DB NETZE

neovendi
projects | engineering | digital transformation

Formal Verification of EULYNX Models Using Event-B and RODIN

Shubhangi Salunkhe, Randolf Berglehner, Abdul Rasheeq
08/06/2021

Railway Signalling System

- **Railway Signalling System:** A system used to direct railway traffic and keep trains clear of each other all the time.
- **Interlocking(IXL):** Middle layer and responsible for monitoring the train route by establishing safety. In few words, Railway Interlocking is basically designed to:
  1. Ensure that any route is safe to operate over.
  2. Maintain the route at which a train is approaching or using.
  3. Ensure the system fails to a safe state.

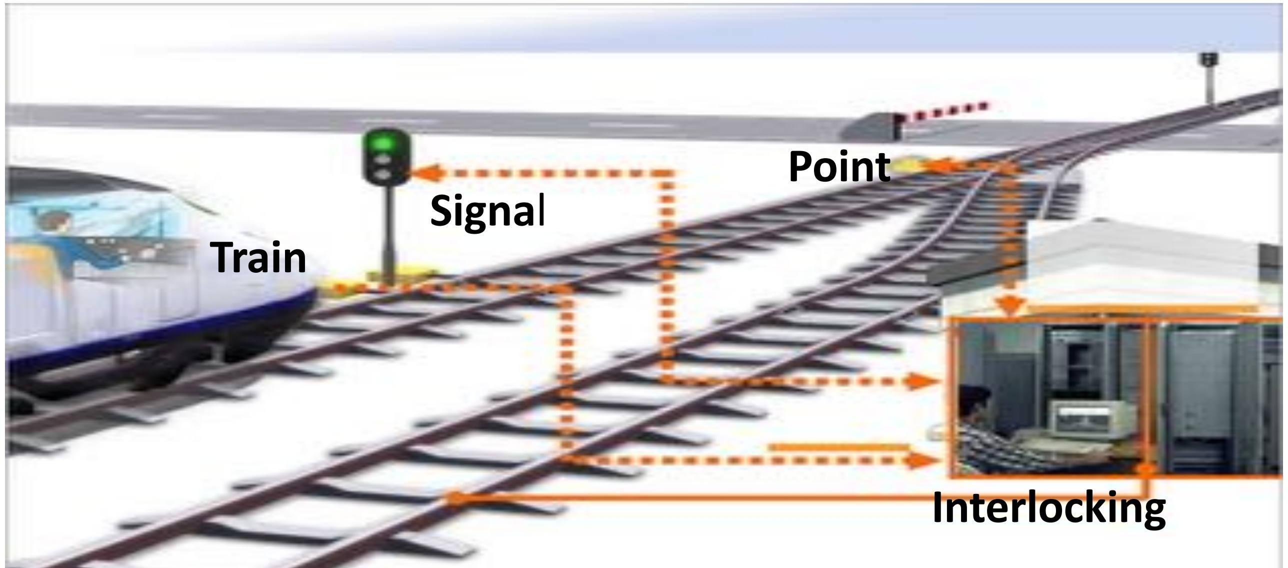


Fig1: Railway Signalling system

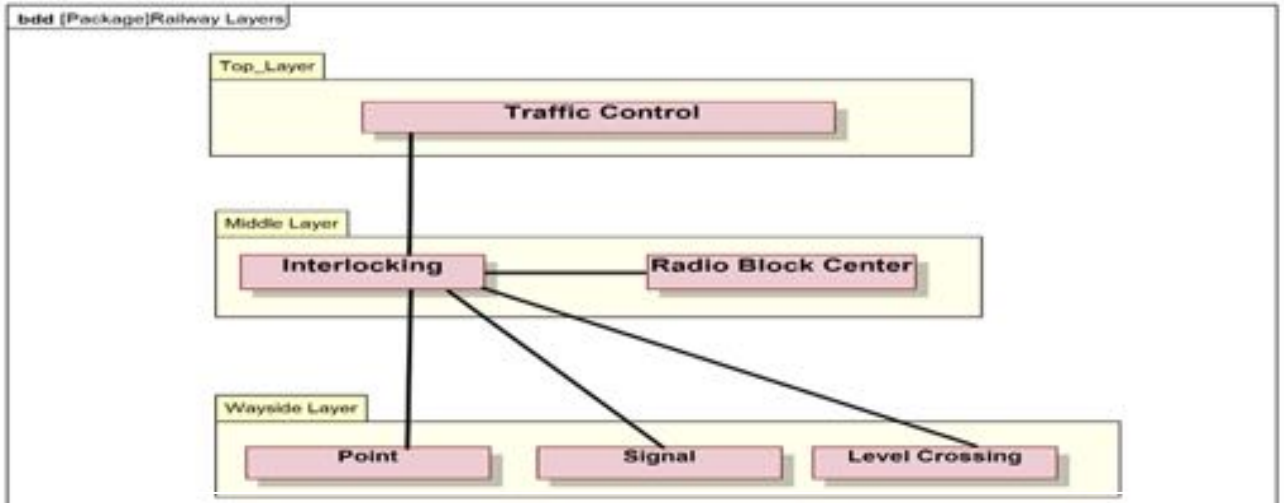
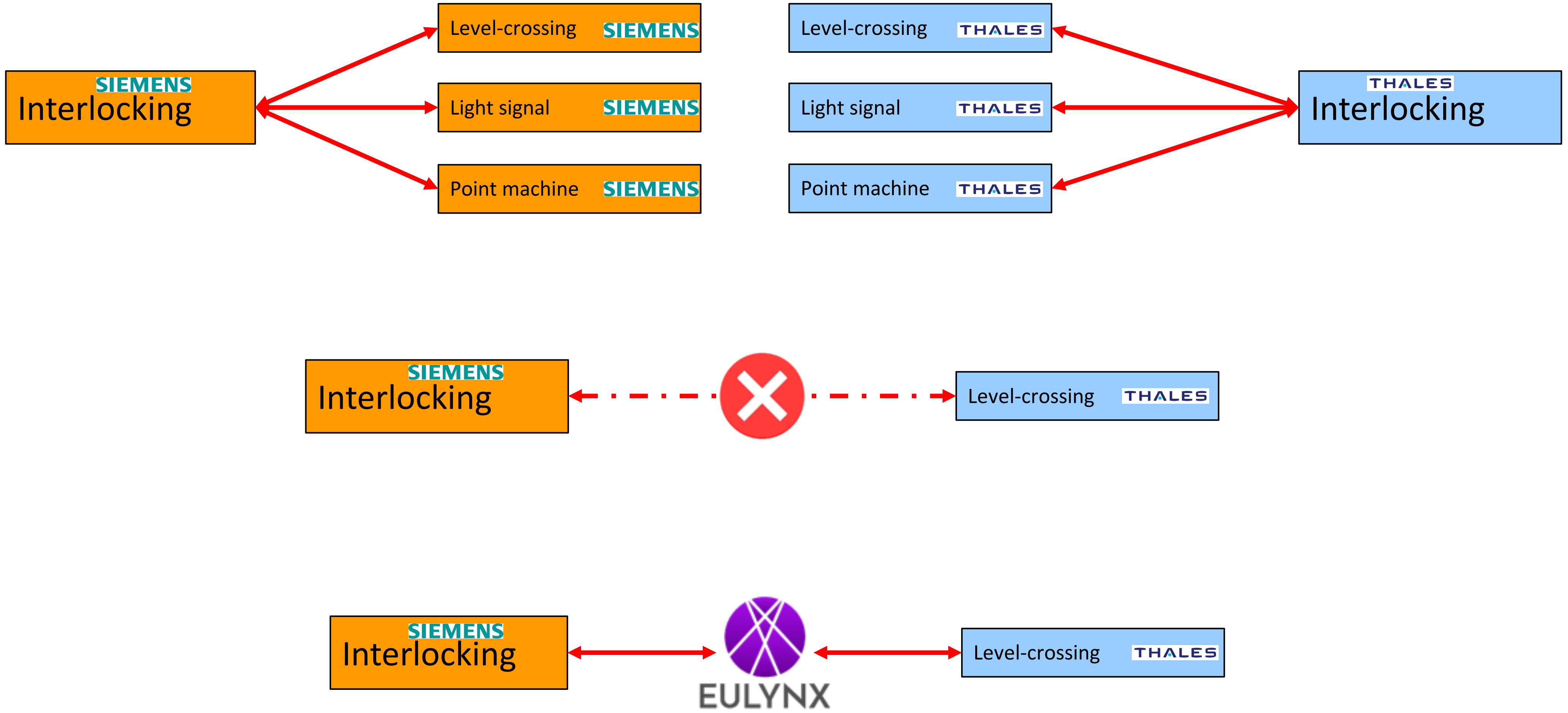


