European Train System Modeling requirements: verification and validation

Telecom SudParis research team
Lead by Prof. Ana R. Cavalli

14 January 2014

OUTLINE

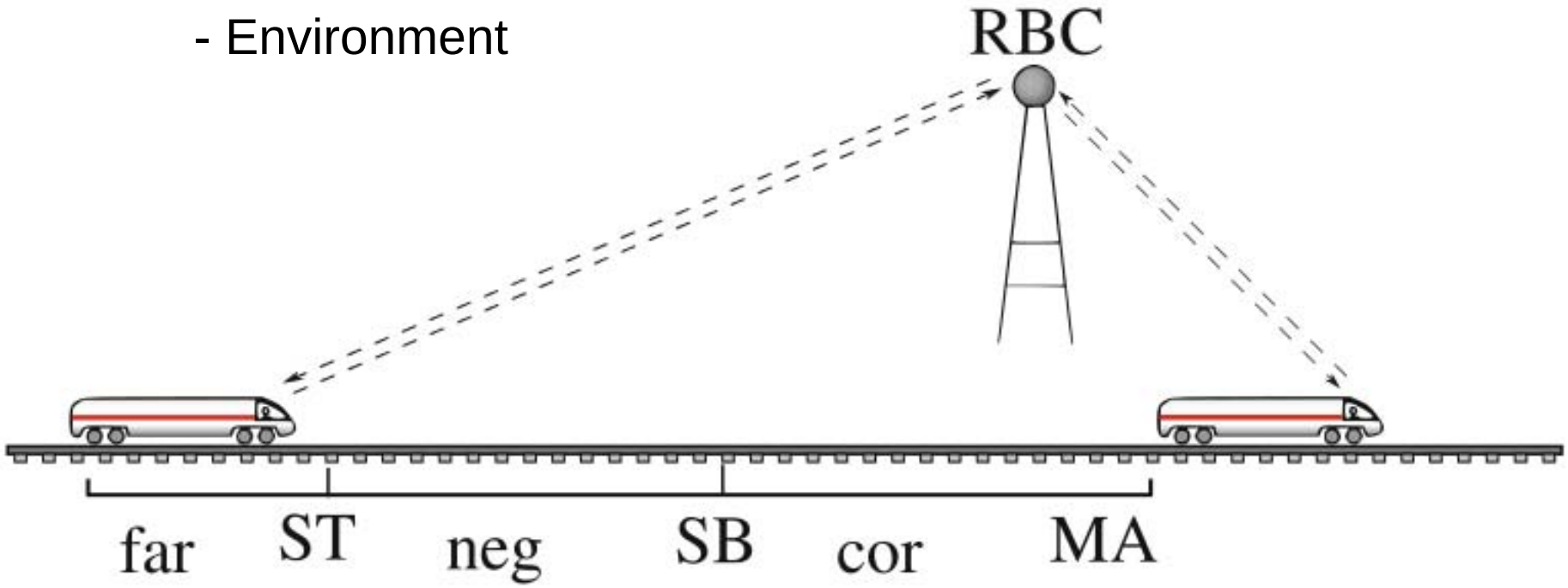
- What do we model?
- What are system requirements?
- TEFSM model
- Verification and validation of the model
- TEFSM based test derivation
- Future work

What do we model?

Actors of modeling

- Radio Block Center (RBC)
- Train
- Environment

Security properties must be verified!!!



