Supporting Railway Infrastructure Managers with Formal Models and Analyses

Bas Luttik

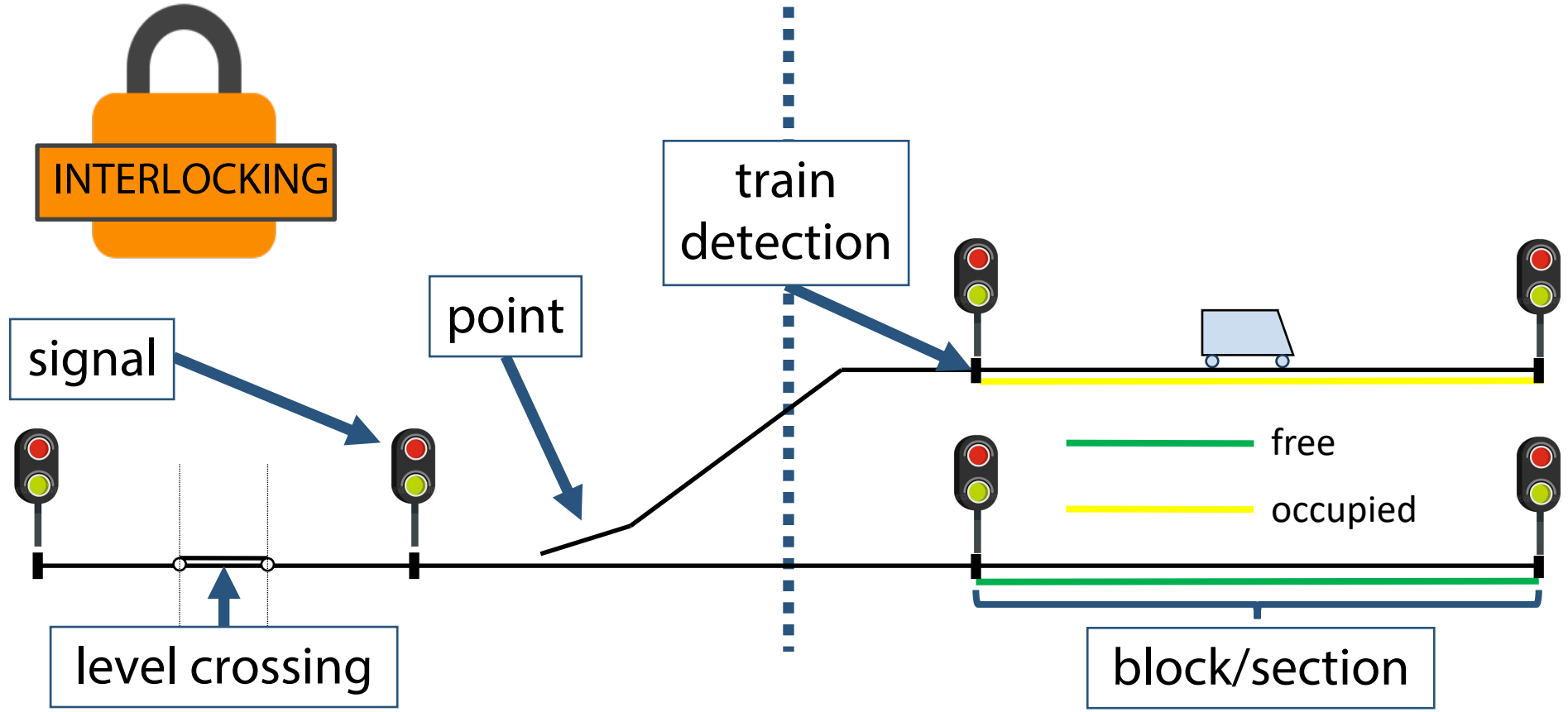
Based on joint work with many people (to be mentioned later!)

AlgoUK/BCTCS, April 6, 2020

European collaboration

Standardisation

INTRODUCTION


INTERLOCKING
train
detection
point
signal
free
occupied
level crossing
block/section

1. An application of FM in the development of EULYNX

   a) EULYNX, FormaSig (general ideas)
   b) First case study: Point
   c) Next steps

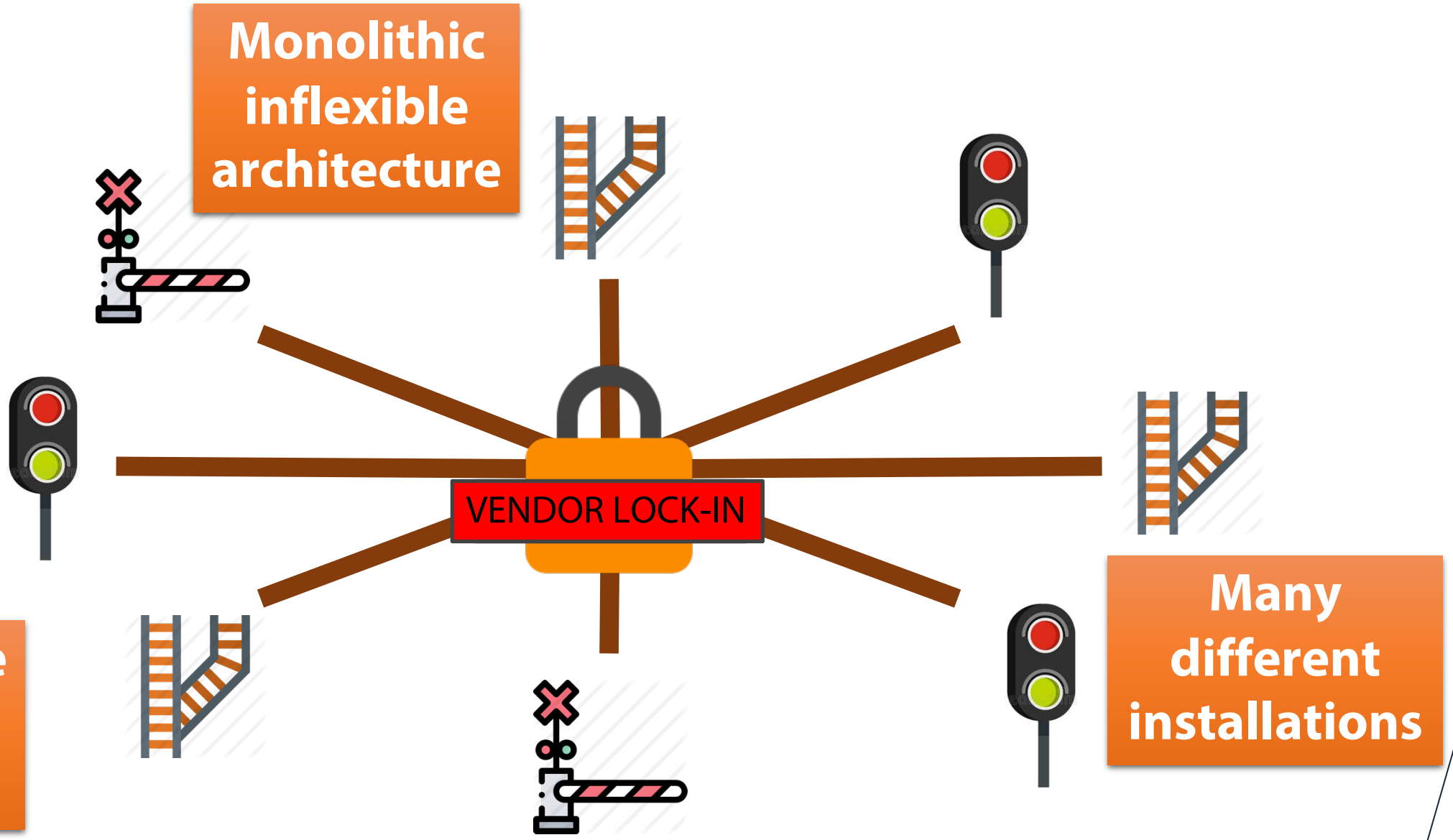
ProRail Collaboration with: DB

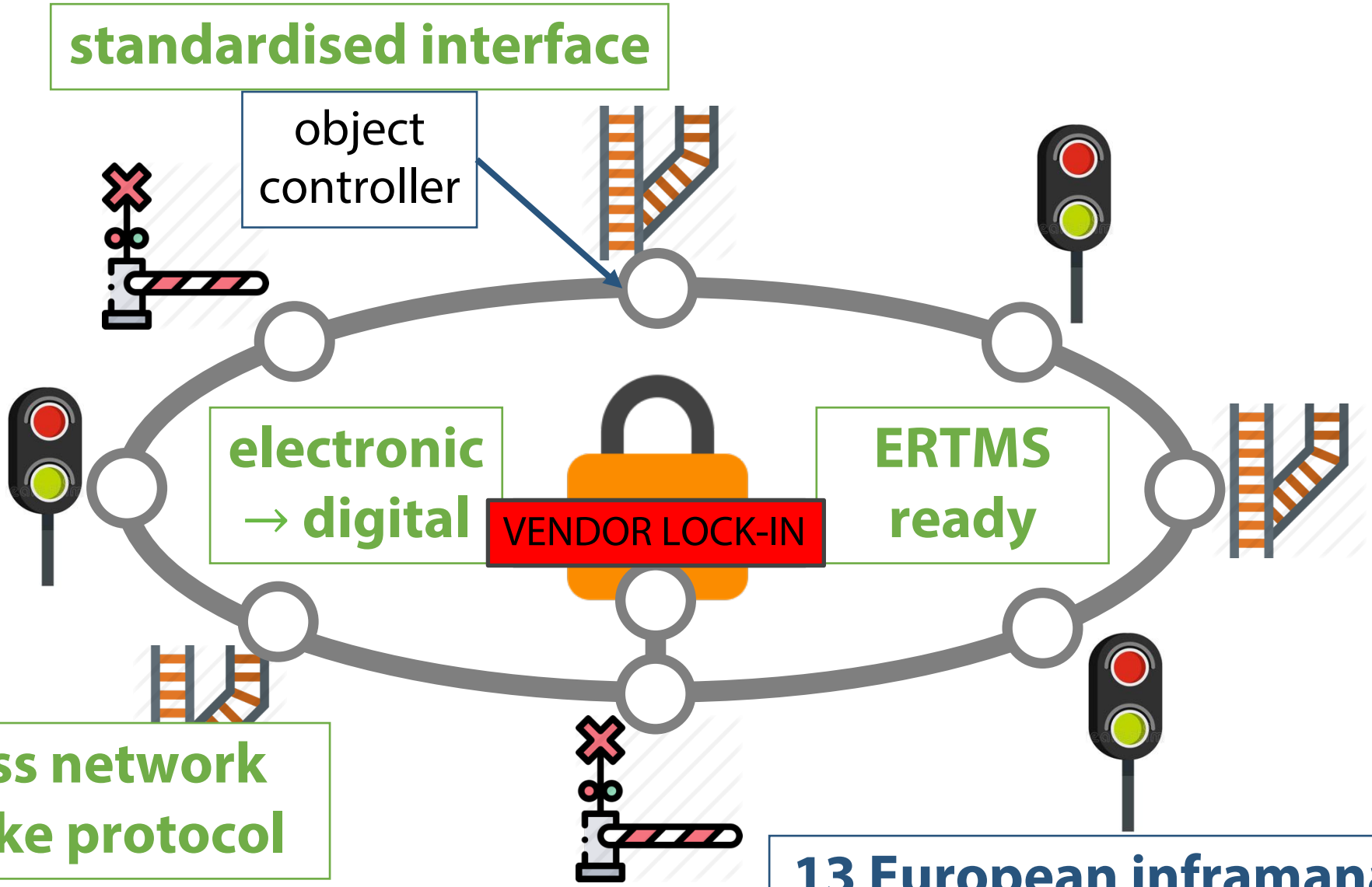
Mark Bouwman, Djurre van der Wal

Jaco van de Pol, Arend Rensink, Marielle Stoelinga

2. An application of FM in the development of ERTMS-HL3

PLAN FOR THIS PRESENTATION


Monolithic inflexible architecture
VENDOR LOCK-IN
Expensive special hardware
Many different installations



STANDARDISE ARCHITECTURE

OBVIOUSLY:

1. Standard should be ***clear*** and ***unambiguous***
2. Standard should be ***complete***
3. Standard should guarantee ***correct*** system behaviour
4. Compliance to standard should be thoroughly **tested**

BUT HOW TO ACHIEVE THIS FOR EULYNX?

