MILESTONE* operator**: to ensure the sequencing of events

theorem s1: $ComputeTrainMA\_Guard \Rightarrow MoveTrainOnTrack\_Guard$
theorem s2: $ComputeTrainMA\_Post \Rightarrow AssignMAtoTrain\_Guard$
theorem s3: $AssignMAtoTrain\_Post \Rightarrow MoveTrainFollowingItsMA\_Guard$
theorem s4: $MoveTrainFollowingItsMA\_Post \Rightarrow MoveTrainOnTrack\_Post$

- $(\Box(G1\_Post \Rightarrow \Diamond G2\_Guard))$ **replaced by** $(G1\_Post \Rightarrow G2\_Guard)$

B System Specification of the First Refinement Level (2/2)

Goals of the first refinement level **give "events"** of the formal specification:

- Events ComputeTrainMA and AssignMAtoTrain
  - **nondeterministically choose** a **contiguous** part of the track: $p\,..\,q \subseteq TRACK \wedge p \leq q$
  - **containing** the train: $front(tr) \in p\,..\,q \wedge (tr \in dom(rear) \Rightarrow rear(tr) \in p\,..\,q)$
  - **not part** of the MA of **another train**: $p\,..\,q \cap union(ran(\{tr\} \lhd\!\!\!- MA)) = \varnothing$
- Event MoveTrainFollowingItsMA constrains the movement of the train regarding its MA: $front(tr) + len \in MA(tr)$

B System Specification of the First Refinement Level (2/2)

Goals of the first refinement level **give "events"** of the formal specification:

- Events ComputeTrainMA and AssignMAtoTrain
  - **nondeterministically choose** a **contiguous** part of the track: $p \mathrel{..} q \subseteq TRACK \wedge p \leq q$
  - **containing** the train: $front(tr) \in p \mathrel{..} q \wedge (tr \in dom(rear) \Rightarrow rear(tr) \in p \mathrel{..} q)$
  - **not part** of the MA of **another train**: $p \mathrel{..} q \cap union(ran(\{tr\} \lhd\!\!\!- MA)) = \varnothing$
- Event MoveTrainFollowingItsMA constrains the movement of the train regarding its MA: $front(tr) + len \in MA(tr)$

B System Specification of the First Refinement Level (2/2)

Goals of the first refinement level **give "events"** of the formal specification:

- Events ComputeTrainMA and AssignMAtoTrain
  - **nondeterministically choose** a **contiguous** part of the track: $p \mathinner{..} q \subseteq TRACK \wedge p \leq q$
  - **containing** the train: $front(tr) \in p \mathinner{..} q \wedge (tr \in dom(rear) \Rightarrow rear(tr) \in p \mathinner{..} q)$
  - **not part** of the MA of **another train**: $p \mathinner{..} q \cap union(ran(\{tr\} \lessdot MA)) = \varnothing$
- Event MoveTrainFollowingItsMA constrains the movement of the train regarding its MA: $front(tr) + len \in MA(tr)$

Subsequent Refinement Levels (1/2)

The **same process** allows the **construction** of **others refinement levels**.

For instance,

- (***ASM 2*** .. ***ASM 4***, ***REQ 6*** .. ***REQ 8***) the **second refinement** level introduces **TTD, VSS, and their states** (*stateTTD* and *stateVSS*).
- (***ASM 5***) **A TTD where a train is located must be in the occupied state:**
  - (***REQ 13***) **non-TIMS train:**
    $\forall ttd, tr \cdot ((\boldsymbol{tr} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge ttd \in TTD \wedge \boldsymbol{front(tr)} \in \boldsymbol{ttd}) \Rightarrow \boldsymbol{stateTTD(ttd){=}OCCUPIED})$
  - (***REQ 12***) **TIMS train:**
    $\forall ttd, tr \cdot ((\boldsymbol{tr} \in \boldsymbol{dom(rear)} \wedge ttd \in TTD \wedge \boldsymbol{(rear(tr)\,..\,front(tr))} \cap \boldsymbol{ttd} \neq \varnothing) \Rightarrow \boldsymbol{stateTTD(ttd){=}OCCUPIED})$

Subsequent Refinement Levels (1/2)

The **same process** allows the **construction** of **others refinement levels**.

For instance,

- (***ASM 2*** .. ***ASM 4***, ***REQ 6*** .. ***REQ 8***) the **second refinement** level introduces **TTD, VSS, and their states** (*stateTTD* and *stateVSS*).
- (***ASM 5***) A **TTD where a train is located must be in the occupied state**:
  - **(*REQ 13*) non-TIMS train:**
    $\forall ttd, tr \cdot ((\boldsymbol{tr} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \land ttd \in TTD \land \boldsymbol{front(tr)} \in \boldsymbol{ttd}) \Rightarrow \boldsymbol{stateTTD(ttd){=}OCCUPIED})$
  - **(*REQ 12*) TIMS train:**
    $\forall ttd, tr \cdot ((\boldsymbol{tr} \in \boldsymbol{dom(rear)} \land ttd \in TTD \land \boldsymbol{(rear(tr)\,..\,front(tr)) \cap ttd \neq \varnothing}) \Rightarrow \boldsymbol{stateTTD(ttd){=}OCCUPIED})$