Train system requirements

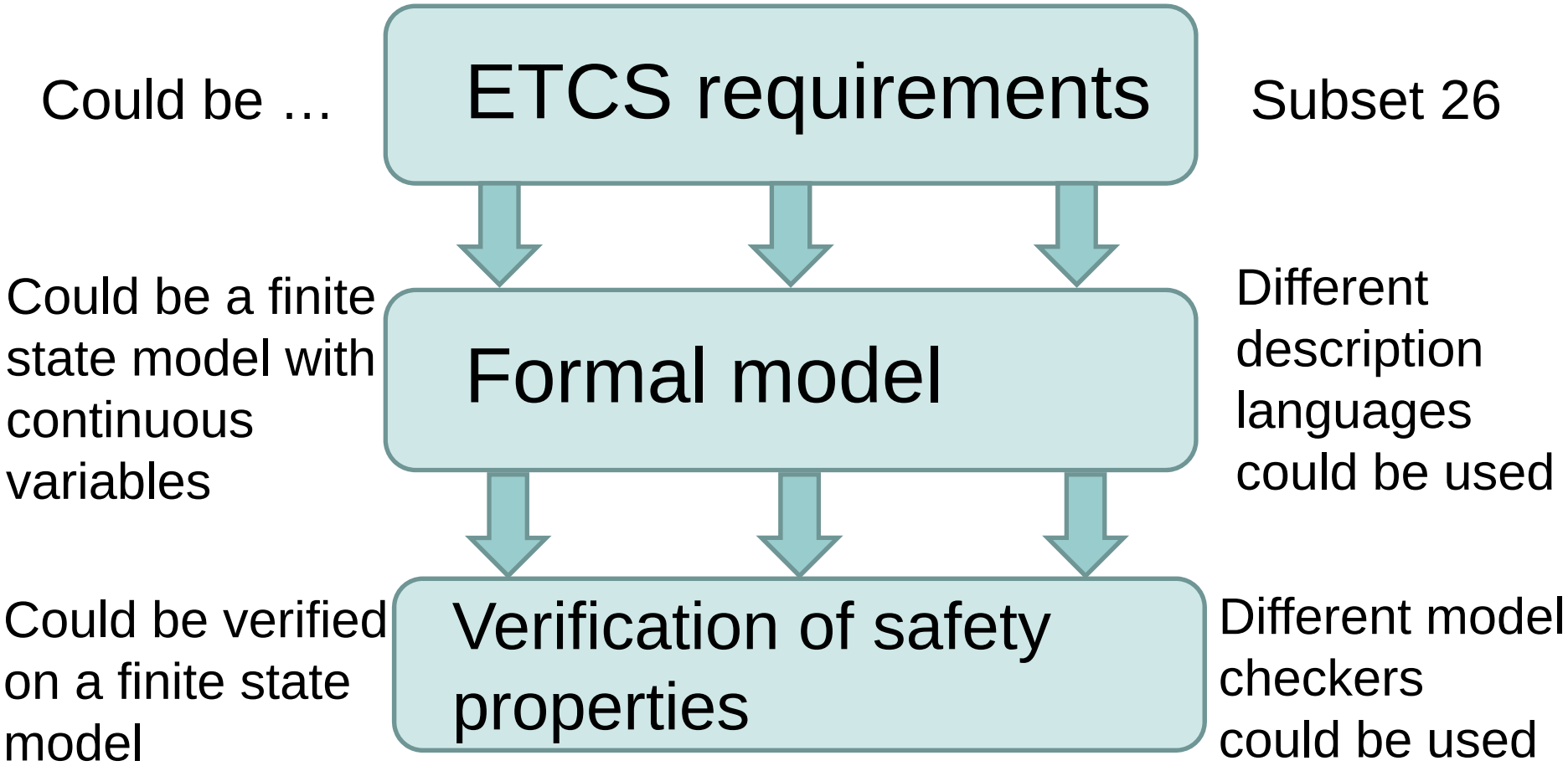
- *R1. For each train, there is the safety distance d*

- *R2. If TrainId is controlled by the RBC, then the train reports its current position (p), speed (v), and acceleration (a) and the next internal state where the output parameters p, v, a are updated according to the train sensors*

- *R3. The input parameter SD represents the safety distance between two trains and is a constant in the model*

- *R4. Messages between the train and the RBC may be lost. However, the train continues moving and it should automatically decide if it is in a safe position or not*

Modeling European Train Control System



Modeling decisions

- Absence of a global time $\Rightarrow$ each player has its own clock
  - How often the train should report to RBC?
  - How often the RBC has to send control messages to the train?

$\Downarrow$

We assume discrete time instances when messages may be received/sent

If there is no input before some timeout expires then the train should be stopped

- Autonomous way of the train to move
  - When the train should negotiate with RBC about the safety distance?
  - When the train should be stopped?

