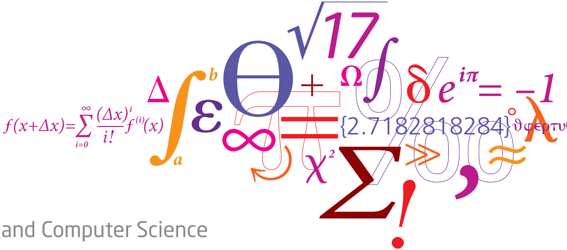
The RobustRailS Verification Tool Set

for Safety Verification of Interlocking Systems

Linh, H. Vu, Technical University of Denmark

Anne E. Haxthausen, Technical University of Denmark

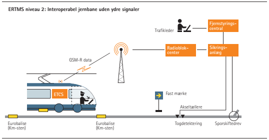
Jan Peleska, University of Bremen



DTU Compute

Department of Applied Mathematics and Computer Science

RobustRailS Verification Method & Tools



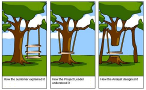
- Method and tool set for *automated*, *formal* *safety verification of interlocking systems*.
- Were developed by Linh H. Vu, Anne Haxthausen, Jan Peleska, in collaboration with the Danish railways in the RobustRailS. research project.
- *RobustRailS* research project, 2012-2017:
  - **Funded** by the Danish Innovation Fund.
  - **Partners:** 4 DTU departments, Bremen University, Banedanmark, Traffic Authorities, DSB, DSB S-train.
  - **Goal:** to develop methods for achieving punctual and safe railway operations

    for the Danish Re-signaling Program implementing ERTMS/ETCS Level 2.
    - methods for efficient safety verification
    - ...

Background: Challenges

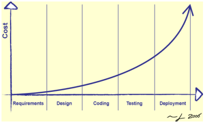
- *Errors* in interlocking systems may have very severe consequences.

- *Conventional specification & verification methods* may be time consuming and not give sufficient guaranties for correctness.


How the customer explained it
How the Project Leader understood it
How the Analyst designed it

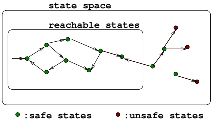
Submitter
Reviewer
NO NEED TO DOUBLE CHECK THIS CHANGE LIST, IF SOME PROBLEMS REMAIN THE REVIEWER WILL CATCH THEM.
NO NEED TO LOOK AT THIS CHANGE LIST TOO CLOSELY, I'M SURE THE AUTHOR KNOWS WHAT HE'S DOING.

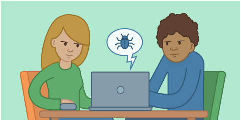
- Bugs typically first found during testing ⟶ expensive to fix.



- ⟶ *Need to get it right from the beginning*.

Smarter Specification and Verification Methods



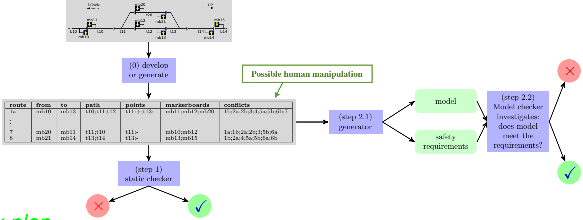


Use Formal Methods and Automation:

- strongly recommended by CENELEC 50128 for safety-critical software
- efficient
  - to avoid bugs
  - to catch bugs early, before implementation and test

⟶ saves time and money

RobustRailS Verification Method & Tools



route	from	to	path	points	markerboards	conflicts
1a	mb10	mb13	t10;t11;t12	t11:+;t13:-	mb11;mb12;mb20	1b;2a;2b;3;4;5a;5b;6b;7
⋮						
7	mb20	mb11	t11;t10	t11:-	mb10;mb12	1a;1b;2a;2b;3;5b;6a
8	mb21	mb14	t13;t14	t13:-	mb13;mb15	1b;2a;4;5a;5b;6a;6b

1.1 Input: *track plan*.

1.2 The tool automatically generates a *route control table*, if not provided.

1.3 The tool checks that the track plan and route control table are correct.

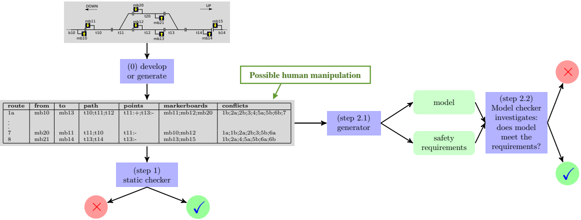
2.1 The tool generates

- a *formal model* of the behaviour of the interlocking system
- *formal safety requirements* (e.g no train collisions).