Fig2: Railway Signalling system: Layered Structure[Bui17]

Railway Signalling Interfaces - Scenario



EULYNX Initiative

- An initiative of 13 European railway infrastructure managers.
- Standardizing interface and elements of railway signalling systems to reduce the cost and installation time of signalling equipment.
- Uses MBSE approach to define requirement specification of each subsystem.
- Uses Systems Modelling Language, i.e. SysML for Modelling.
- The derived requirements from SysML model are unambiguous, correct, consistence and comprehensive.
- EULYNX architecture establishes long-term stable interfaces ensuring *"safety"*

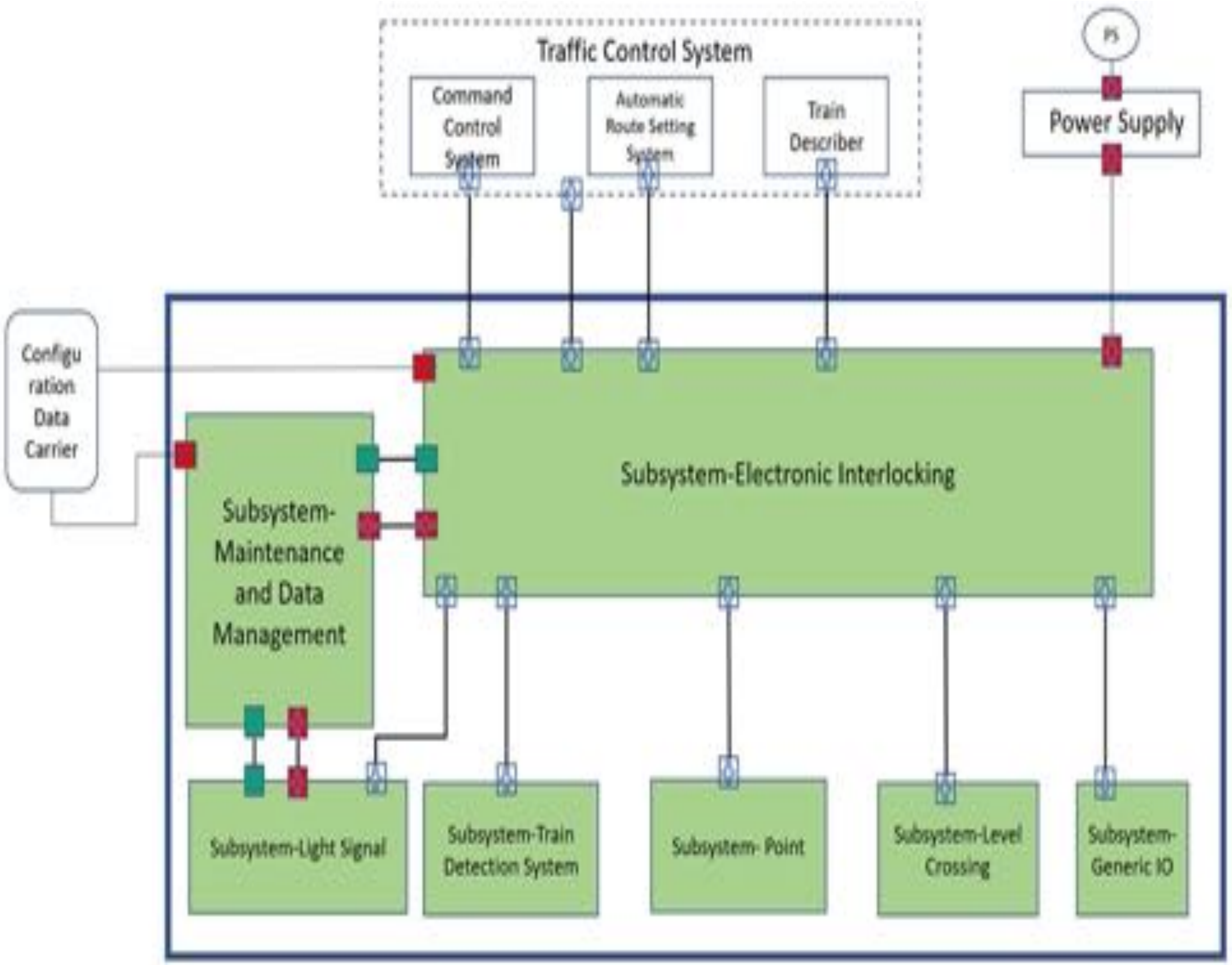
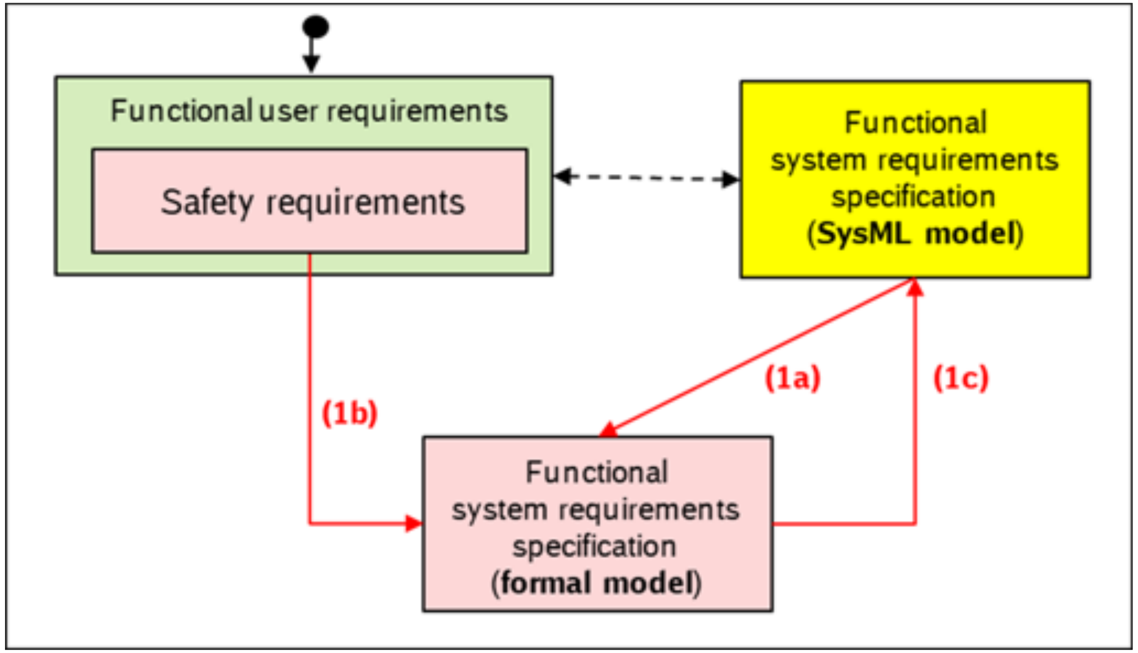


Fig3: EULYNX Architecture[Eul20].

Methodology



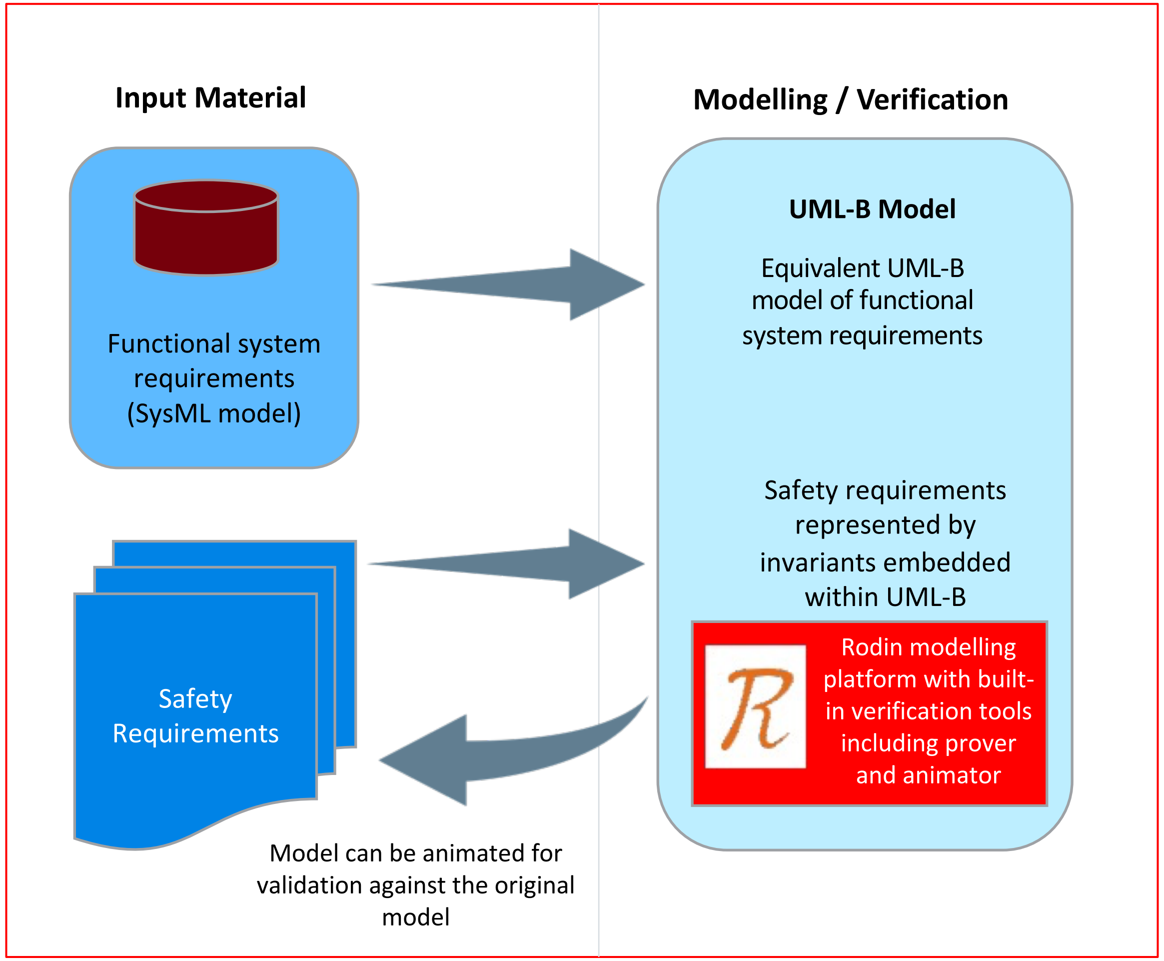
(1a) Transformation of the SysML model into a formal model based on defined transformation rules and **verification of the transformation**.