Modeling decisions (2)

- Absence of a global time

  Can be solved by integrating (input and output) timeouts into the model

- Autonomous way of the train to move

  Can be solved by modeling different states

  ⇓

  A finite state model with integrated time aspects might be helpful

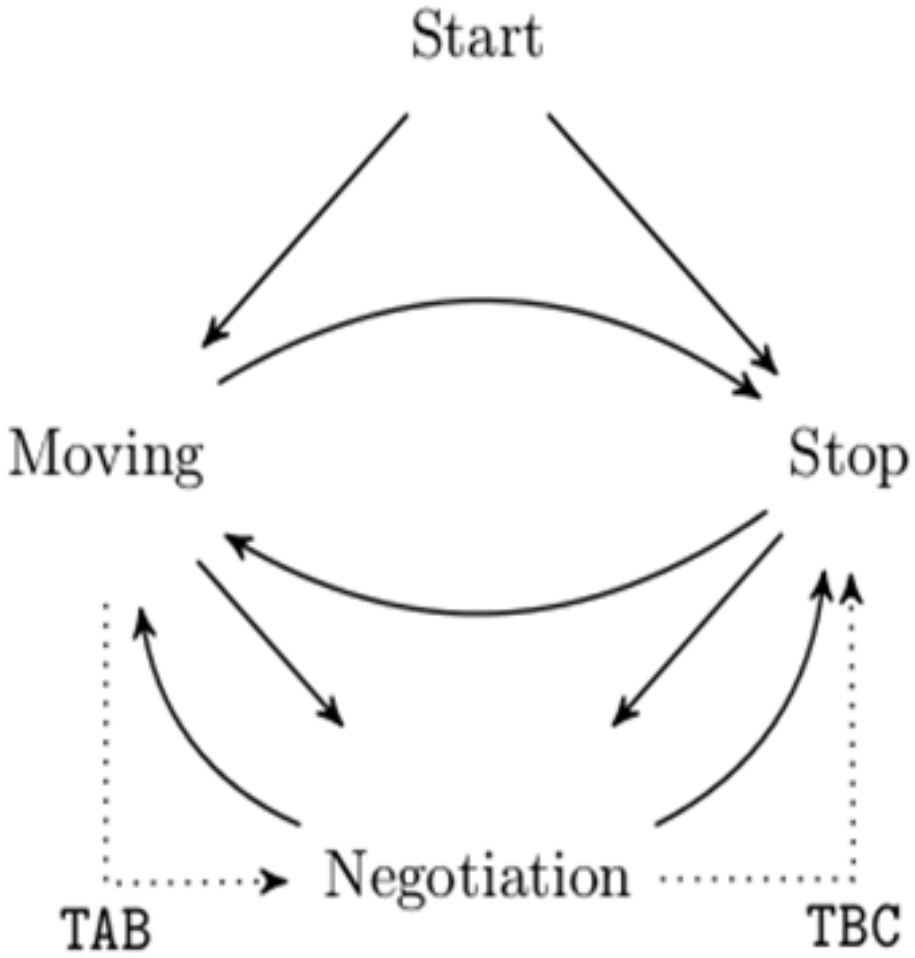
  ⇓

  A model based on a Timed Extended Finite State Machine has been developed during the first year of the project