2.2 A model checker (dis-)proves the model meets the requirements.

3.1 The tool generates *test cases* and a *test oracle* for software integration testing.

RobustRailS Verification Method & Tools



(0) develop or generate

Possible human manipulation

route	from	to	path	points	markerboards	conflicts
1a	mb10	mb13	t10;t11;t12	t11:+;t13:-	mb11;mb12;mb20	1b;2a;2b;3;4;5a;5b;6b;7
⋮						
7	mb20	mb11	t11;t10	t11:-	mb10;mb12	1a;1b;2a;2b;3;5b;6a
8	mb21	mb14	t13;t14	t13:-	mb13;mb15	1b;2a;4;5a;5b;6a;6b

(step 1) static checker

✗ ✓

(step 2.1) generator

model

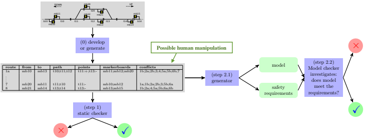
safety requirements

(step 2.2) Model checker investigates: does model meet the requirements?

✗ ✓

- Verification in three steps:
  - The *static checking step* is used to find errors in the control table.
  - The *model checking step* is used to find errors in the control algorithms.
  - The model-based *testing step* is used to find errors in the implemented system.
- Features:
  - *“Model hiding”:* Models automatically generated from domain-specific railway specifications
    ⟶ can be used by railway engineers without background in formal methods.
  - Verification based on induction reasoning using bounded model checking *pushes the limits for state space explosion*.

Applications of the Method & Tools



(0) develop or generate

Possible human manipulation

route	from	to	path	points	markerboards	conflicts
1a	mb10	mb13	t10;t11;t12	t11:+;t13:-	mb11;mb12;mb20	1b;2a;2b;3;4;5a;5b;6b;7
⋮						
7	mb20	mb11	t11;t10	t11:-	mb10;mb12	1a;1b;2a;2b;3;5b;6a
8	mb21	mb14	t13;t14	t13:-	mb13;mb15	1b;2a;4;5a;5b;6a;6b

(step 1) static checker

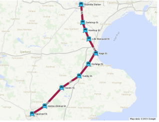
(step 2.1) generator

model

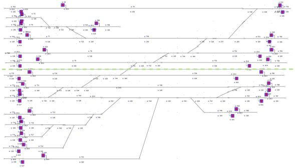
safety requirements

(step 2.2) Model checker investigates: does model meet the requirements?

- The Early Deployment Line, Roskilde - Næstved, in Denmark [Vu, Haxthausen, Peleska 2017]:

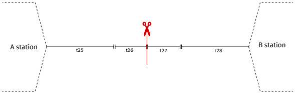


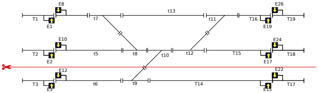
- Florence station in Italy [Fantechi, Haxthausen, Macedo 2017]:



Compositional Verification

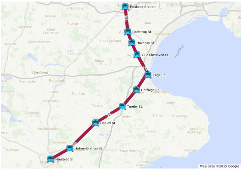
- Suggested by **Fantechi, Haxthausen, Macedo** 2017-... .
- Goal: to further *increase the scalability* of the verification method.
- Idea: *cut* the interlocking logic of large layouts *into separate, more manageable, portions*, so that proving safety of the portions implies safety of the whole.

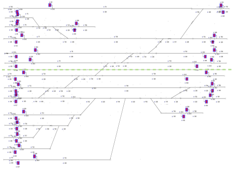




- Experiments show: compositional verification is $2.5 - 3\times$ *faster*, uses $30 - 40\%$ *less memory*.

Early Deployment Line (EDL) in Denmark and Florence Station in Italy





Thank you for your attention.