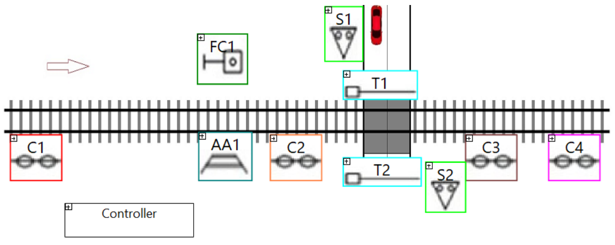
A model-based approach for the formal verification of specification

This work proposes a methodology for the automatic translation of a system model (with the level-crossing model example) into TLA+ for the formal verification of its safety properties. To achieve this, the proposed approach considers the interactions among the controller software, hardware components, sensors, actuators (including signaling equipment), and the dynamic behavior of the train and contextual systems. The primary safety objective in the given example is to ensure that no train passes through the level-crossing while road traffic remains possible.



Our approach consisted of specifying the SoI (System of Interest) model behavior and description of its environment. The entire model is defined using a specialized Domain-Specific Language (DSL) within the arKItect tool.

We introduced a structured language for textual requirement descriptions, similar to the Formal Requirements Elicitation Tool (FRET) developed by NASA Ames Research Center. This approach enables the definition of requirements in plain text with a specialized structure:

{If} text describing additional conditions {Formula},

{Then} text describing the resulting behavior {Formula}

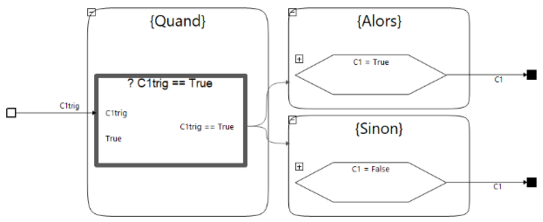
{Else} text describing the alternative behavior {Formula}

The syntax is short in comparison to FRET, and the set of available keywords is very small: preconditions are given after {When} and {If} keywords given in curly braces, all variables are treated as input signals, the action is described after {Then} and {Else} keywords and treated as output signal. The difference between {When} and {If} is only semantic: usually, we put simple requirement conditions after {If} and temporal conditions after {When}. *{Formula}* is a text in curly brackets providing the logic for the precedent keyword.

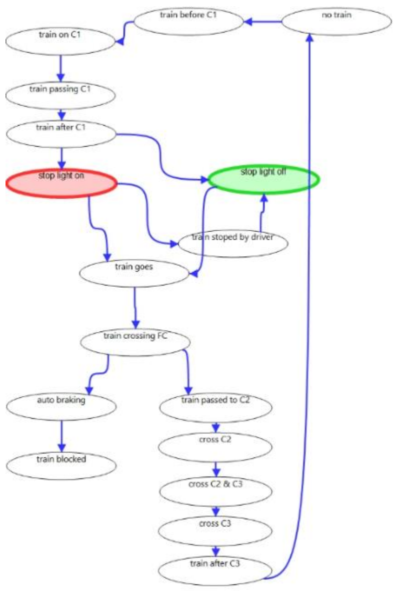
For example, Sensor C1 registers Train (note that the distance between C1 and FC1 can be so big that the C1 output signal can become False before the train comes to FC1). The C1trig variable represents the interaction between the train and the sensor. This can be described as:

{When} sensor is triggered {C1trig = True} {Then} output signal is True {C1 = True} {Else} output signal is False {C1 = False}

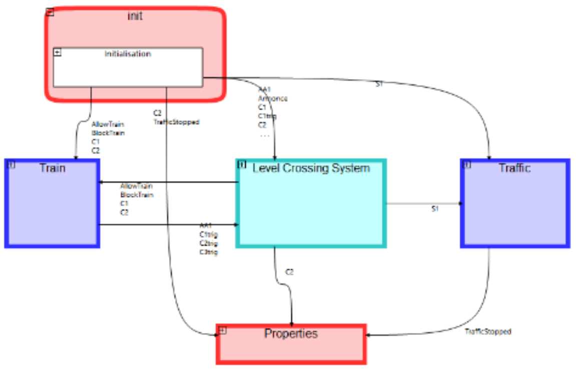
Each requirement is presented in textual form, and when translated into the model description, it is represented as a set of objects connected by appropriate data flows. All translated requirements are stored within the model's hierarchy, which describes the corresponding formulas. This approach facilitates the analysis of data flows and generates relational diagrams that depict the dependencies between requirements. Here is the example of C1 behavior description (automatically generated with the DSL help from the textual form) given as AST of the formula (for C1: C1 = True if C1trig = True else False) in the model:



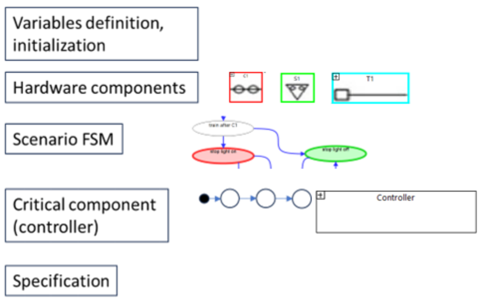
We utilize a scenario-based approach to reduce possible alternatives studied by Model Checking. In our case, this was done manually using a state machine to describe the behavior of a train passing through the level crossing, moving from one zone to another, triggering the appropriate sensors, and interacting with components in a specified sequence. After detailing trivial steps, an alternative scenario was described: if car traffic is blocked, Barrier T1 should have True on its output, but if it fails to close, the resulting signal is supposed to be equal to False. In such cases, the train is supposed to be stopped to prevent crossing the road while there is traffic. Therefore, if the semaphore for the train is Red, the train should stop. If the train does not stop, it is intended to be automatically halted by the system AA1.