  ⇓

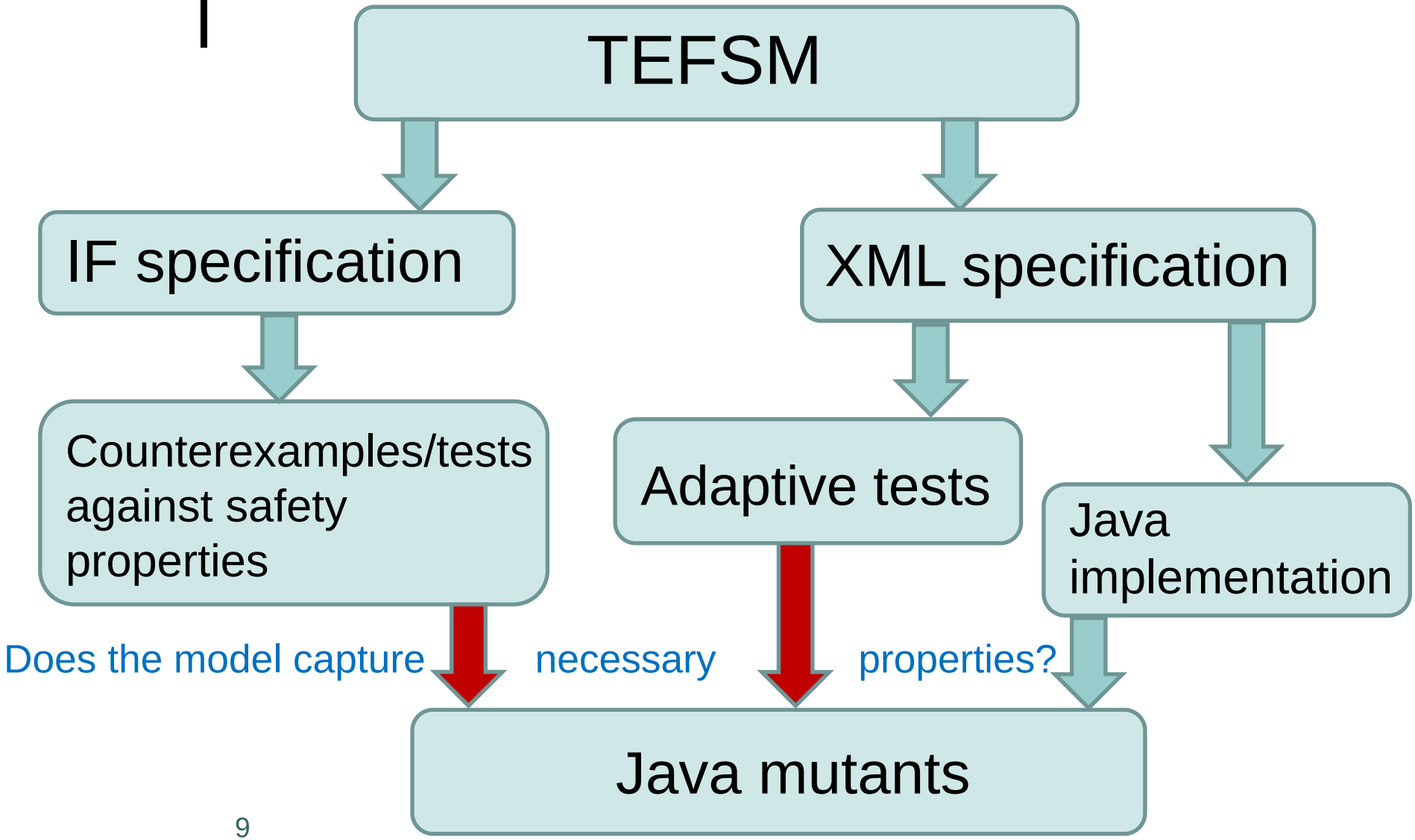
  Last year has been devoted to the verification and validation of the model

TEFSM model



- Transitions are labeled by input/output pairs, predicates, updating functions, and timeouts
- Total number of transitions is around 50
- From state START under the message *control*(*j*) the train can move to different states depending on its position and the train reports its position, speed and state to RBC

Verifying and validating the model



IF language

- IF allows to describe temporized machines
- A real-time system is a composition of processes running in parallel and interacting asynchronously
- Processes interact through shared variables and message exchanges via communication channels
- Definition of the system includes
  - data types
  - constants
  - shared variables
  - communication signals
  - processes