(1b) Transformation of safety requirements and **formal verification** of the formal model based on safety requirements (a subset of functional user requirements).

(1c) Correction of the SysML model as appropriate.

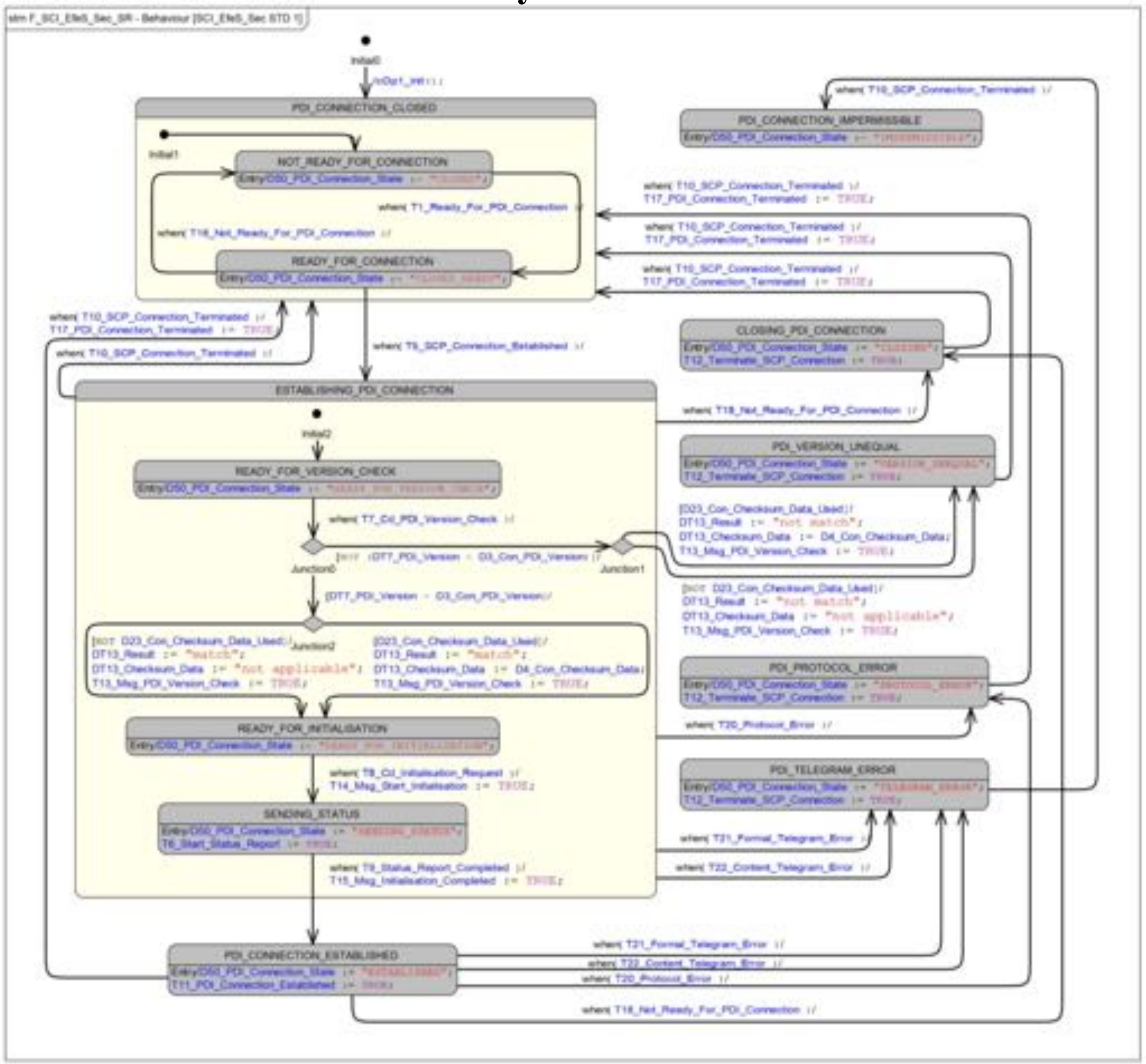
The process starts again with **(1a)** until no errors are found anymore.