Subsequent Refinement Levels (2/2)

- **Train locations must respect the safety constraints:**
  - **(*REQ 12*) Two TIMS trains must be on disjoint track segments:**
    $\forall\, tr1, tr2 \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(rear)} \wedge \boldsymbol{tr2} \in \boldsymbol{dom(rear)} \wedge tr1 \neq tr2)$
    $\Rightarrow \boldsymbol{(rear(tr1)..front(tr1)) \cap (rear(tr2)..front(tr2)) = \varnothing})$
  - (*REQ 12, REQ 13*) A non TIMS train can follow a TIMS train:
    $\forall\, tr1, tr2 \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(rear)} \wedge \boldsymbol{tr2} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge tr1 \neq tr2)$
    $\Rightarrow \boldsymbol{front(tr2)} < \boldsymbol{rear(tr1)})$
  - (*REQ 13*) Two non TIMS trains must be on different TTDs:
    $\forall\, tr1, tr2, ttd \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)}$
    $\wedge\ \boldsymbol{tr2} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge tr1 \neq tr2 \wedge ttd \in TTD$
    $\wedge\ \boldsymbol{front(tr1)} \in \boldsymbol{ttd}) \Rightarrow \boldsymbol{front(tr2)} \notin\ \boldsymbol{ttd})$

Subsequent Refinement Levels (2/2)

- **Train locations must respect the safety constraints**:
  - **(*REQ 12*) Two TIMS trains must be on disjoint track segments**:

    $\forall\, tr1, tr2 \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(rear)} \wedge \boldsymbol{tr2} \in \boldsymbol{dom(rear)} \wedge tr1 \neq tr2)$

    $\Rightarrow \boldsymbol{(rear(tr1)..front(tr1)) \cap (rear(tr2)..front(tr2)) = \varnothing})$
  - **(*REQ 12, REQ 13*) A non TIMS train can follow a TIMS train**:

    $\forall\, tr1, tr2 \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(rear)} \wedge \boldsymbol{tr2} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge tr1 \neq tr2)$

    $\Rightarrow \boldsymbol{front(tr2)} < \boldsymbol{rear(tr1)})$
  - **(*REQ 13*) Two non TIMS trains must be on different TTDs**:

    $\forall\, tr1, tr2, ttd \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)}$

    $\wedge\ \boldsymbol{tr2} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge tr1 \neq tr2 \wedge ttd \in TTD$

    $\wedge\ \boldsymbol{front(tr1)} \in \boldsymbol{ttd}) \Rightarrow \boldsymbol{front(tr2)} \notin \boldsymbol{ttd})$

Subsequent Refinement Levels (2/2)

- **Train locations must respect the safety constraints**:
  - **(*REQ 12*) Two TIMS trains must be on disjoint track segments**:
    $\forall\, tr1, tr2 \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(rear)} \wedge \boldsymbol{tr2} \in \boldsymbol{dom(rear)} \wedge tr1 \neq tr2)$
    $\Rightarrow \boldsymbol{(rear(tr1)..front(tr1)) \cap (rear(tr2)..front(tr2)) = \varnothing})$
  - **(*REQ 12, REQ 13*) A non TIMS train can follow a TIMS train**:
    $\forall\, tr1, tr2 \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(rear)} \wedge \boldsymbol{tr2} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge tr1 \neq tr2)$
    $\Rightarrow \boldsymbol{front(tr2)} < \boldsymbol{rear(tr1)})$
  - **(*REQ 13*) Two non TIMS trains must be on different TTDs**:
    $\forall\, tr1, tr2, ttd \cdot ((\boldsymbol{tr1} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)}$
    $\wedge\ \boldsymbol{tr2} \in \boldsymbol{dom(front)} \setminus \boldsymbol{dom(rear)} \wedge tr1 \neq tr2 \wedge ttd \in TTD$
    $\wedge\ \boldsymbol{front(tr1)} \in \boldsymbol{ttd}) \Rightarrow \boldsymbol{front(tr2)} \notin \boldsymbol{ttd})$

The **proof obligations** have been **discharged** using the **Rodin** tool extended with *Atelier B provers* and *SMT solvers*.

Refinement level	**L0**	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**
Invariants	**4**	**11**	**13**	**4**	**6**	**5**	**9**
Proof Obligations (PO)	**20**	**40**	**50**	**13**	**5**	**5**	**14**
Automatically Discharged POs	**17**	**30**	**30**	**11**	**0**	**0**	**4**
Interactively Discharged POs	**3**	**5**	**20**	**2**	**5**	**5**	**10**

It seemed that the **automatic provers** have **troubles** with:

- **data ranges** such as $p..q$
- **conditional actions** (*if-then-else*) such as $rear := (\{TRUE \mapsto rear \lhd\!\!- \{tr \mapsto rear(tr) + len\}, FALSE \mapsto rear\})(bool(tr \in dom(rear)))$

Outline

The use of the SysML/KAOS approach allows:

- To **bridge the gap** between the system **textual description** and its **formal specification**
- To better **delineate** the **system boundaries**
- A better **reusability** and **readability** of models
- The ability to **visualize** and **explore** the entire system and any part of it by **changing the focus and level of detail**
- A strong **traceability** between system **requirements** and **formal specification**

The use of the SysML/KAOS approach allows:

- To **bridge the gap** between the system **textual description** and its **formal specification**
- To better **delineate** the **system boundaries**
- A better **reusability** and **readability** of models
- The ability to **visualize** and **explore** the entire system and any part of it by **changing the focus and level of detail**
- A strong **traceability** between system **requirements** and **formal specification**

The use of the SysML/KAOS approach allows:

- To **bridge the gap** between the system **textual description** and its **formal specification**
- To better **delineate** the **system boundaries**
- A better **reusability** and **readability** of models
- The ability to **visualize** and **explore** the entire system and any part of it by **changing the focus and level of detail**
- A strong **traceability** between system **requirements** and **formal specification**

The use of the SysML/KAOS approach allows:

- To **bridge the gap** between the system **textual description** and its **formal specification**
- To better **delineate** the **system boundaries**
- A better **reusability** and **readability** of models
- The ability to **visualize** and **explore** the entire system and any part of it by **changing the focus and level of detail**
- A strong **traceability** between system **requirements** and **formal specification**

The use of the SysML/KAOS approach allows:

- To **bridge the gap** between the system **textual description** and its **formal specification**
- To better **delineate** the **system boundaries**
- A better **reusability** and **readability** of models
- The ability to **visualize** and **explore** the entire system and any part of it by **changing the focus and level of detail**
- A strong **traceability** between system **requirements** and **formal specification**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering is not enforced**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering is not enforced**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering is not enforced**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering is not enforced**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering is not enforced**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering** is **not enforced**

Another specification, using plain Event-B (1/2)

Use of plain Event-B, in the traditional style, by a distinct specifier

The specification includes **four refinement levels**:

- **TTDs** and **trains** are introduced in the **root level**
- **VSSs** are introduced in the **second** refinement **level**, as **refinements** of **TTDs**
- **MAs** and **VSS states** are introduced in the **third** refinement **level**
- **Safety properties** are introduced in the **fourth** refinement **level**
- A strategy is proposed to **prove the determinism of the transitions** of VSS states
- **Events and state variables are partitioned**: the environment view and the controller view
- The **execution ordering is not enforced**

Another specification, using plain Event-B (2/2)

With the SysML/KAOS Approach

The specification includes **seven refinement levels**:

- **Safety properties** are introduced in the **first** and **second** refinement **levels**
- ERTMS trains ***with or without*** *TIMS* are considered, so **a ERTMS train may or may not broadcast its rear**
- The **execution ordering** and the **refinement strategy** are **enforced** using **proof obligations** expressed as theorems

- The **SysML/KAOS** method represents a more **structured** and **methodological process** to the **formal specification** of the system
  - it makes it possible to **trace the source** and **justify the need** for each formal component and its contents, in relation with SysML/KAOS models

Another specification, using plain Event-B (2/2)

With the SysML/KAOS Approach

The specification includes **seven refinement levels**:

- **Safety properties** are introduced in the **first** and **second** refinement **levels**
- ERTMS trains ***with or without*** *TIMS* are considered, so **a ERTMS train may or may not broadcast its rear**
- The **execution ordering** and the **refinement strategy** are **enforced** using **proof obligations** expressed as theorems

- The **SysML/KAOS** method represents a more **structured** and **methodological process** to the **formal specification** of the system
  - it makes it possible to **trace the source** and **justify the need** for each formal component and its contents, in relation with SysML/KAOS models

Another specification, using plain Event-B (2/2)

With the SysML/KAOS Approach

The specification includes **seven refinement levels**:

- **Safety properties** are introduced in the **first** and **second** refinement **levels**
- ERTMS trains ***with or without*** *TIMS* are considered, so **a ERTMS train may or may not broadcast its rear**
- The **execution ordering** and the **refinement strategy** are **enforced** using **proof obligations** expressed as theorems

- The **SysML/KAOS** method represents a more **structured** and **methodological process** to the **formal specification** of the system
  - it makes it possible to **trace the source** and **justify the need** for each formal component and its contents, in relation with SysML/KAOS models

Another specification, using plain Event-B (2/2)

With the SysML/KAOS Approach

The specification includes **seven refinement levels**:

- **Safety properties** are introduced in the **first** and **second** refinement **levels**
- ERTMS trains ***with or without*** *TIMS* are considered, so **a ERTMS train may or may not broadcast its rear**
- The **execution ordering** and the **refinement strategy** are **enforced** using **proof obligations** expressed as theorems

- The **SysML/KAOS** method represents a more **structured** and **methodological process** to the **formal specification** of the system
  - it makes it possible to **trace the source** and **justify the need** for each formal component and its contents, in relation with SysML/KAOS models

Future Work

1. **Improve the representation of domain predicates** (to make them more user-friendly)

2. **Evaluate the impact of updates on *B System* specifications within domain models**