Automated, Formal Verification of Safety Requirements for Interlocking Systems

Linh Hong Vu (lvho@dtu.dk)
Anne E. Haxthausen (aeha@dtu.dk)
Jan Peleska (jp@verified.de)

DTU Compute
Department of Applied Mathematics and Computer Science

Outline

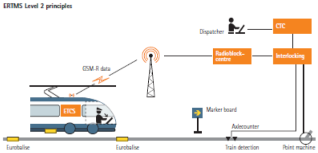
1. Background

2. Method

3. Conclusion

Introduction

- **Context**: The Danish Signalling Programme[1] (2009-2021) - replace the railway signalling systems in the entire country with standardized ERTMS/ETCS Level 2
- **ERTMS/ETCS**: European standardized railway traffic management/train control systems → seamless railway travel through Europe
- **RobustRailS**: (Robustness in Railway OperationS[2])
  - Funded by the Danish Strategic Research Council
  - Accompanies the Danish Signalling Programme on a scientific level
- **(One of the) goals**: Provide methods and tools supporting *efficient* modelling and verification of railway control systems (WP.4.1)
  → primary focus: ETCS Level 2 compatible interlocking systems



Source: ertms.net

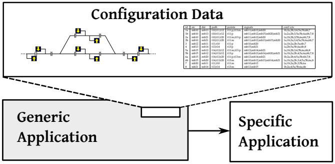
1 http://www.bane.dk/signalprogrammet

2 http://robustrails.man.dtu.dk

Interlocking Systems

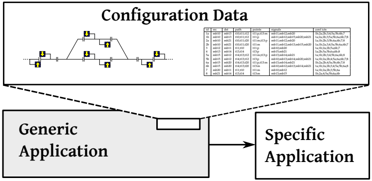
- **Interlocking system**: A signalling system component that is responsible for *safe* routing of trains through the (fraction of) railway network under its control
- **Safety-critical**: A vital component with highest safety integrity level (SIL4)
- **Our goal**: A method for efficient verification of safety requirements (no collisions, no derailments) for the new Danish interlocking systems

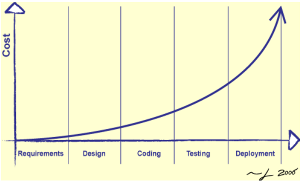
Conventional Development of Interlocking Systems



- An application consists typically of:
  1. a generic part
  2. configuration data: the railway network and an interlocking table.
- Once and for all:
  - *Informal* specification, design, and implementation of *generic application*.
  - *Informal, manual* verification of generic application (*"type certification"*).
- For each installation:
  - Creation and *Informal, manual* validation of the configuration data.
  - Instantiation of the generic application by means of configuration data.
  - Verification of the resulting specific application by testing.

Problems in Conventional Development





- *Manual, informal* specification, validation and verification are *time-consuming* and *error-prone*.
  → Some errors are first detected when testing specific applications → costly.

We need a better method:

1. *Formal verification:* use formal methods.
2. *Automated verification.*
3. *Easy to use.*
4. *Discover errors as early as possible.*
5. *Scalable.*

Formal Methods

- **Formal Methods**: employ mathematically based languages, techniques, and tools for specifying and verifying software/hardware systems.
- *Advantages*:
  - Unambiguous
  - Support advanced analysis techniques in early phases (specification, design) of the development cycle.
  - ...

  → strongly recommended by CENELEC 50128 for SIL4 applications
- *Obstacles*:
  - Not easy to use, require training
  - Scalability: state explosion problem – *the size of a verification problem increases exponentially with the number of components* → exhaust the limited computing resources