Manual Translation to UML-B

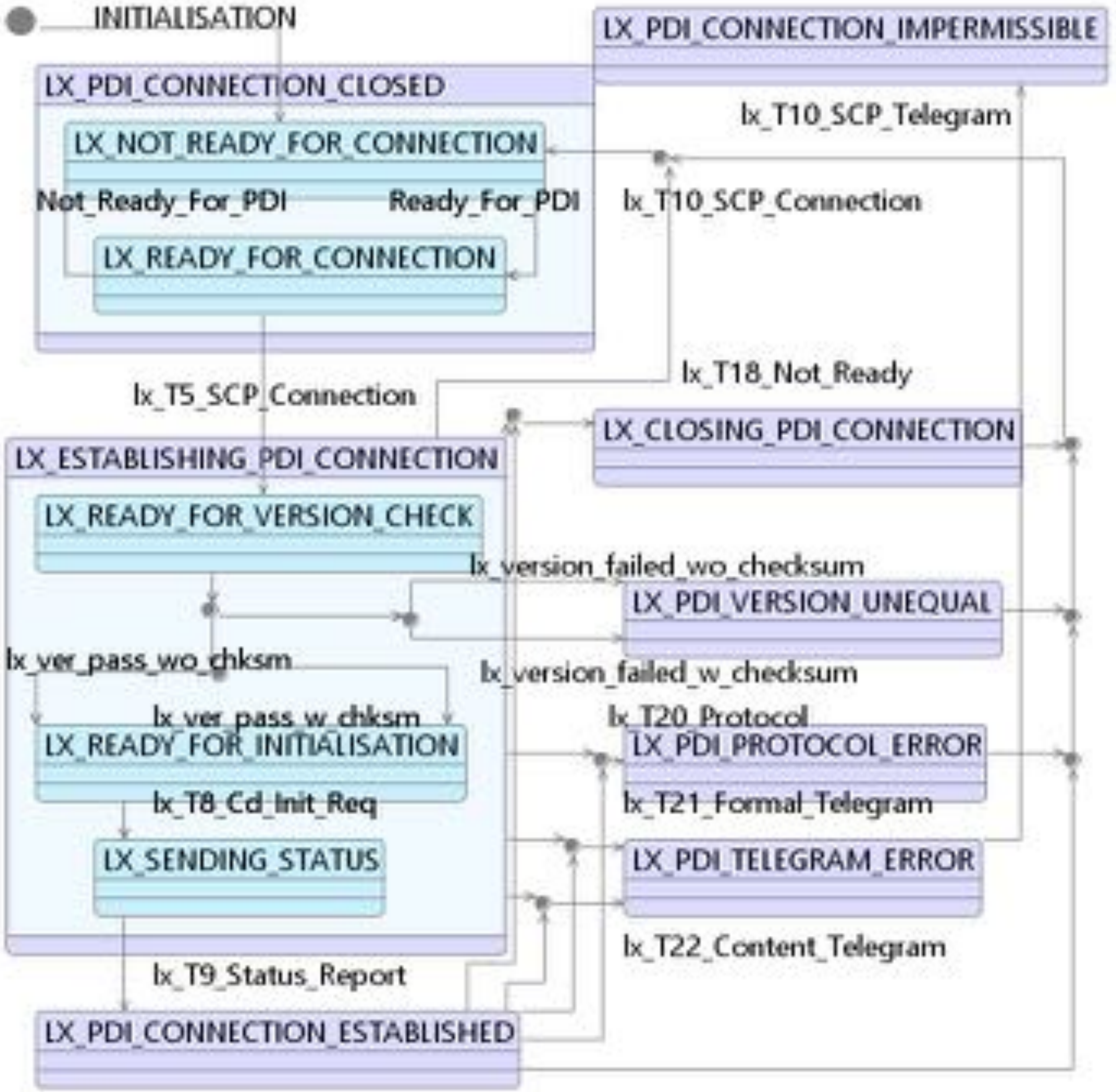


Model Comparision

SysML state machine



UML-B state machine



Results Manual Translation

- Identified two data type mismatch errors in the EULYNX SCI-P interface specification.
  - 'DT1_Move_Point_Target' in the EIL side is of STRING type and it is sent to 'T1_Cd_Move_Point' which is of type BOOL.
  - 'T1_Cd_Move_Point' in the EIL side is of BOOL type and it is sent to 'DT1_Move_Point_Target' which is of type STRING.

  The data type errors have been corrected by EULYNX in the latest release of SCI-P [9].

- Identified a non-existing state 'CLOSED' in the EULYNX SCI-P interface specification.

The variable 'D21_F_SCI_Efes_Gen_SR_State' is of STRING type which indicates the state in the EULYNX generic interface, and it was assigned to a non-existing state. This error was reported, and it has been corrected by EULYNX in the latest release of SCI-P [Eu.Doc.36]

- Deadlocks

Challenges

Manual Tranformation is time consuming.

Lack of expertise for Infrastructure Managers like Deutsche Bahn

Efficiency can be increased by automating the translation process.

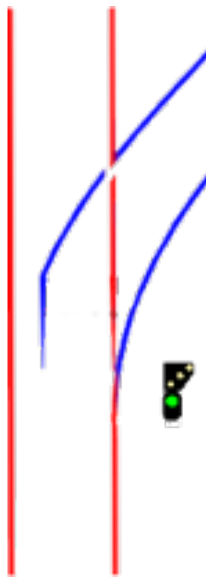
Automatic Transformation

- **Objectives:**

1. Propose a methodology and toolchain to automate transformation of SysML specification models into formal models (Event-B) and the generated Event-B Models can be verified.
2. To maintain traceability between informal requirements and the modelled system, specifically for the safety properties.
3. Reduce the efforts of the manual transformation of SysML semi-formal model to a formal model.
4. Support a modular railway signalling architecture with standardized interfaces.