TU/e

DB

- Growing interest in formal methods.

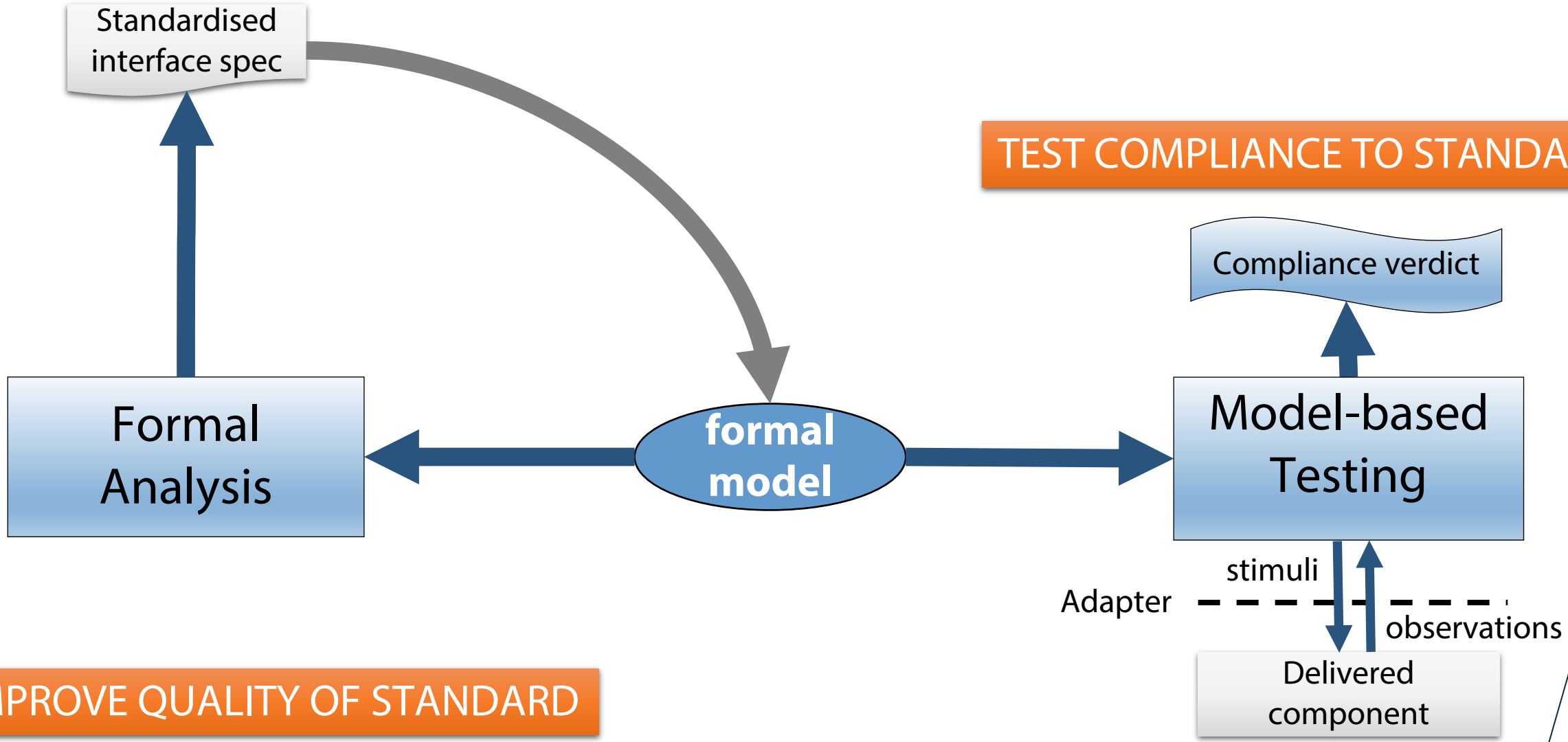
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

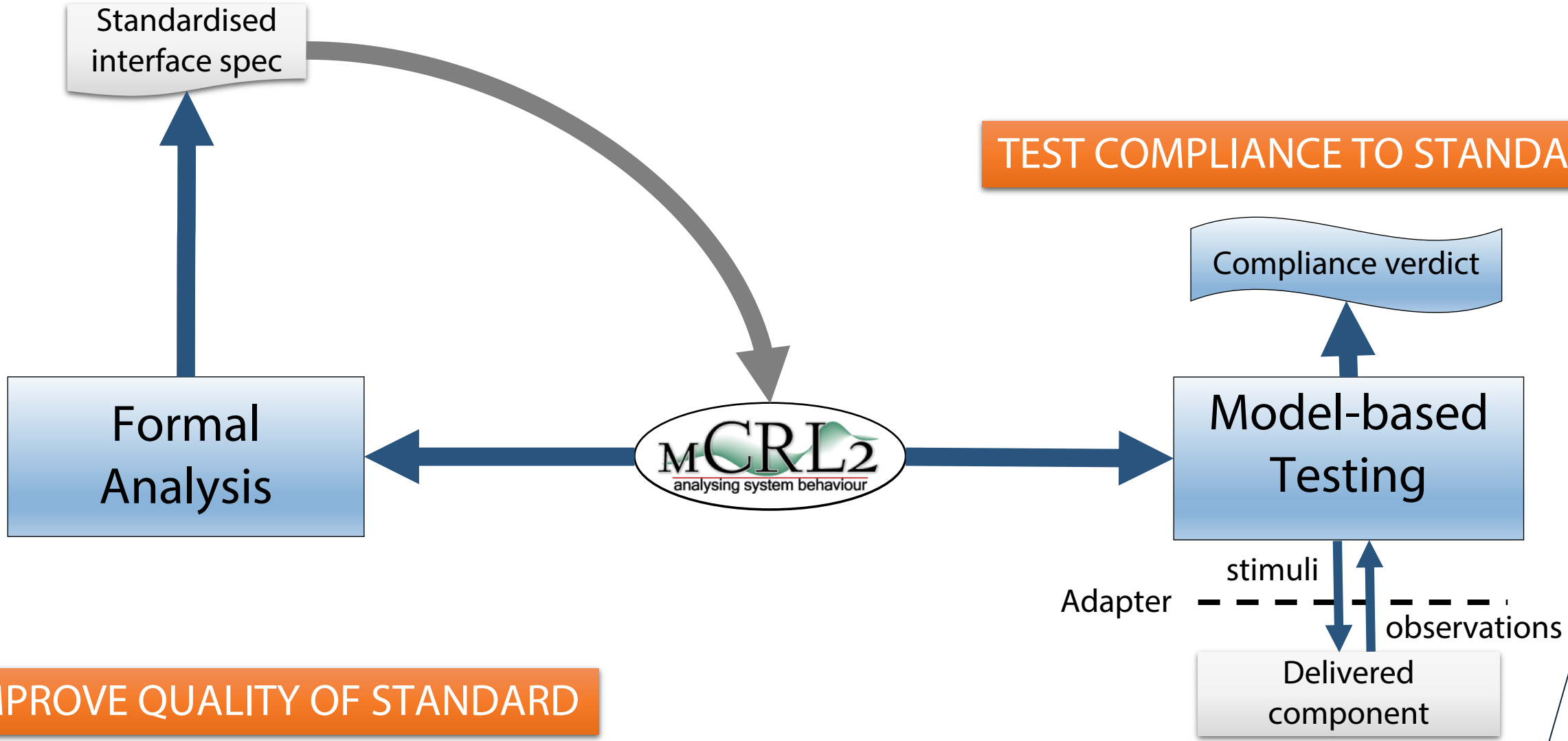
- Funded two 4-year PhD projects to investigate the application of formal methods in their context.

ProRail

- PhD students spend considerable time at Railway Infrastructure manager's premises.

UNIVERSITY OF TWENTE.


Standardised interface spec
TEST COMPLIANCE TO STANDARD
Compliance verdict
Formal Analysis
formal model
Model-based Testing
stimuli
Adapter
observations
Delivered component
IMPROVE QUALITY OF STANDARD


Standardised interface spec
TEST COMPLIANCE TO STANDARD
Compliance verdict
Formal Analysis
mCRL2
analysing system behaviour
Model-based Testing
stimuli
Adapter
observations
Delivered component
IMPROVE QUALITY OF STANDARD