IF specification

```
system ETCS;

/* Constant definitions */
const SD = 1;
const NbTrains = 2;

/* Type definitions */
type states = enum
  start, moving, _stop, negotiation
endenum;
...

/* Signals definitions */
signal ETCS_no_controlled(pid);
signal ETCS_report(pid, integer, integer, integer, states);
signal ETCS_control();
signal ETCS_move(dType, VmaxType, AmaxType);
...

/*Signal route definitions */
signalroute train_to_RBCenv(1)
    from Train to env
    with ETCS_no_controlled, ETCS_report;
....
```

```
/* Main process */
 process Train(2);

 /* Local variables */
    var Vmax integer private;
    var c_TAB clock private;
...

/* States specification */
state start #start ;
    deadline lazy;
    input ETCS_move(d, Vmax, Amax);
        output ETCS_no_controlled(self);
        nextstate -;
    ...
endstate;
...
endprocess;
endsystem;
```

We model the train behavior in the IF language

An example of a safety property

The train is in the *Moving state*, running at 100km per hour, with an acceleration up to 10km per h^2 on a distance of 100km

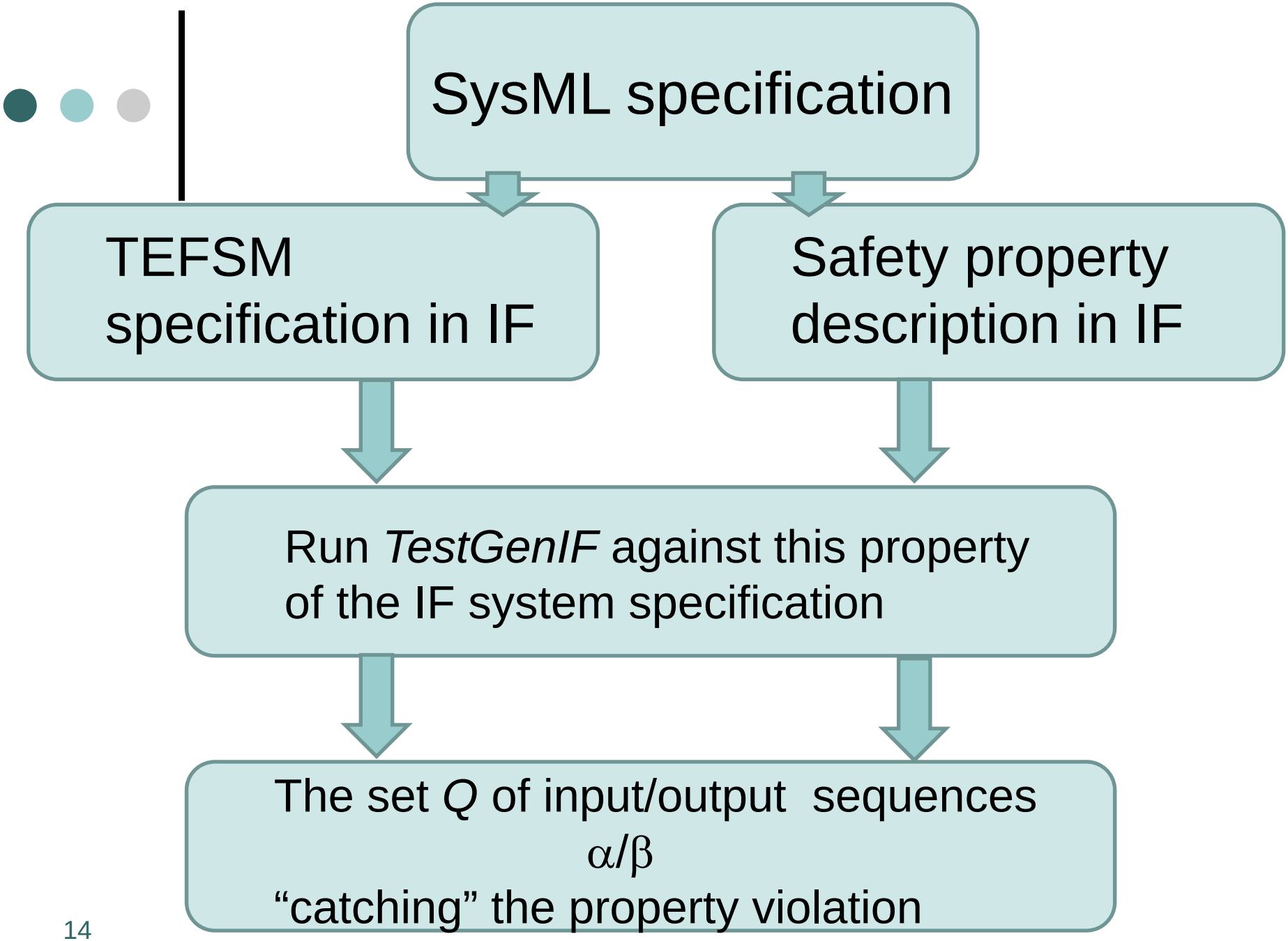
Due to some external unexpected reasons, the train must be stopped as soon as possible