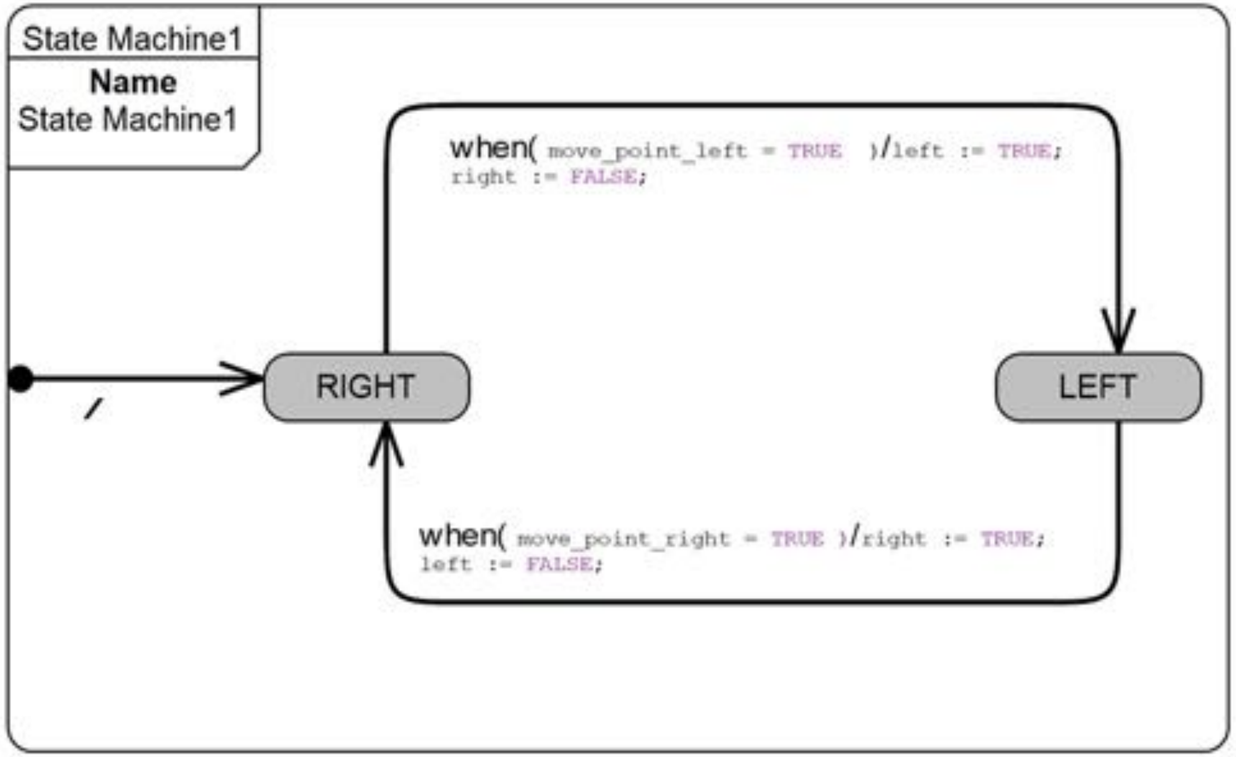
Example Use Case for Automatic Transformation

Case Study : Point Machine



Case Study : Point Machine

Use Case State Machine



Semantics Mapping

SysML Concept	Event-B Concept
State machine	Machine (Project)
State	Variable
Ports	Variable
Effects	Actions
Trigger	Guard
Transition	Event
State	Default Invariant

Methodology (Model-To-Model Transformation)

A. Model-To-Model Transformation Technique

– Model-to-Model Transformation - model mapping with the help of set “Rules” and “ Attribute Condition”.

– A Model-to-Model Transformation is the automated creation of m target models from n source models.

- Each model conforms to a given reference model i.e., a meta-model.
- The meta-models are commonly used as a “*schema*”.

Tool-Chain

– Model-to-model transformation uses Eclipse Modelling Framework as basic tool.
– The EMF project is a modelling framework and code generation facility for building tools.
– Transformations are executed by transformation engines that are plugged into the Eclipse Modeling infrastructure.
– This approach uses eMoflon:IBeX as a transformation engine.
– **eMoflon:IBeX** is a MDE tool for Triple Graph Grammars (TGGs) for bidirectional model transformation.

Methodology(Cont.)

- Define the Meta-model for both SysML state machine and Event-B language.
- Define the Set of “rules” required for mapping between SysML and Event-B language.
- Export State machine model from PTC integrity modeler in “.xmi” format.
- Apply “ Forward” transformation operation.
- Import generated model in RODIN Platform.