https://www.mcrl2.org

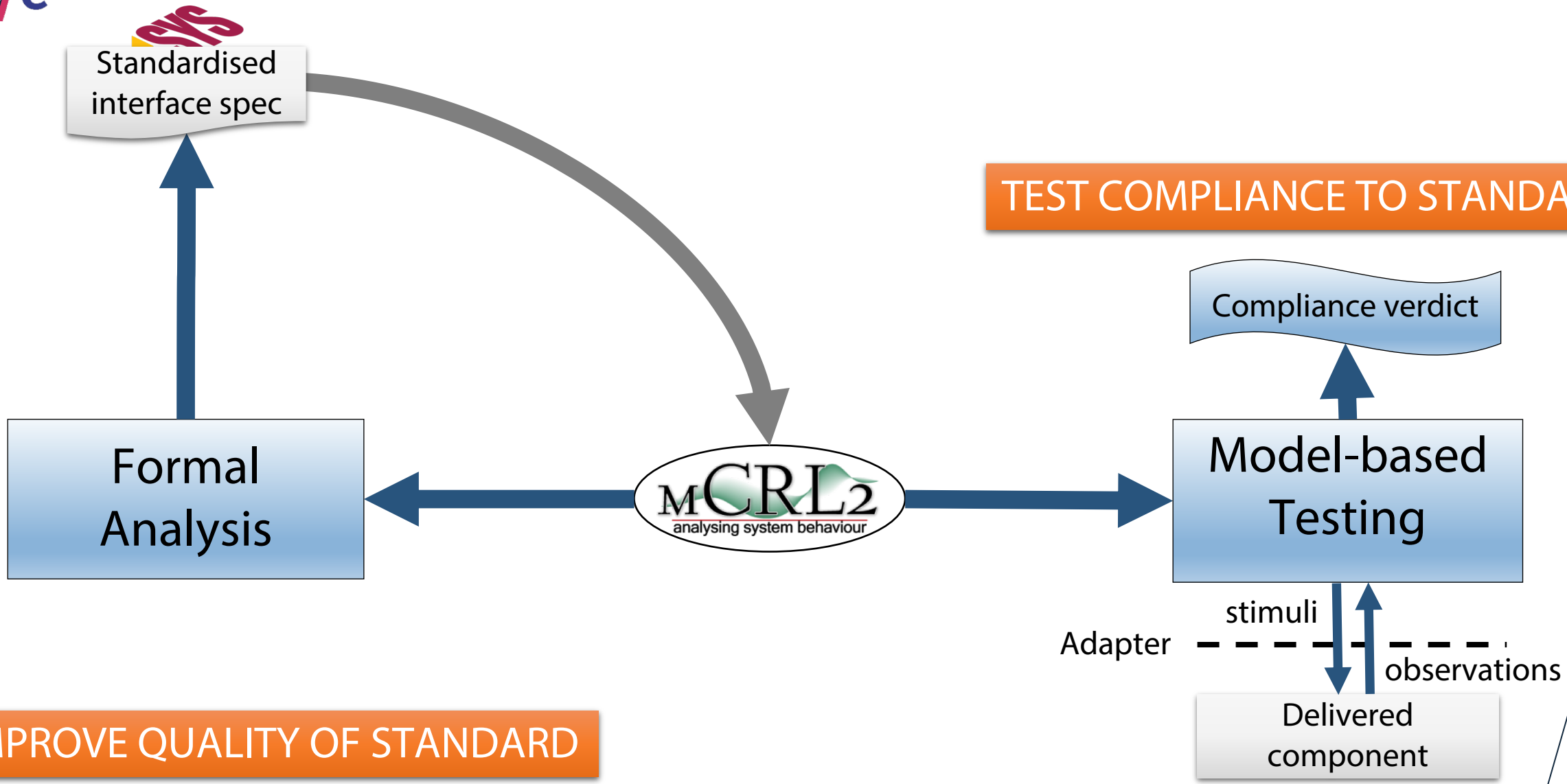
mCRL2
analysing system behaviour

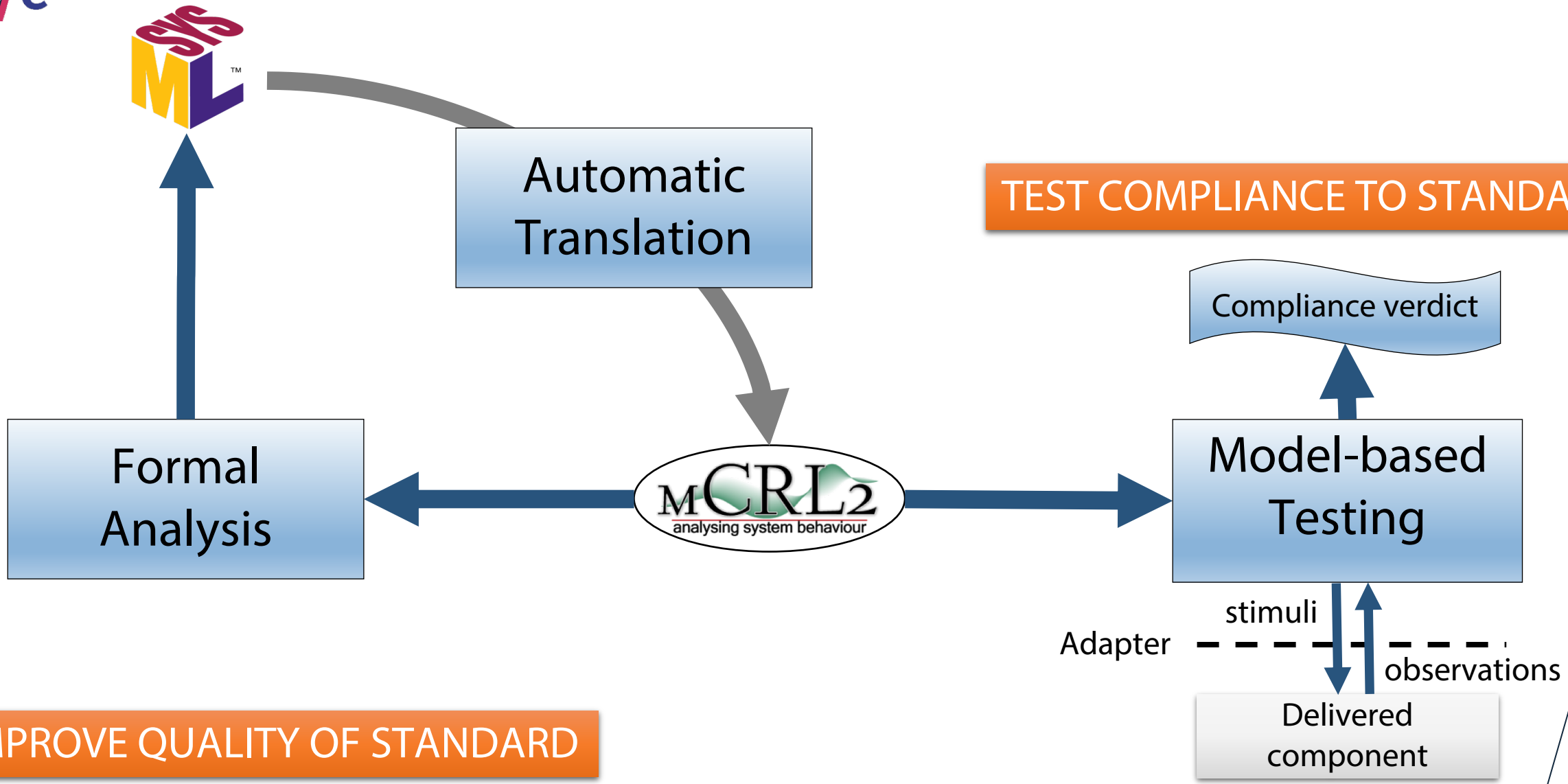
Simple, but expressive, process-algebra based modelling language

- Rich data language (sets, lists, functions, integers, …)
- Parallel composition of (recursively defined) sequential processes

Analysis

- Simulation
- Model checking
  - First-order extension of the modal μ-calculus
  - Extensive counterexample facility
- Behavioural equivalence checking
- Connection to model-based test tool JTorX


Standardised interface spec
TEST COMPLIANCE TO STANDARD
Compliance verdict
Formal Analysis
mCRL2
analysing system behaviour
Model-based Testing
stimuli
Adapter
observations
Delivered component
IMPROVE QUALITY OF STANDARD


Automatic Translation
TEST COMPLIANCE TO STANDARD
Compliance verdict
Formal Analysis
mCRL2
analysing system behaviour
Model-based Testing
stimuli
Adapter
observations
Delivered component
IMPROVE QUALITY OF STANDARD

FIRST CASE STUDY: POINT

Manual

~~Automatic~~ Translation

TEST COMPLIANCE TO STANDARD

Compliance verdict

Formal Analysis

mCRL2 analysing system behaviour

Model-based Testing

Input from railway engineers needed

stimuli

Adapter

observations

SysML simulator

~~Delivered component~~

IMPROVE QUALITY OF STANDARD

FIRST CASE STUDY: POINT

Verification results:

- **9 properties checked**
- **1 false (deadlock due to overflowing buffer)**
- **(Interpretation) of SysML models under discussion with DB**

Manual

Automatic Translation

Formal Analysis

Input from railway engineers needed

IMPROVE QUALITY OF STANDARD

TEST COMPLIANCE TO STANDARD

Compliance verdict

Model-based Testing

Test results:

- **Offline testing → generated 82 tests**
- **All tests passed (24/82 ended prematurely)**

Adapter

stimuli

observations

SysML simulator

Delivered component

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY ProRail DB UNIVERSITY OF TWENTE.

Point case study showed that FormaSig principle works

Next steps:

- Automate translation SysML → mCRL2
  - ➢ Semantics?
- Analyse correctness other object controllers
  - ➢ Requirements?
- Online model-based testing, coverage-based testing
- Test mutants, test real component

FORMASIG: NEXT STEPS

1. An application of FM in the development of EULYNX

   a) EULYNX, FormaSig (general ideas)
   b) First case study: Point
   c) Next steps

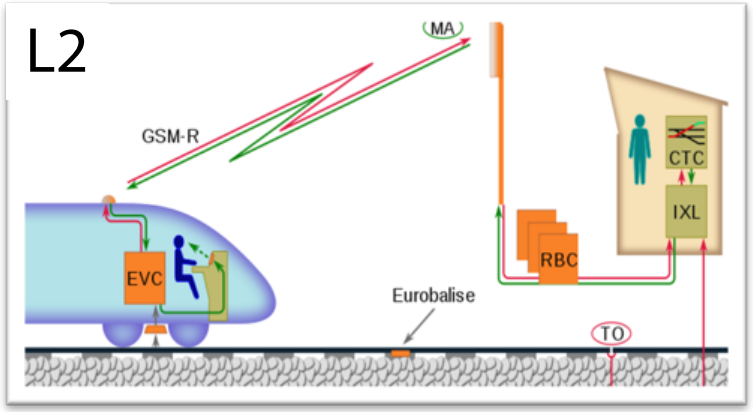
2. An application of FM in the development of ERTMS-HL3

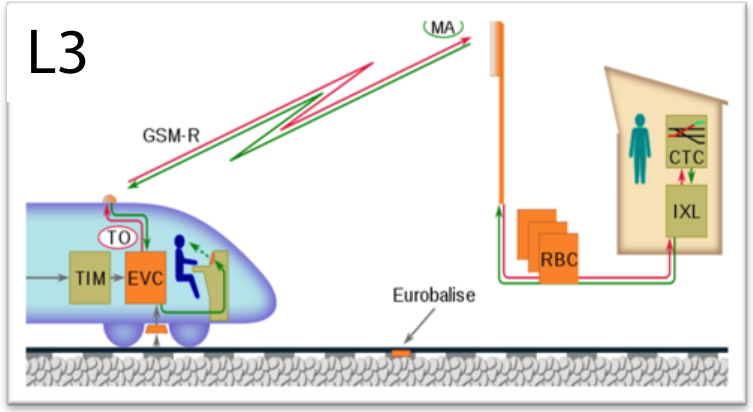
   a) Principles of ERTMS Hybrid level 3
   b) Formal analysis
   c) Next steps

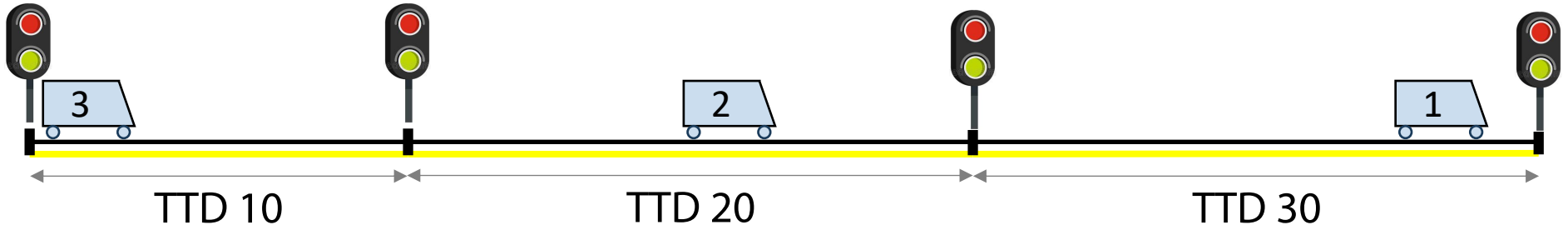
Collaboration with:

Maarten Bartholomeus
Rick Erkens
Tim Willemse

PLAN FOR THIS PRESENTATION

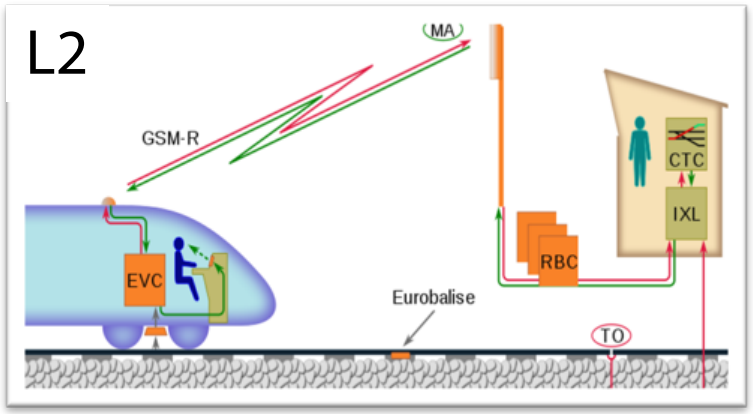


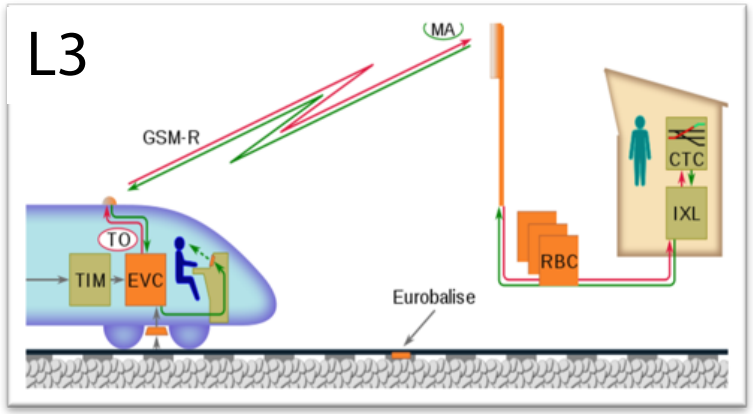




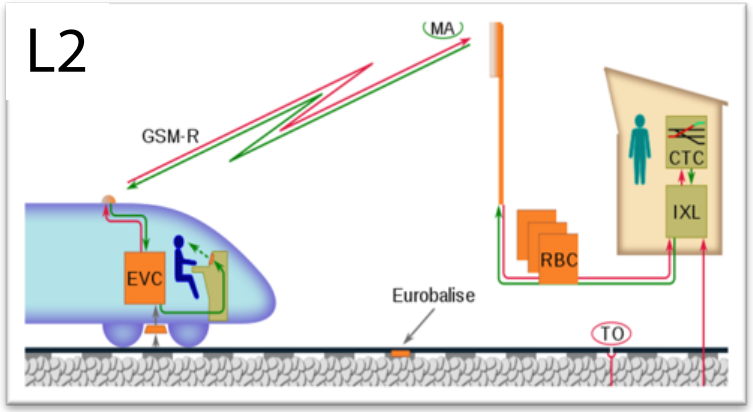
free

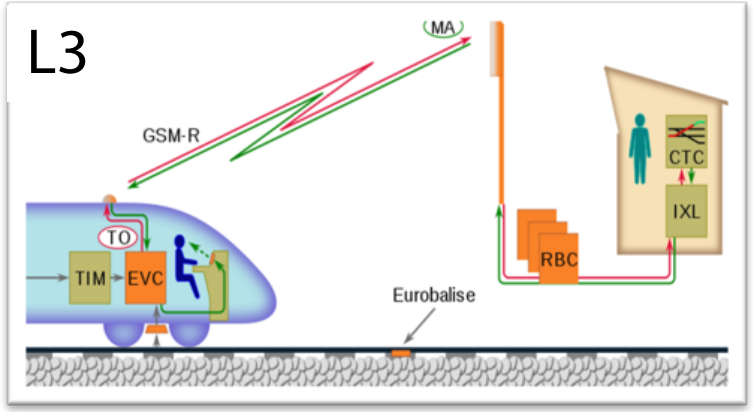
occupied





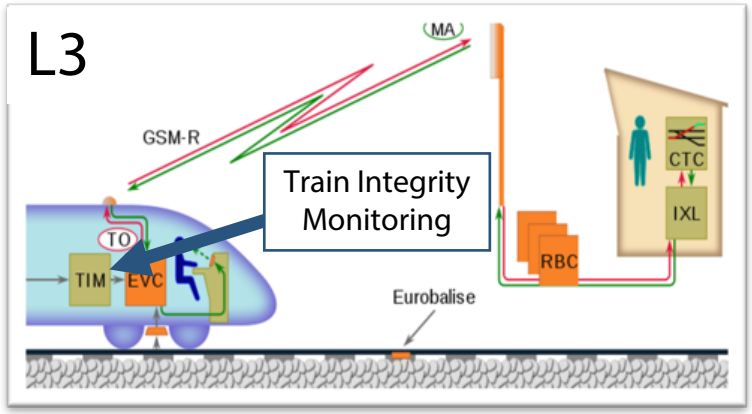
ERTMS LEVEL 3: ELIMINATE TRACKSIDE EQUIPMENT





ERTMS LEVEL 3

L3 promises considerable cost reduction and capacity increase



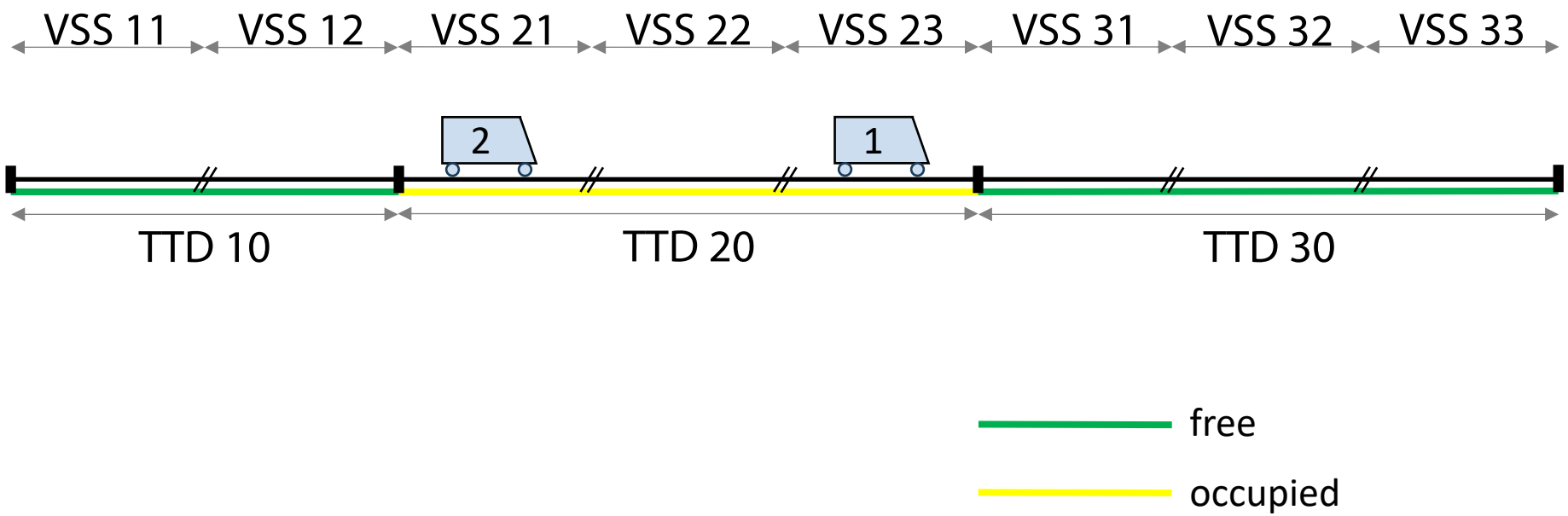
SERIOUS DRAWBACKS:

- All trains must have **TIM**
- Not very tolerant for radio connection problems

ERTMS Hybrid Level 3

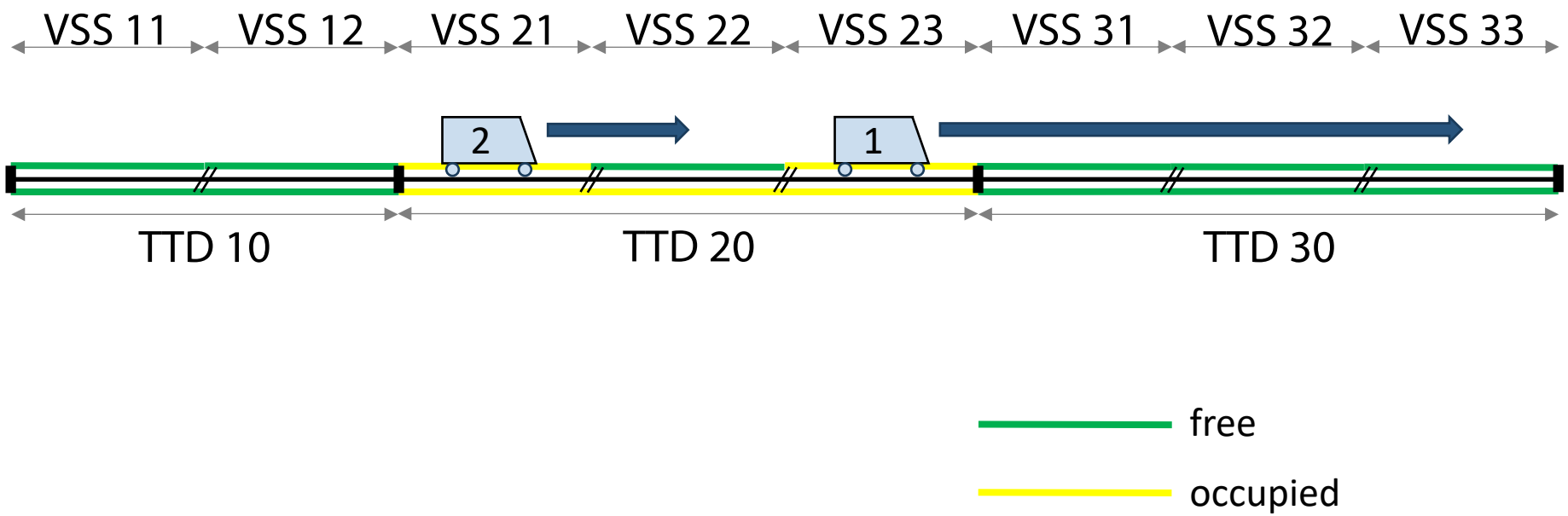
Main ideas:

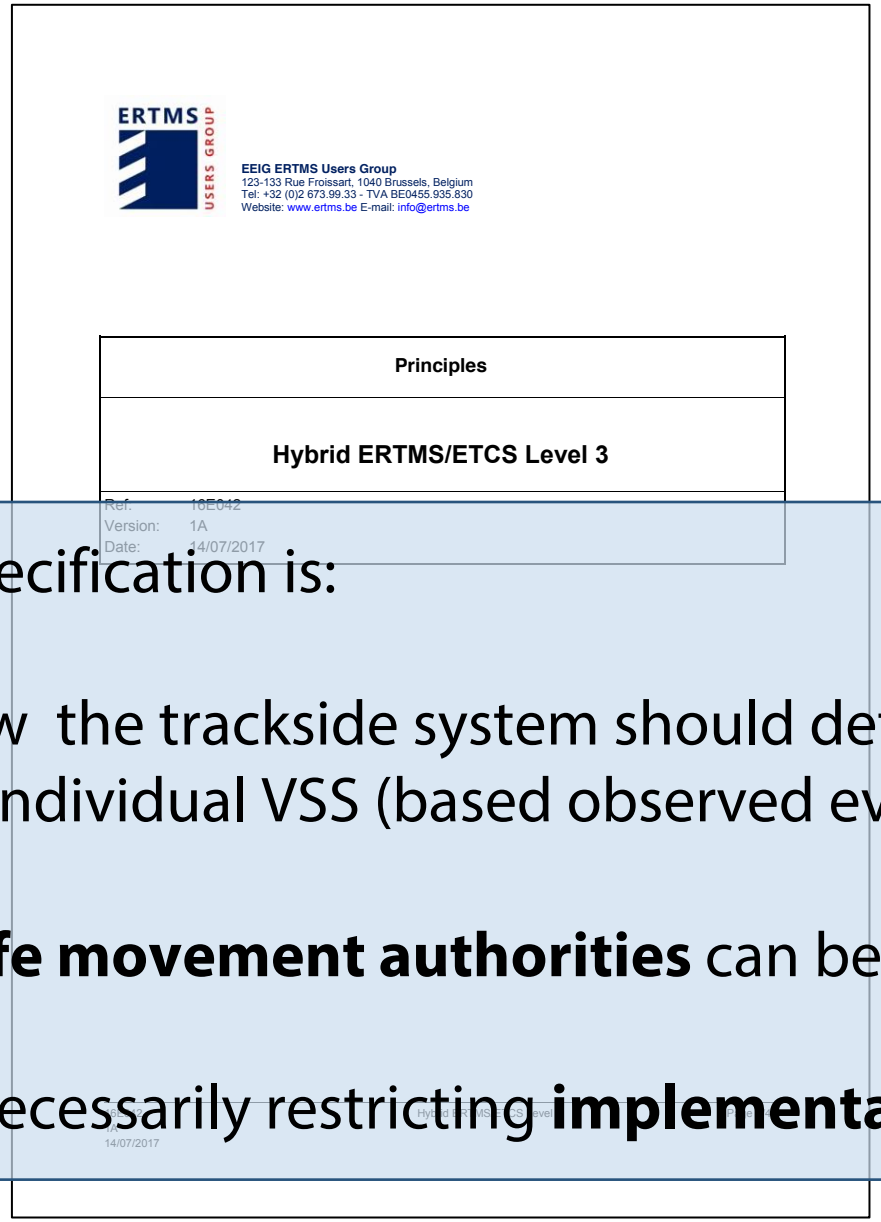
- Partition TTDs into *Virtual Sub-Sections* (VSSs)
- Multiple TIM-equipped trains can be on different VSSs of same TTD