An alarm from the environment is sent to the train

Automatically, the train should be stopped and has to provide a reporting message with its current parameters (state, position, etc.) to the RBC

Modeling the property in IF

```
OBJ(1) = OBJ(ord) = {obj1, obj2}
obj1 = cond1 ∧ cond2 ∧ cond3 ∧ cond4
    obj1 = cond1 ∧ cond2 ∧ cond3 ∧ cond4
    cond1 = process: instance = {train}0
    cond2 = state :source: moving
    cond3 = state: destination:stop
    cond4 = action: input alarm_to_train()

obj2= cond1 ∧ cond2 ∧ cond3 ∧ cond4
    cond1 = process: instance = {train}0;
    cond2 = state: source: moving
    cond3 = state: destination:stop
    cond4 = action: output report(train_id,p,v,a,s)
```


SysML specification
TEFSM specification in IF
Safety property description in IF
Run *TestGenIF* against this property of the IF system specification
The set *Q* of input/output sequences
α/β
"catching" the property violation

Verifying the set *Q*

? Does the IF model correspond safety requirements?

? Is the set Q of input/output sequences derived based on IF specification of TEFSM good enough?

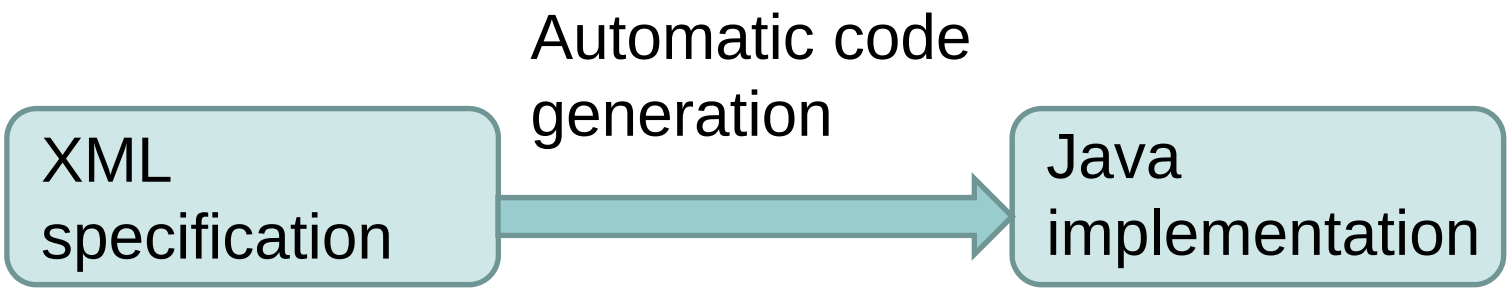
- *Good = allows to detect safety property violations*