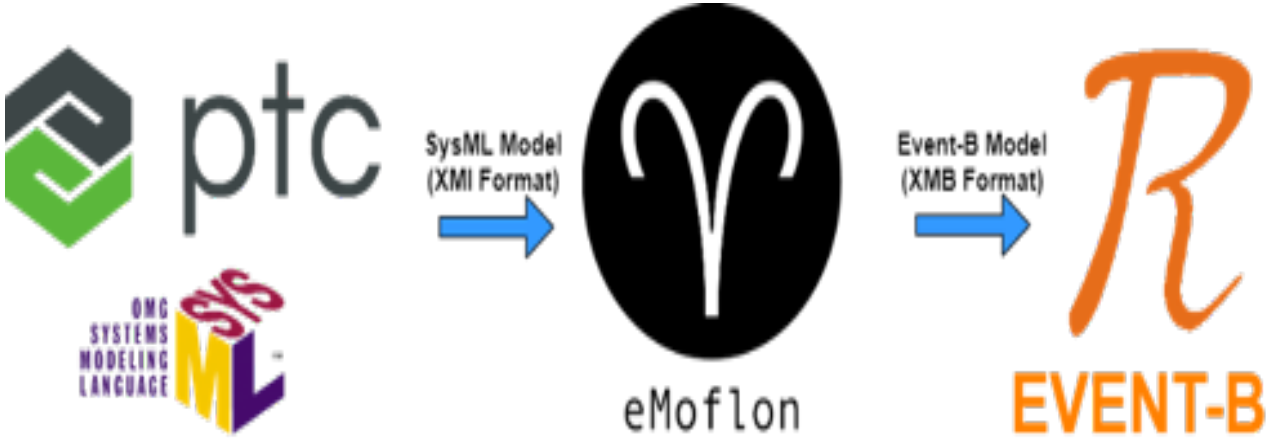


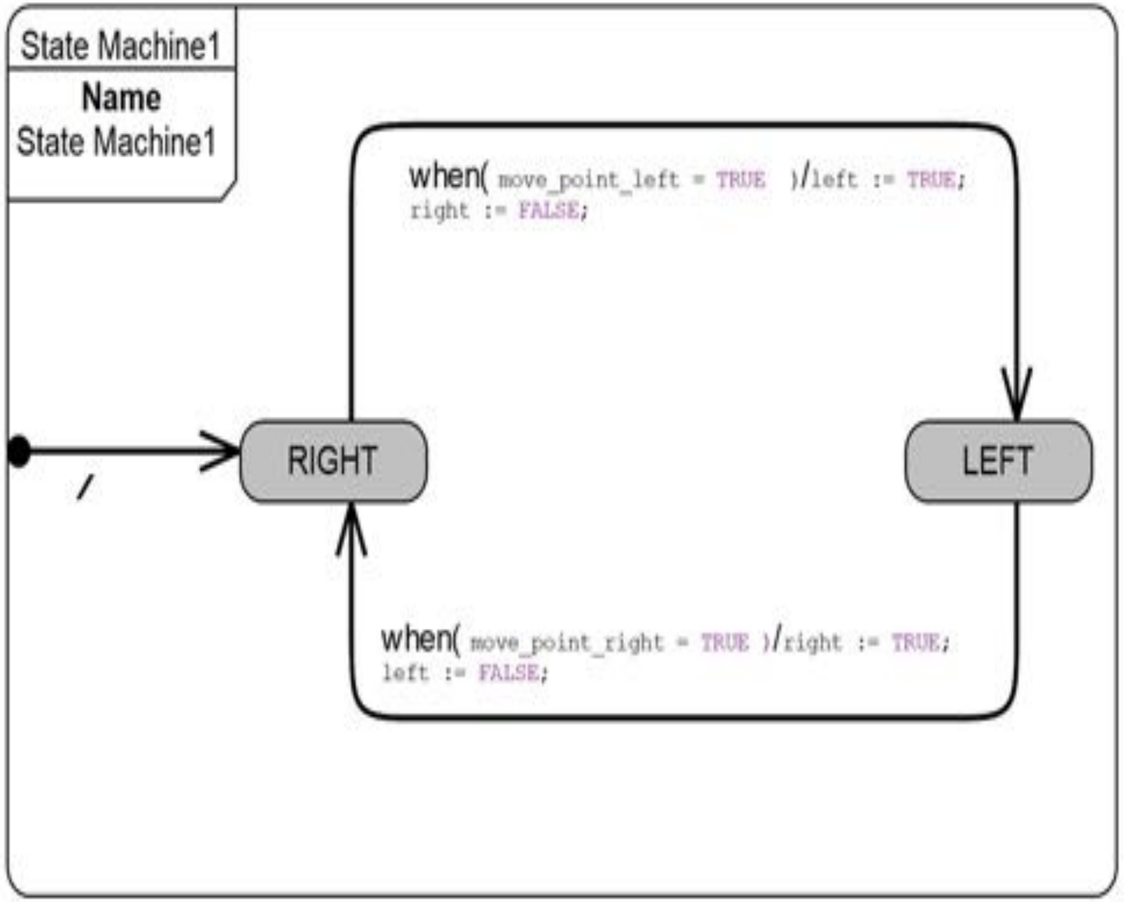
Fig5: Software Architecture For Model Transformation

Results

- Total 14 rules and 13 attribute conditions are defined for small use case.
- All the defined semantics mapping is accomplished and evaluated.
- More complex input models are provided as an input to evaluate the transformation.

Results

Use Case State Machine



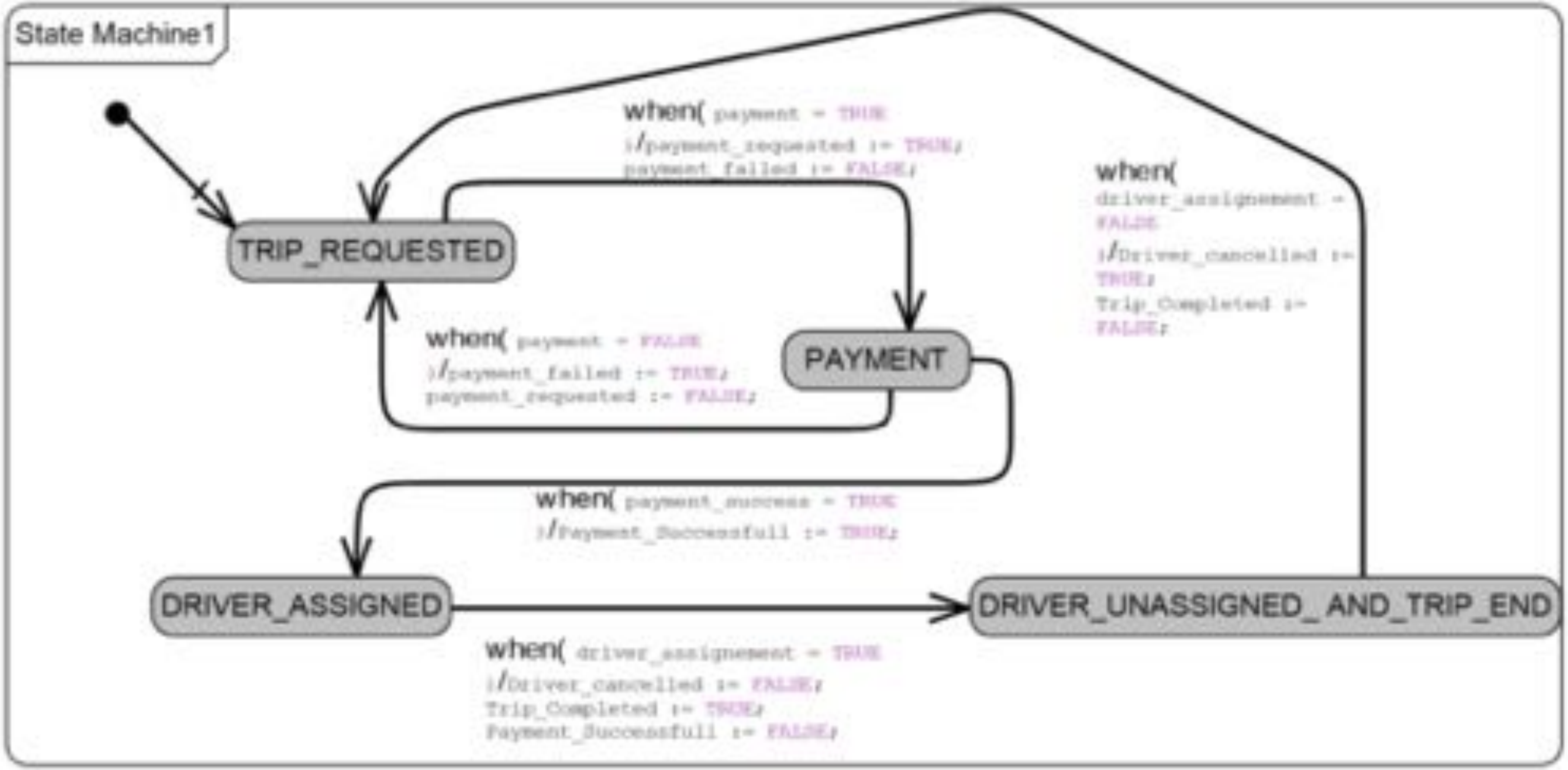
Generated Event-B code

```
MACHINE
    machine
VARIABLES
    right
    move_point_right
    move_point_left
    LEFT
    RIGHT
    left
INVARIANTS
    TYPEOF_LEFT    :  LEFT∈BOOL
    TYPEOF_RIGHT   :  RIGHT∈BOOL
EVENTS
    INITIALISATION  ≙
    STATUS
      ordinary
    BEGIN
      init_RIGHT   :  RIGHT≔TRUE
      init_LEFT    :  LEFT≔TRUE
    END

    Previous   ≙
    STATUS
      ordinary
    WHEN
      isin_LEFT    :  LEFT=TRUE
      guard2   :  move_point_right = TRUE
    THEN
      enter_RIGHT   :  RIGHT≔TRUE
      action4   :  left ≔ FALSE
      action3   :  right ≔ TRUE
      leave_LEFT   :  LEFT≔FALSE
    END

    Next   ≙
    STATUS
      ordinary
    WHEN
      guard1    :  move_point_left = TRUE
      isin_RIGHT   :  RIGHT=TRUE
    THEN
      action1    :  left ≔ TRUE
      enter_LEFT   :  LEFT≔TRUE
      action 2   :  right ≔ FALSE
      leave_RIGHT   :  RIGHT≔FALSE
    END

END
```