Main ideas:

- Trackside maintain info on VSS status, based on regular train reports and TTD information
- Movement authorities issued based on VSS status



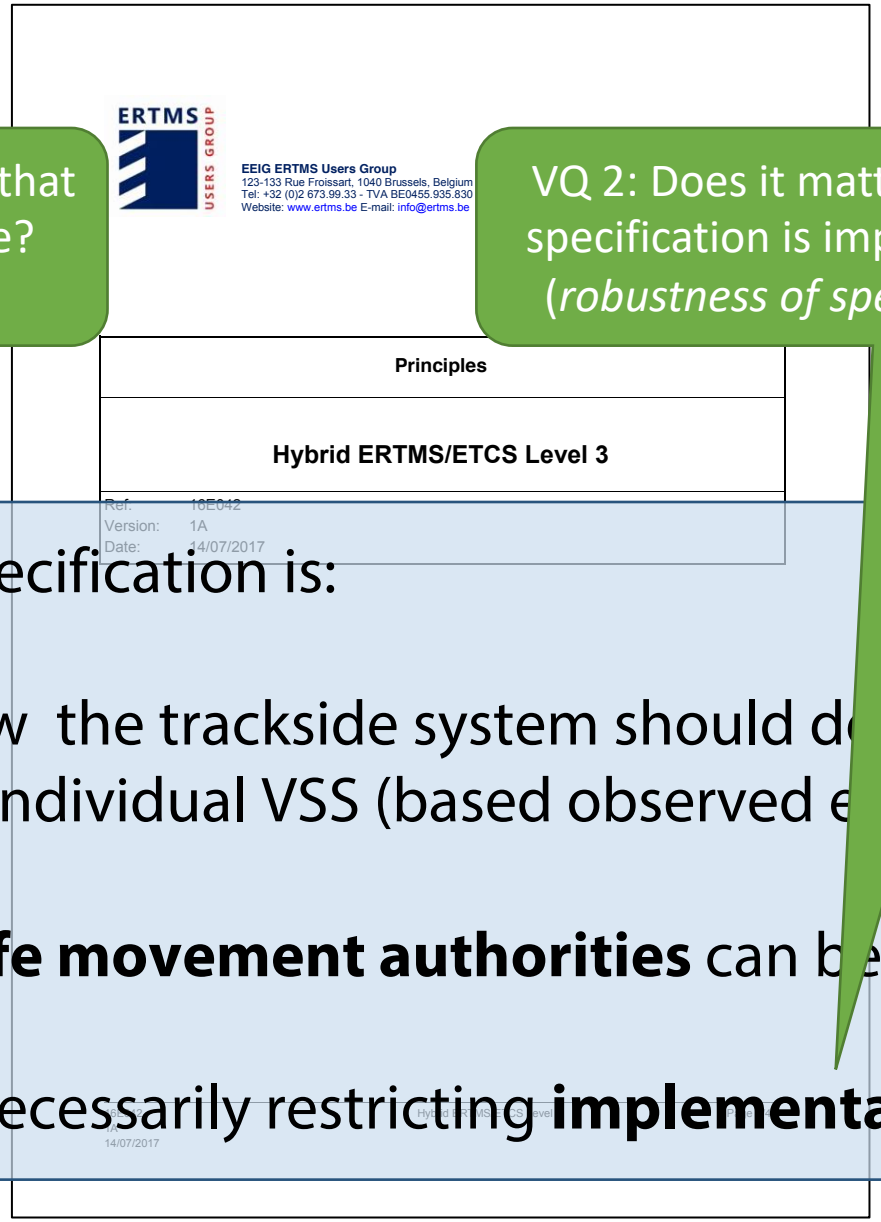


Purpose of the specification is:

- to describe how the trackside system should determine the status of each individual VSS (based observed events)…
- …such that **safe movement authorities** can be issued…
- …without unnecessarily restricting **implementation freedom**.

VQ1: Does the specification guarantee that issued movement authorities are safe? (*no collisions*)

VQ 2: Does it matter how the specification is implemented? (*robustness of specification*)



Purpose of t pecification is:

- to describe w the trackside system should d rmine the status of eac individual VSS (based observed e nts)…
- …such that **safe movement authorities** can b issued…
- …without unnecessarily restricting **implementation freedom**.

5 State machine for VSS

5.1.1.1 The Figure 7 represents the state machine of each VSS. The Table 2 gives the conditions for the transition from each state to each other. The sub-conditions (e.g. #1A, #1B) are always combined with a logical OR to give the result for the main condition, e.g. #4 = #4A OR #4B OR #4C.

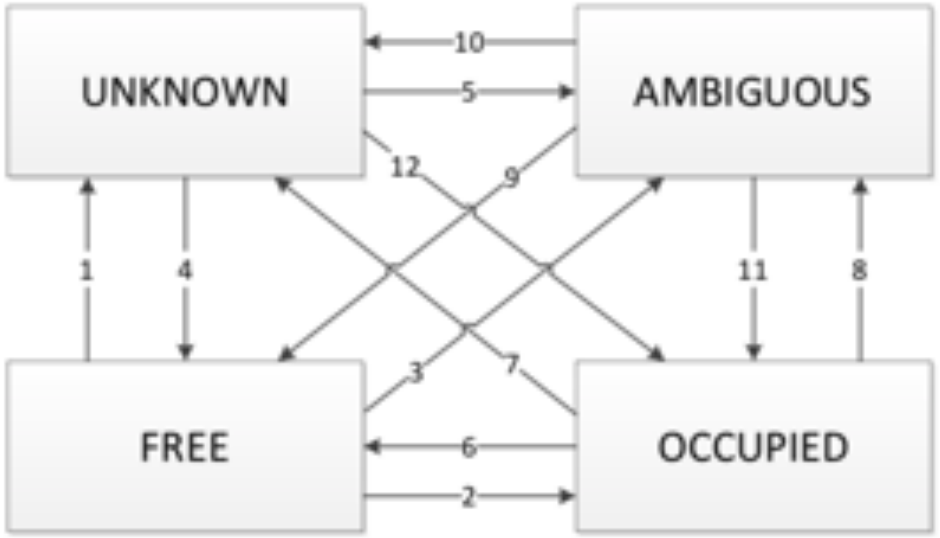


Figure 7: VSS section state diagram

		and the TTD for which the "ghost train propagation timer" is expired)		
	AND	(VSS is not located on the TTD for which the timer is expired)		
#2A		(TTD is occupied)	#3	3.3.3
	AND	(train is located on the VSS)		4.5.1.6
	AND	(VSS where the estimated front end of the train was last reported, was "occupied" after the processing of this previous position report)		
	AND	(current state of the VSS where the train was last reported is not "unknown")		
#2B		(TTD is occupied)	#3	3.3.3.6
	AND	(TTD in rear is free)		
	AND	(train location is not on the TTD)		
	AND	(VSS is the first VSS of the TTD)		
	AND	(VSS where the estimated front end of the train was last reported, "occupied" after the processing of this previous position report)		
	AND	(VSS is part of the MA sent to this train)		
#3A		(TTD is occupied)		

- Relevant data is easily modelled using mCRL2's data language
- Transition conditions can then be modelled as predicates.

5 State machine for VSS

5.1.1.1 The Figure 7 represents the state machine of each VSS. The Table 2 gives the conditions for the transition from each state to each other. The sub-conditions (e.g. #1A, #1B) are always combined with a logical OR to give the result for the main condition, e.g. #4 = #4A OR #4B OR #4C.

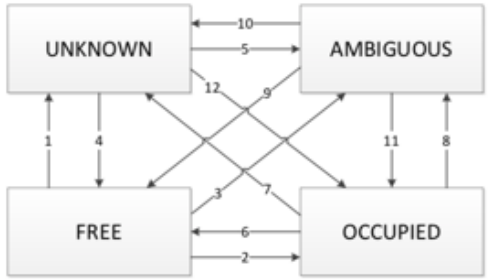


Figure 7: VSS section state diagram

TU/e

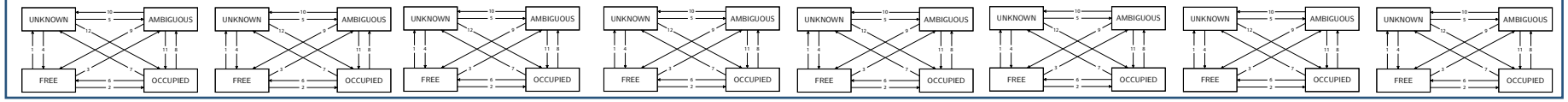
TRACKSIDE SYSTEM

Monitor TTD status

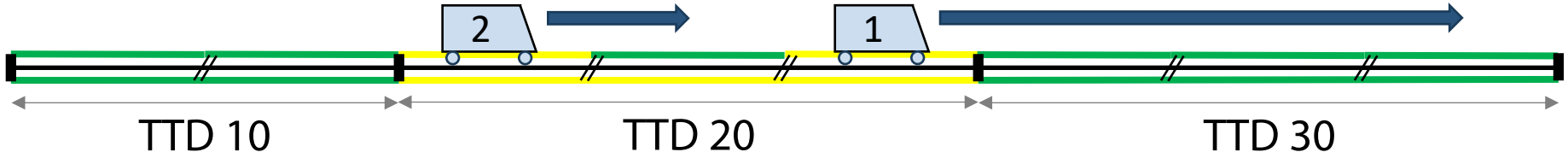
Receive position reports from train

Compute VSS status

Issue movement authority to train



VSS 11 VSS 12 VSS 21 VSS 22 VSS 23 VSS 31 VSS 32 VSS 33



bad weather behaviour

TRAIN

Move, break, disconnect, connect

Send position report to trackside system

Receive movement authority from trackside system

free

ambiguous

occupied

unknown

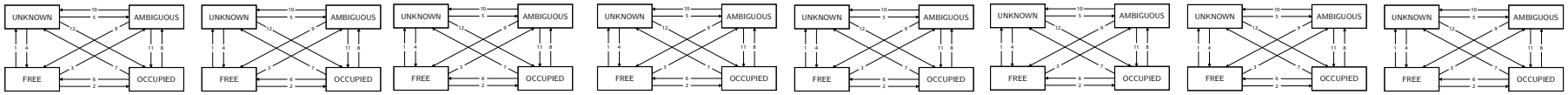
TRACKSIDE SYSTEM

Monitor TTD status

Receive position reports from train

Compute VSS status

Issue movement authority to train



Event-handling assumption:
After each event a **stable** VSS status must be reached before considering the next event.