⇓

- We implement the Train System Simulator

⇓

- We specify the TEFSM is XML



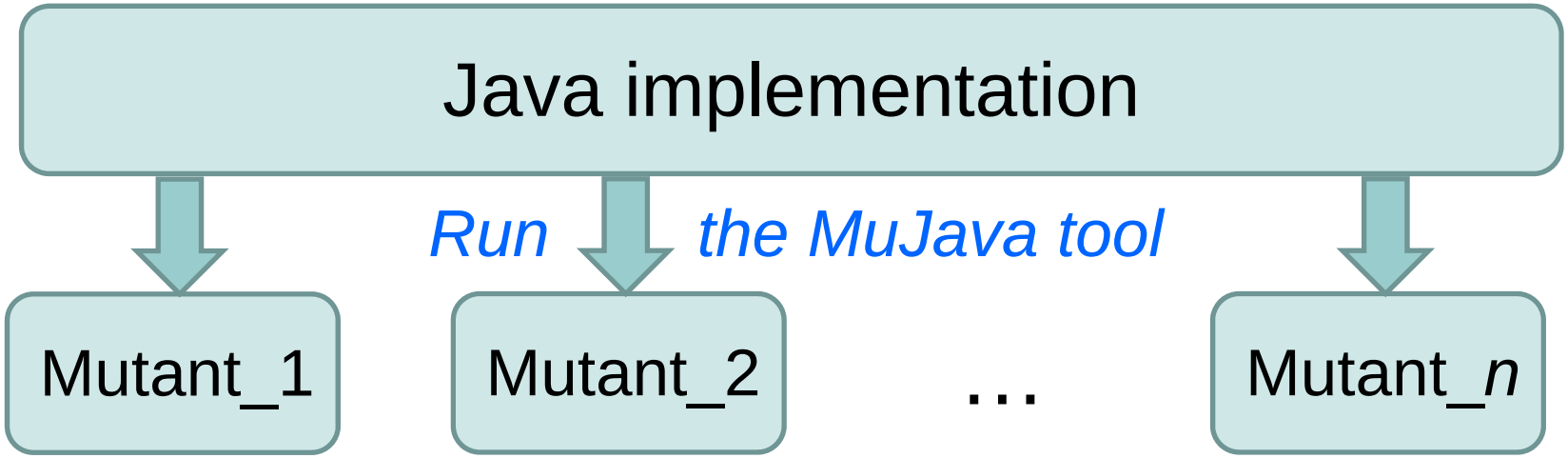
XML file

Contains the TEFSM specification

```
<model>
<state id="s0">
<transition dest="s1" input="a" output="z" time="1">
</state>

<state id="s1">
<transition dest="s2" input="a" output="y" time="5">
<timeout dest="s3" time="5">
</state>

<state id="s2">
<var id="v1" value="w">
<var id="v2" value="v">
<transition dest="s0" input="b" output="m" time="3">
<condition>
v1>v2
</condition>
</transition>
<transition dest="s2" input="a" output="z" time="3">
<input id="x">
<input id="y">
<input id="z">
<output id="w">
<output id="v">
<update>w=x+y</update>
<update>v=2*z</update>
</transition>
</state>

<state id="s3">
<transition dest="s2" input="c" output="n" time="3">
</state>
</model>
```

Mutants for safety properties

- Java implementation has been automatically derived from the XML file
- A number of Java mutants have been developed taking into account Train safety properties



- The quality the set *Q* has been checked on the set of generated mutants

Experimental results

- XML and IF specifications have been compared
- Specifications conform each other w.r.t. the set *Q* of generated input/output sequences
- Hundred of mutants (in total) have been generated for the Java implementation
- More than 90% of mutants have been killed by the sequences of the set *Q*