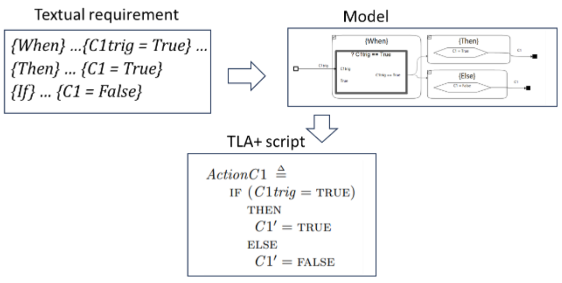
Having modeled all system parts, we can get a schema representing all data flows between the SoI, Train behavior FSM, traffic FSM, and initial variables.



When the model is consistent, we can generate valid TLA+ code automatically, helping in model validation with the Model Checking. Here is the basis structure of the TLA+ code we generate.


Variables definition, initialization
Hardware components
Scenario FSM
Critical component (controller)
Specification
Controller

Hardware components are given with simple logical blocks, where the 'future' values are directly given to all impacted variables. When the requirement does not define the alternative value of a variable, it is automatically described as 'unchanged'.



```
ActionC1 ≜
    IF (C1trig = TRUE)
      THEN
        C1' = TRUE
      ELSE
        C1' = FALSE
```

The specification is given as a set of actions being performed at each tick ('instantly' in terms of behavior), except for ActionTrain and Controller components (representing Train and Controller behaviors), which are marked as 'critical' in the model and should be executed one after another.

```
Next ≜ (ActionTraffic ∧ ActionS1 ∧ ActionAA1 ∧ ActionC3 ∧ ActionC2 ∧ ActionC1
            ∧ ActionT1 ∧ ActionFC1 ∧
            (ActionTrain ∨ ActionController)
        )
Spec ≜  ∧ Init ∧ □[Next]_vars
```

Behavior FSM is exported as classic FSM interpretation in TLA+, triggering the following states with transitions given in the model and producing signals for hardware components. In addition, behavior FSM is protected with the ControllerReady (CR) variable, staying True only when the controller finishes all its actions (assuming the controller responds instantly compared to the environment). A list of all unchanged variables (related to controller behavior) is generated.

```
ActionTrain ≜
  ∧ CR
  ∧ UNCHANGED ⟨CR, FC1, waitC3, stateController, S1act, leaving, CloseBarrier, S2act, Annonce⟩
  ∧
    ∨ ∧ (stateTrain = "no train")
      ∧  (TRUE)  train before C1
      ∧  (stateTrain' = "train before C1")
      ∧  UNCHANGED ⟨AA1, C1trig, C2trig, C3trig⟩
    ∨ ∧ (stateTrain = "train before C1")
      ∧  (TRUE)  train come to C1
      ∧  (stateTrain' = "train on C1")
      ∧  (C1trig' = TRUE)
      ∧  UNCHANGED ⟨AA1, C2trig, C3trig⟩
    ∨ ∧ (stateTrain = "train on C1")
      ∧  (TRUE)  train passing C1
      ∧  (stateTrain' = "train passing C1")
      ∧  UNCHANGED ⟨AA1, C1trig, C2trig, C3trig⟩
    ∨ ∧ (stateTrain = "train after C1")
      ∧  (FC1 = TRUE)  train go form C1
      ∧  (stateTrain' = "stop light on")
      ∧  UNCHANGED ⟨AA1, C1trig, C2trig, C3trig⟩
    ∨ ∧ (stateTrain = "train after C1")
```

Controller (critical component) generation is also represented as FSM, yet it is described with a set of requirements in the model. Our goal was to ensure that all actions described in the requirements were completed before activating the next stage in the Train behavioral finite state machine.

```
ActionController ≜
    UNCHANGED ⟨waitC3, stateTrain, C1trig, AA1, C3trig, S2act, C2trig⟩
    ∧
    (
    ∨  ∧  stateController = "initController"
       ∧  stateController' = "1._c1_--_Announcement"
       ∧  UNCHANGED ⟨CR, FC1, leaving, Announcement, CloseBarrier, S1act⟩
    ∨  ∧  stateController = "1._c1_--_Announcement"
       ∧  stateController' = "2._Announcement_--_close,_S1"
       ∧  IF (C1 = TRUE)  1. c1 – Announcement
           THEN
             Announcement' = TRUE
           ELSE
             UNCHANGED ⟨Announcement⟩
       ∧ UNCHANGED ⟨CR, FC1, S1act, CloseBarrier, leaving⟩
    ∨  ∧  stateController = "2._Announcement_--_close,_S1"
       ∧  stateController'     = "3._T1_--_FC1"
       ∧  IF (Announcement = TRUE)  2. Announcement – close, S1
           THEN
             S1act' = TRUE ∧ CloseBarrier' = TRUE
           ELSE
             UNCHANGED ⟨S1act, CloseBarrier⟩
       ∧ UNCHANGED ⟨CR, FC1, Announcement, leaving⟩
```

The automatically generated TLA+ code can be executed without modifications with the TLC model checker. The implemented Model-in-the-loop approach allows the testing of systems with different behaviors. Having all data stored in the Model, this solution implements a no-code approach for systems design and validation. This, collectively, clarifies the model definition, accelerates the coding process, and significantly lowers the entry barrier for engineers using TLA+.