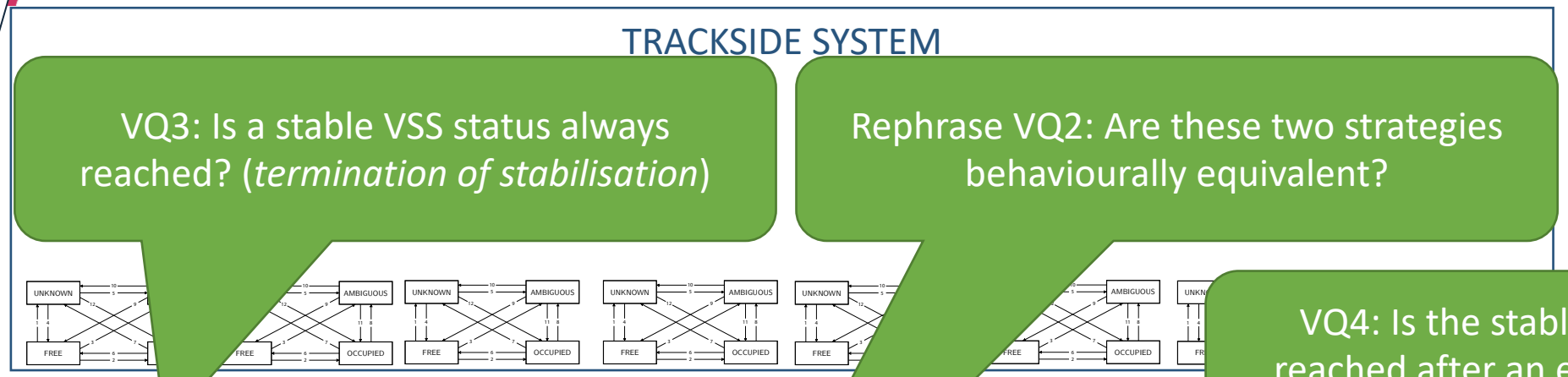
Immediate Update strategy:
Non-deterministically choose a VSS with an enabled transition and update its status.

Repeat until stable.

Simultaneous Update strategy:
First compute VSSs with transition enabled and apply updates simultaneously.

Repeat until stable.

COMPUTE VSS STATUS



VQ3: Is a stable VSS status always reached? (*termination of stabilisation*)

Rephrase VQ2: Are these two strategies behaviourally equivalent?

VQ4: Is the stable VSS status reached after an event unique? (*deterministic stabilisation*)

Event-handling assumption:
After each event a **stable** VSS status must be reached before considering the next event.

Immediate Update strategy:
Non-deterministically choose a VSS with an enabled transition and update its status.

Repeat until stable.

Simultaneous Update strategy:
First compute VSSs with transition enabled and apply updates simultaneously.

Repeat until stable.

COMPUTE VSS STATUS

HL3 version 1A did not satisfy VQ1– VQ4

We revised our model such that it does satisfy VQ1 – VQ4

HL3 version 1D seems very much in line with our revision

ProRail has asked us to upgrade our model to 1D because they want a simulator for the principles

TRAIN t GETS A MOVEMENT AUTHORITY

$$[\text{true}^*]\neg\left(\begin{array}{l}\exists t, t' : \text{TRAIN\_id}, v : \text{VSS\_id}. \\ \quad \langle \text{ExtendEoA}(t, v)\rangle \text{true} \\ \quad \wedge \\ \quad \exists l, l' : \text{List}(\text{VSS\_id}).(v = \text{rear}(l') \vee \text{in\_between}(\text{rear}(l'), \text{front}(l), v)) \\ \qquad \wedge \langle \text{report\_location}(t, l)\rangle \text{true} \wedge \langle \text{report\_location}(t', l')\rangle \text{true}\end{array}\right)$$

GLOBALLY IT IS NOT THE CASE THAT

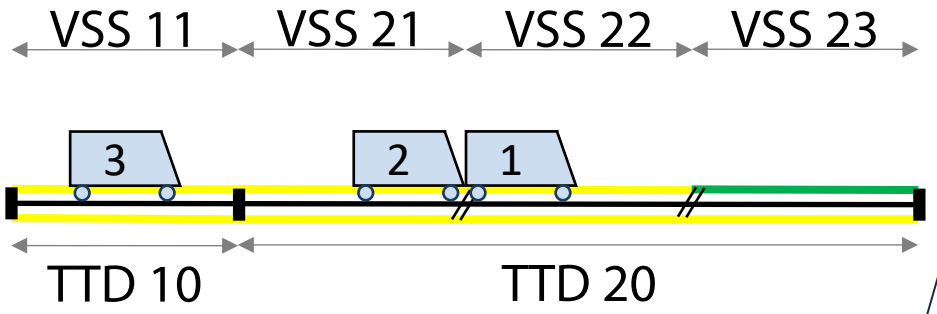
FOR A VSS THAT BELONGS TO THE LOCATION OF TRAIN t'

NO COLLISIONS?

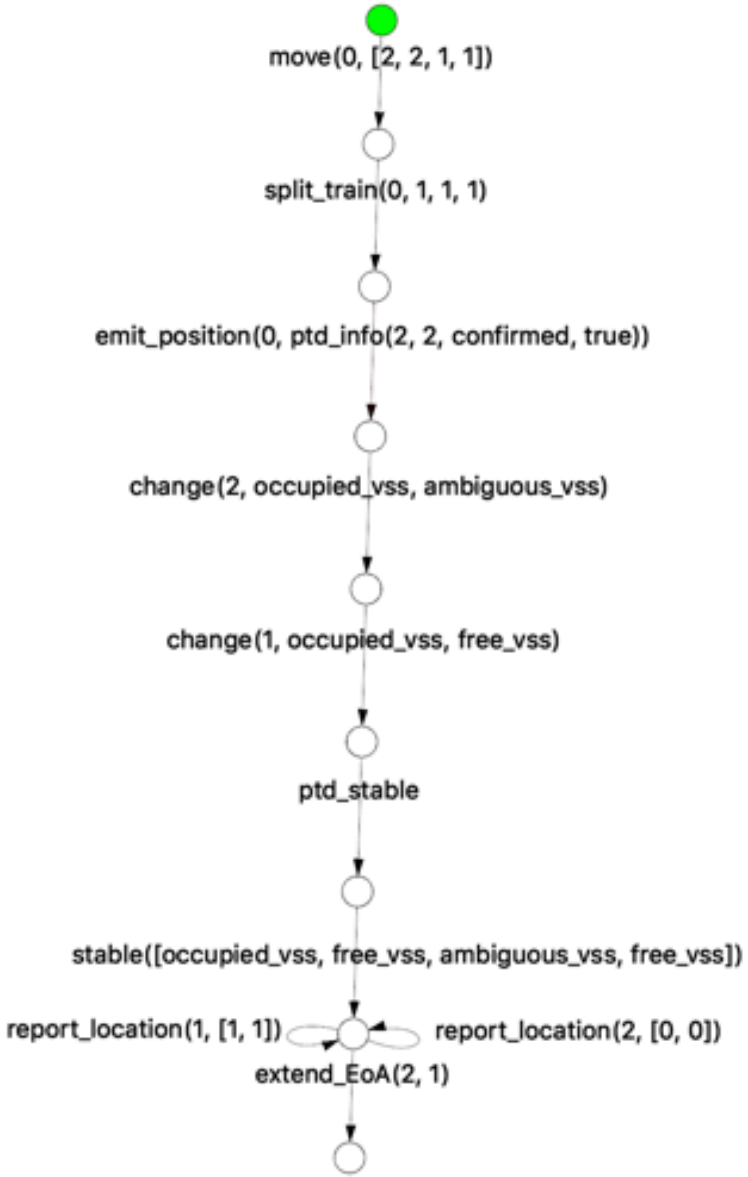
$$[\text{true}^*]\neg \left(\begin{array}{l} \exists t, t' : \text{TRAIN\_id}, v : \text{VSS\_id}. \\ \quad \langle \text{ExtendEoA}(t, v) \rangle \text{true} \\ \quad \wedge \\ \quad \exists l, l' : \text{List}(\text{VSS\_id}).(v = \text{rear}(l') \vee \text{in\_between}(\text{rear}(l'), \text{front}(l), v)) \\ \qquad \wedge \langle \text{report\_location}(t, l) \rangle \text{true} \wedge \langle \text{report\_location}(t', l') \rangle \text{true} \end{array} \right)$$

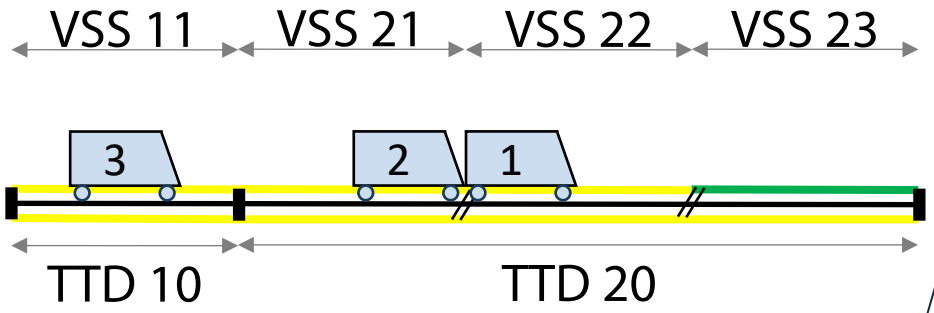
General procedure in mCRL2:

1. Apply appropriate abstraction
2. Generate state space
3. Minimise modulo behavioural equivalence
4. Generate PBES from formula and state space
5. Solve PBES
6. If false: obtain counterexample