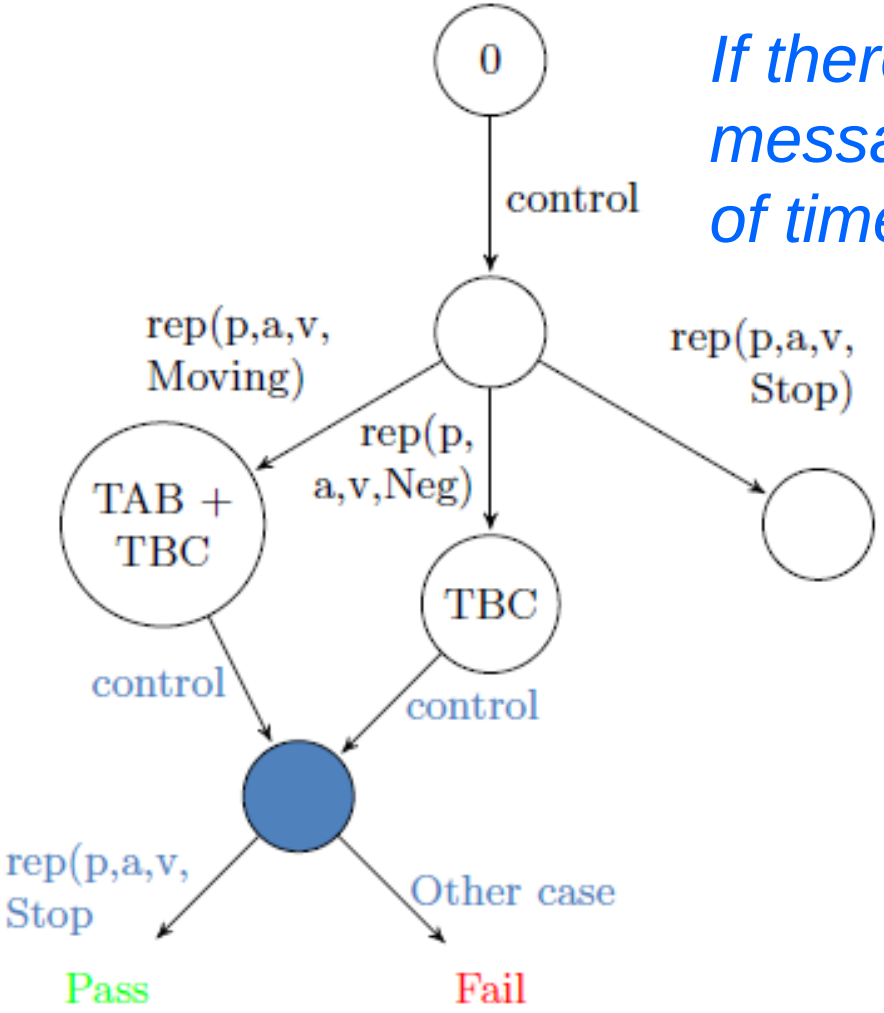
$\Downarrow$

The TEFSM model is efficient for deriving test sequences when taking into account Train safety properties

Adaptive testing

- Differently from preset tests, in adaptive testing the next input is selected based on previously observed outputs
- We have proposed adaptive testing scenario based on the TEFSM description of the Train system
- Adaptive test is represented as a test case
- A test case as an acyclic TEFSM where only one timed input is defined at each intermediate state
- The quality of the adaptive test has been evaluated in the same way, based on mutants of the JAVA implementation

Example of an adaptive scenario



If there is no signal from RBC (lost messages) for an appropriate period of time then a train must stop itself

- Similar safety properties are formulated in adaptive way
- We have experimented with adaptive tests against the same Java mutants and all the mutants have been killed

Main results

- A Timed Extended Finite State Machine has been used for describing the Train System
- Based on this model a set of adaptive tests has been generated taking into account safety properties
- For evaluating the quality and abilities of the developed model, the TEFSM model has been specified in different languages
- TEFSM based tests for safety properties have been derived based on the IF specification
- The Train System JAVA simulator has been elaborated
- The quality of the TEFSM model has been evaluated based on the IF specification and JAVA simulator

⇓

Experimental results clearly show the effectiveness of describing the Train System in terms of TEFSM

Future work

- The model should be extended in the following directions

  a) A time for transition execution should be integrated into the model

  b) More complex continuous updating functions should be considered

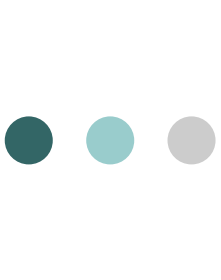
  c) Some of TEFSM states should be unrolled, i.e. should be considered as composite

  d) The OBU itself should be modeled

  e) More formulas should be included into the model (for example, when describing the train profile)
- More experiments will be performed considering taking into account safety properties
- Different model checking techniques and tools will be considered

Discussion on possible collaboration

- Could different Train System models be compared w.r.t. corresponding sets of input/output sequences?
- Could the TEFSM based tests be executed against implementations of Partners?
- Could a language for describing the Train System model be unified? Like SysML, for instance?
- …



Thank you!