Evaluation : Test Case

DB NETZE neovendi

projects I engineering I digital transformation



SysML State Machine(Test Case1)

Evaluation : Test Case

```
MACHINE
  machine
VARIABLES
  payment
  payment_failed
  driver_assignement
  Payment_Successfull
  Trip_Completed
  payment_success
  payment_requested
  DRIVER_ASSIGNED
  PAYMENT
  DRIVER_UNASSIGNED_ AND_TRIP_END
  TRIP_REQUESTED
  Driver_cancelled
INVARIANTS
  TYPEOF_DRIVER_ASSIGNED  :  DRIVER_ASSIGNED∈BOOL
  TYPEOF_PAYMENT  :  PAYMENT∈BOOL
  TYPEOF_DRIVER_UNASSIGNED_ AND_TRIP_END  :  DRIVER_UNASSIGNED_ AND_TRIP_END∈BOOL
  TYPEOF_TRIP_REQUESTED  :  TRIP_REQUESTED∈BOOL
EVENTS
  To_DRIVER_UNASSIGNED_ AND_TRIP_END  ≙
  STATUS
   ordinary
  WHEN
   isin_DRIVER_ASSIGNED  :  DRIVER_ASSIGNED=TRUE
   guard4  :  driver_assignement = TRUE
  THEN
   leave_DRIVER_ASSIGNED  :  DRIVER_ASSIGNED≔FALSE
   enter_DRIVER_UNASSIGNED_ AND_TRIP_END  :  DRIVER_UNASSIGNED_ AND_TRIP_END≔TRUE
   action11  :  Payment_Successfull := FALSE
   action10  :  Trip_Completed := TRUE
   action9  :  Driver_cancelled := FALSE
  END

  To_Trip_Requested  ≙
  STATUS
   ordinary
  WHEN
   isin_PAYMENT  :  PAYMENT=TRUE
   guard2  :  payment = FALSE
  THEN
   action4  :  payment_failed := TRUE
   enter_TRIP_REQUESTED  :  TRIP_REQUESTED≔TRUE
   action5  :  payment_requested := FALSE
   leave_PAYMENT  :  PAYMENT≔FALSE
  END

  To_Driver_Assigned  ≙
  STATUS
   ordinary
  WHEN
   isin_PAYMENT  :  PAYMENT=TRUE
   guard3  :  payment_success = TRUE
  THEN
   leave_PAYMENT  :  PAYMENT≔FALSE
   enter_DRIVER_ASSIGNED  :  DRIVER_ASSIGNED≔TRUE
   action6  :  Payment_Successfull := TRUE;
  END

  To_Payment  ≙
  STATUS
   ordinary
  WHEN
   guard1  :  payment = TRUE
   isin_TRIP_REQUESTED  :  TRIP_REQUESTED=TRUE
  THEN
   action2  :  payment_failed := FALSE
   enter_PAYMENT  :  PAYMENT≔TRUE
   action1  :  payment_requested := TRUE
   leave_TRIP_REQUESTED  :  TRIP_REQUESTED≔FALSE
  END

  INITIALISATION  ≙
  STATUS
   ordinary
  BEGIN
   skip
  END

  To_End_Trip_Requested  ≙
  STATUS
   ordinary
  WHEN
   isin_DRIVER_UNASSIGNED_ AND_TRIP_END  :  DRIVER_UNASSIGNED_ AND_TRIP_END=TRUE
   guard5  :  driver_assignement = FALSE
  THEN
   enter_TRIP_REQUESTED  :  TRIP_REQUESTED≔TRUE
   action13  :  Trip_Completed := FALSE
   leave_DRIVER_UNASSIGNED_ AND_TRIP_END  :  DRIVER_UNASSIGNED_ AND_TRIP_END≔FALSE
   action12  :  Driver_cancelled := TRUE
  END

END
```

Generated Event-B Code

Conclusion

- Our work provides an approach and a prototype to design and implement the automatic transformation of SysML state machine to Event-B model.
- All the defined semantics mapping is accomplished and evaluated.
- The semantic similarities have been investigated between SysML state machine and Event-B to define a meta-model.
- The mapping between the SysML model and Event-B is elaborated with the help of TGG rules and attribute conditions.
- The simplicity of rule implementation and extension to transformation allows to add furthermore complex features to the SysML state machine.

Future Work

- The state machine meta-model can be extended to add more complex features for more complex state machines.
- Backward transformation can be implemented to maintain the traceability in case of detected errors.
- The creation of a suitable user interface for this transformation, that will make the transformation faster and more efficient.

References

- https://youtu.be/GhoNoMm4om0.
- EULYNX Requirements specification for subsystem Point Eu.Doc.36 v3.1. June 19, 2020.[Eu.Doc.36]

Vielen Dank


DB
NETZE