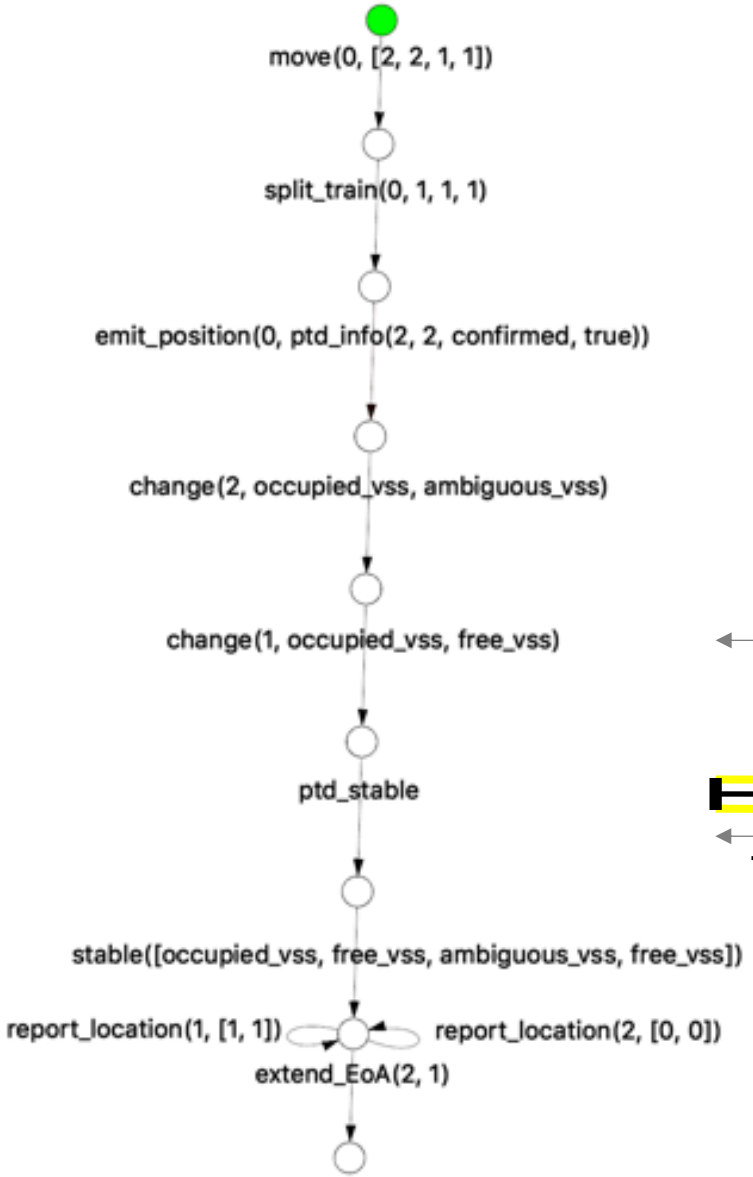


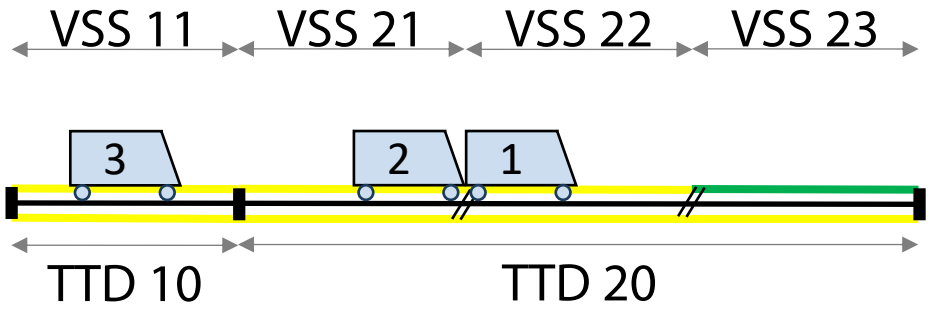
NO COLLISIONS?



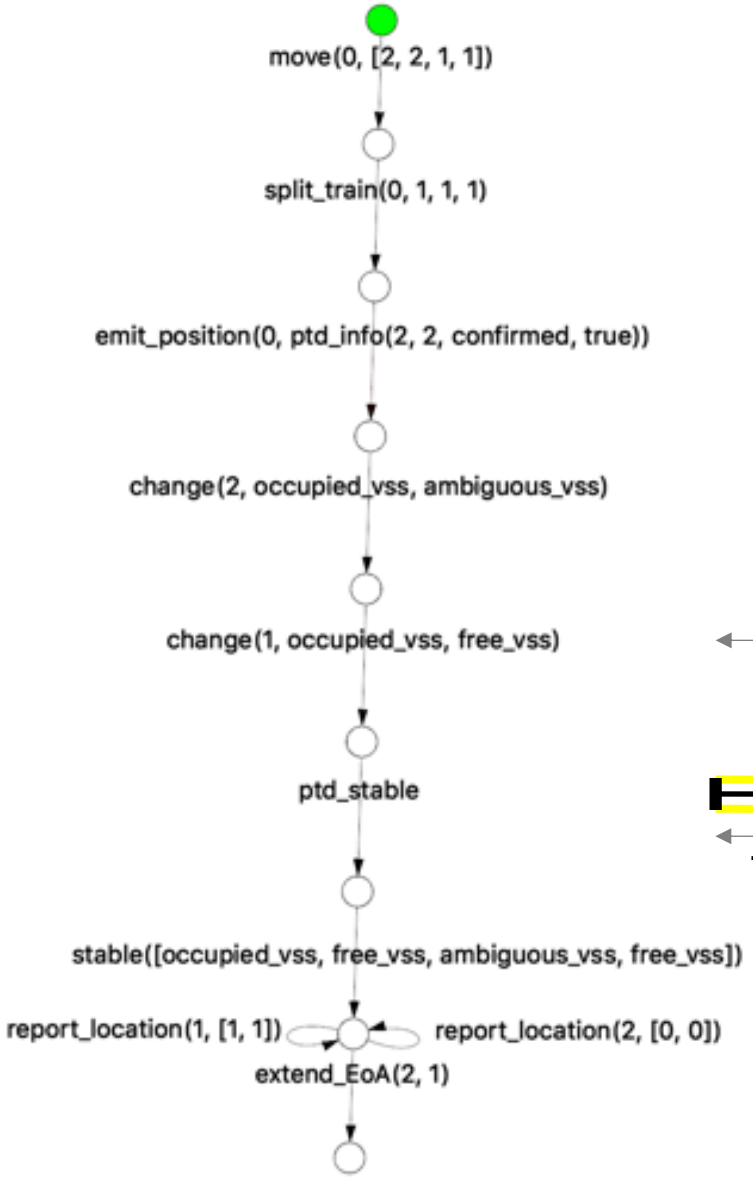


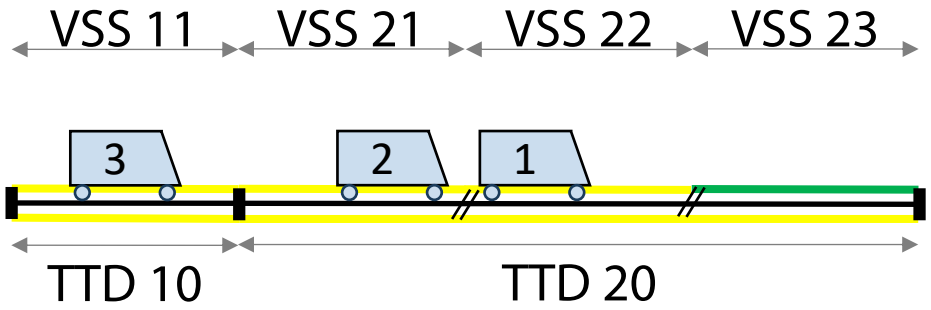
NO COLLISIONS: COUNTEREXAMPLE



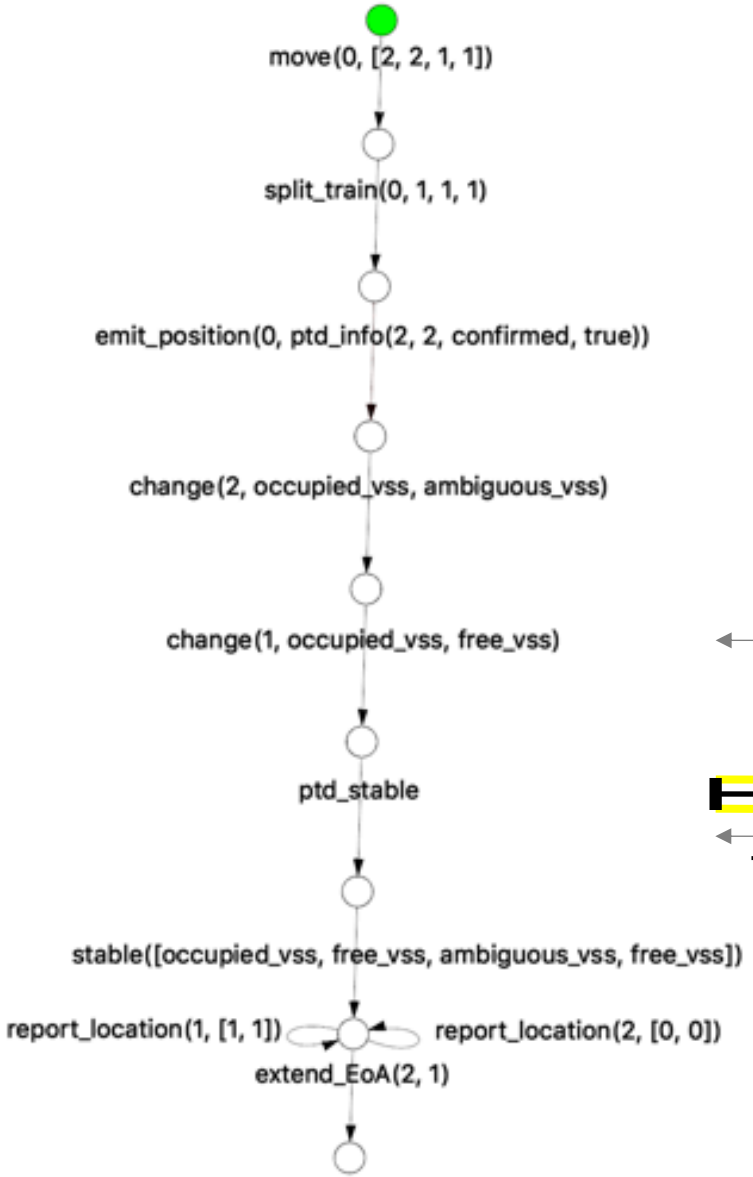


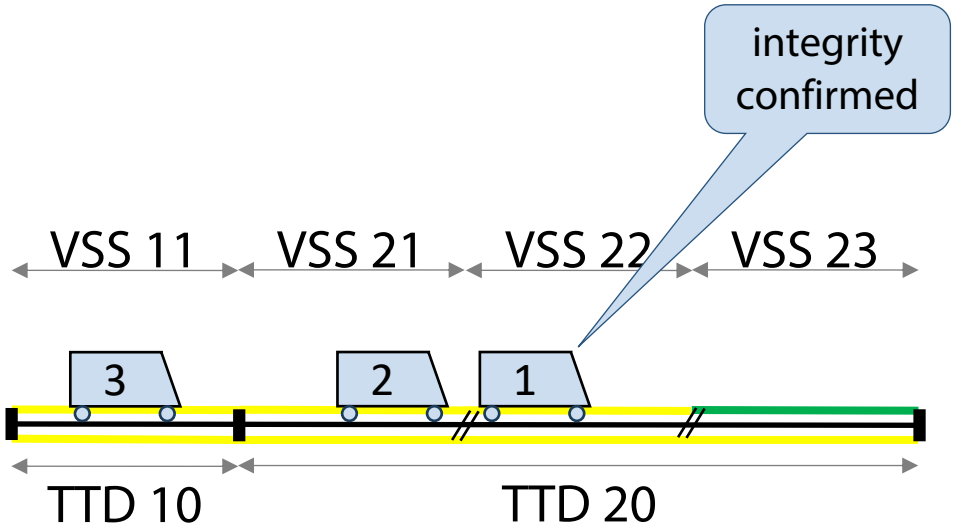
NO COLLISIONS: COUNTEREXAMPLE



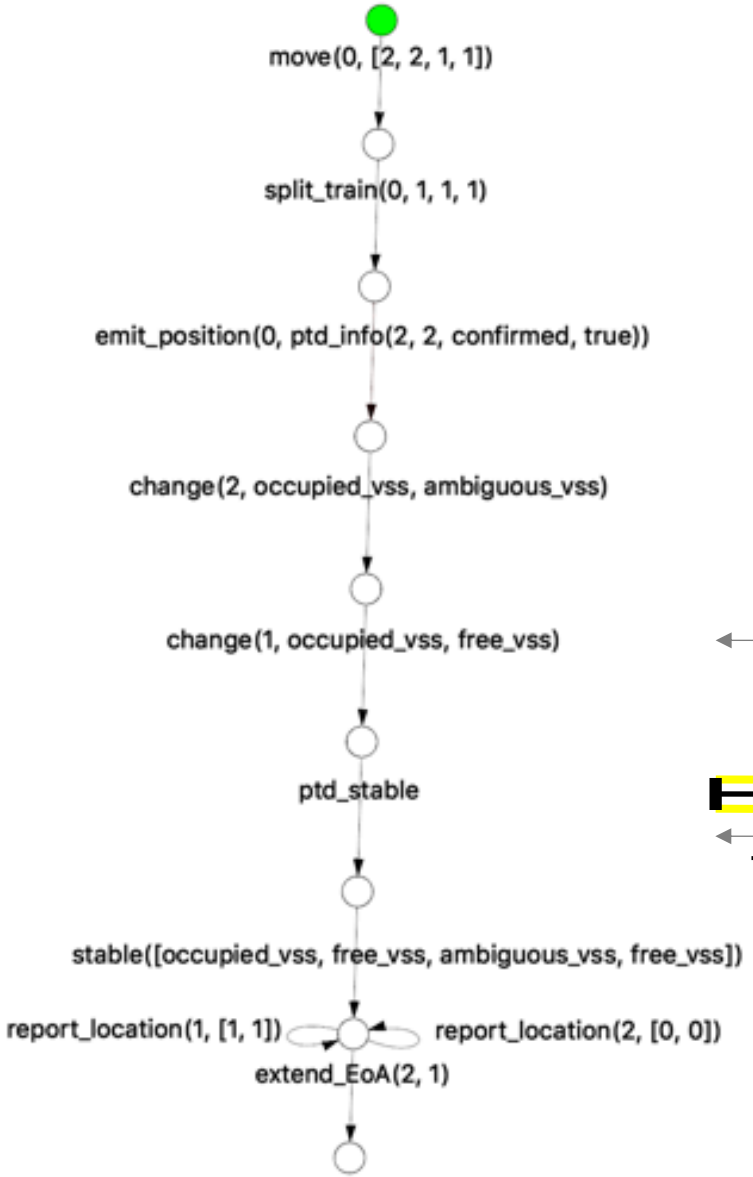


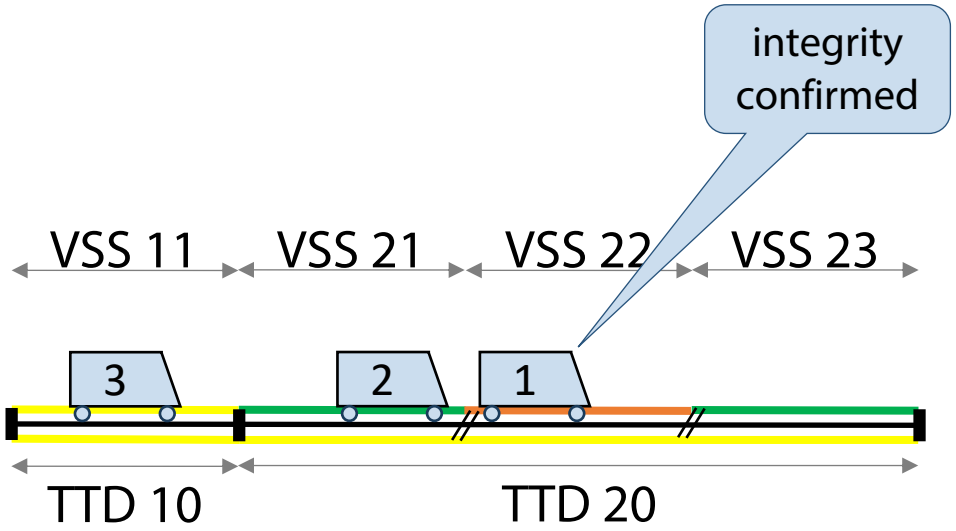
NO COLLISIONS: COUNTEREXAMPLE



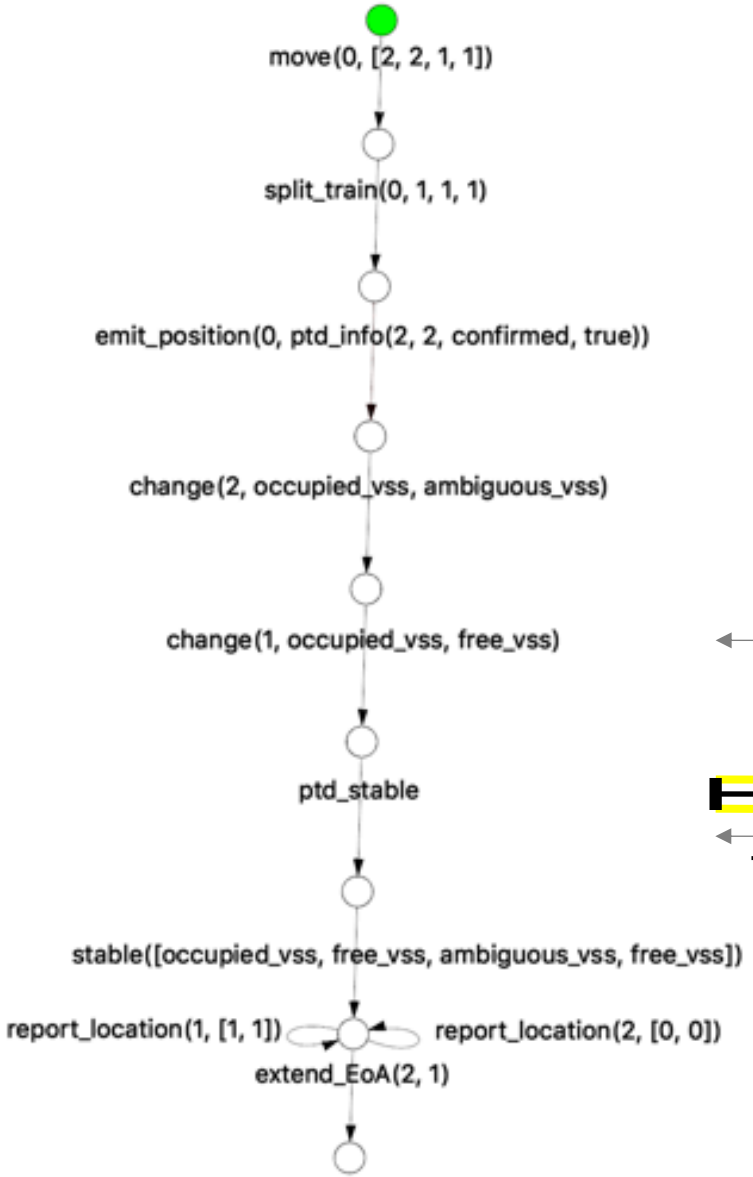


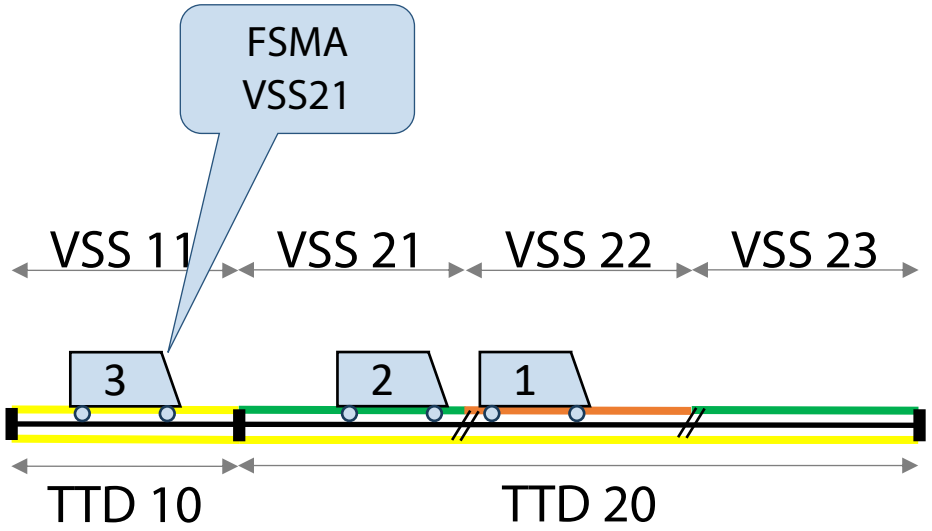
NO COLLISIONS: COUNTEREXAMPLE



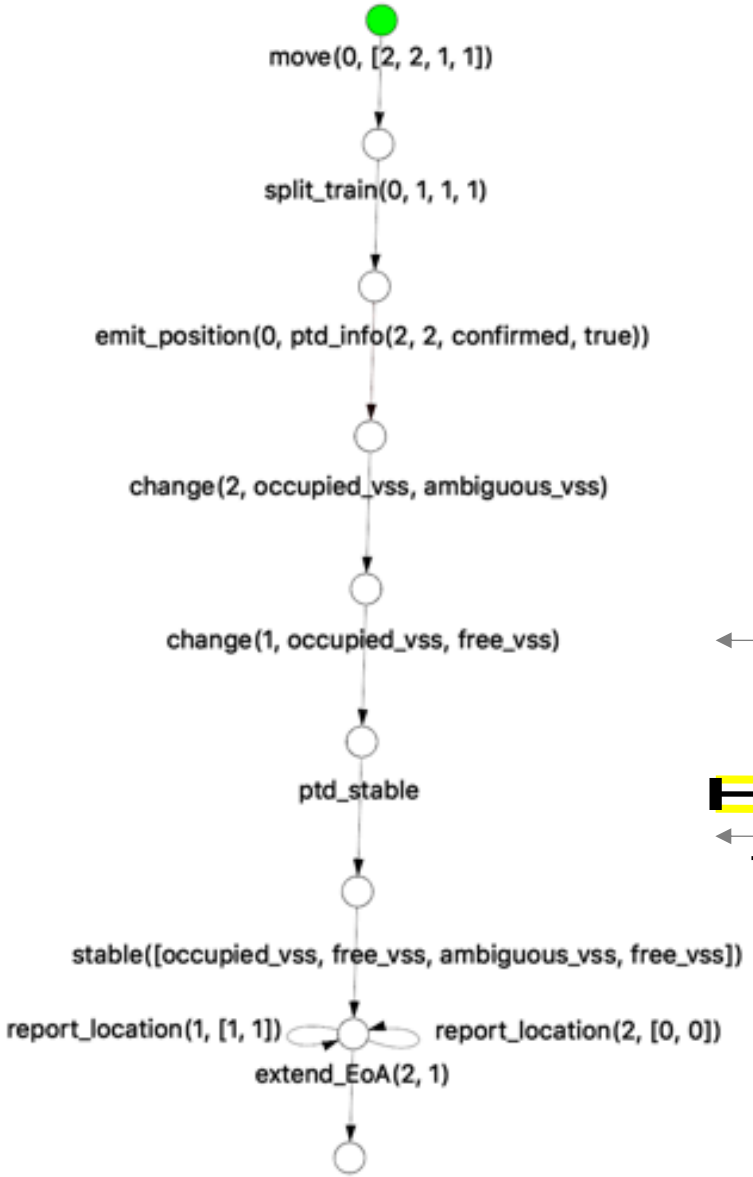


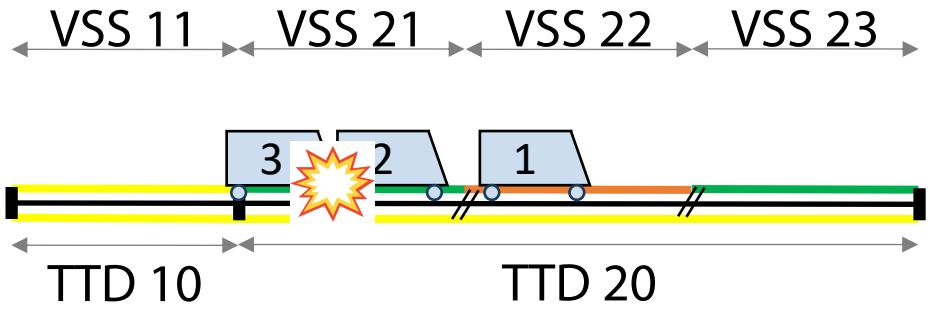
NO COLLISIONS: COUNTEREXAMPLE





NO COLLISIONS: COUNTEREXAMPLE





NO COLLISIONS: COUNTEREXAMPLE

THANK YOU FOR YOUR ATTENTION

QUESTIONS?