  → our method addresses these obstacles

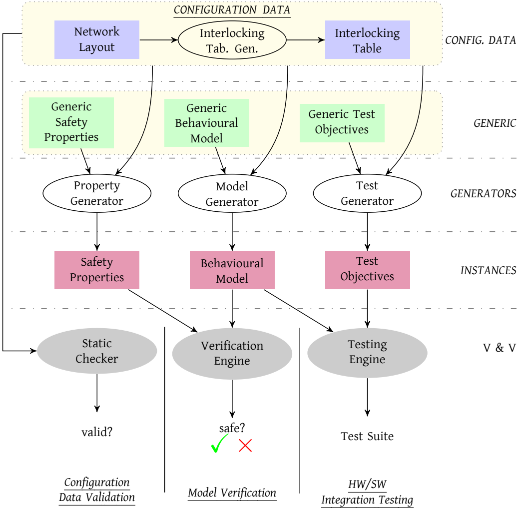
Outline

1. Background

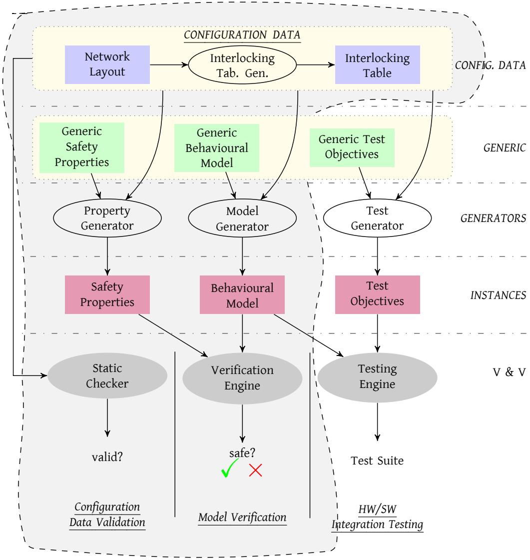
2. Method

3. Conclusion

Method Overview

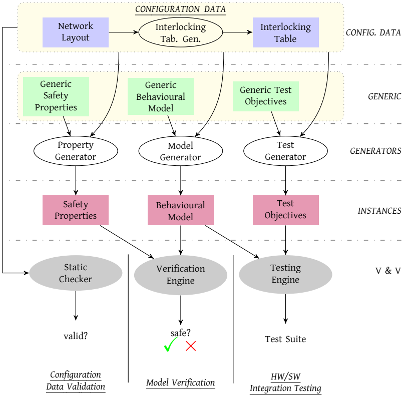


Method Overview



How is it better?

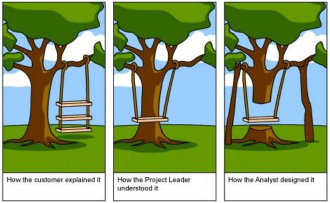
1. Formal
2. Automated
3. Easy to use
4. Discover errors efficiently and early
5. Scalable

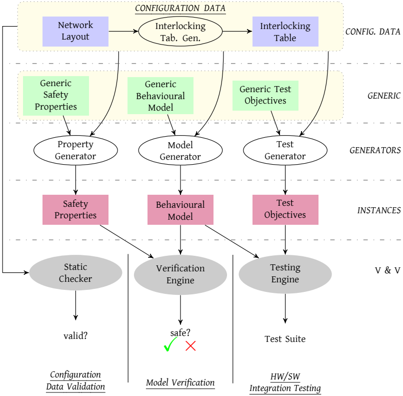


Formal

Based on mathematical models and techniques

- Unambiguous
- Facilitate advanced mathematical analyses on specifications and designs
- Provide better understanding of the systems
- Models can be use as the base for implementation



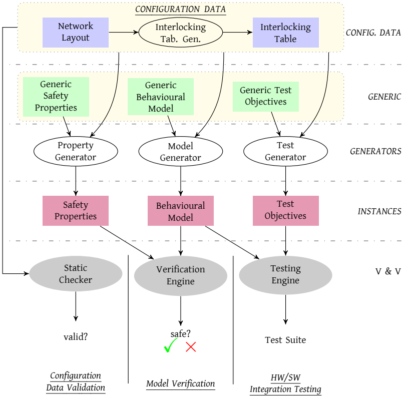


Automated

Most of the steps in the flow are *automated*

- Interlocking table generation
- Validation of configuration data
- Instantiating the generic application
- Verification of safety properties
- Test generation and execution

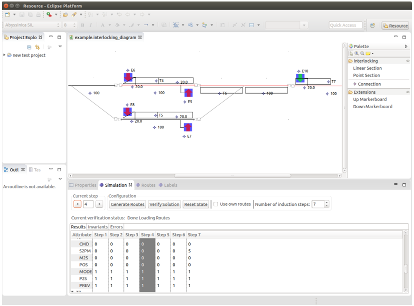
→ "press-a-button": quick and efficient



Easy to use

Encapsulate the underlying mathematical artefacts by familiar concepts and notions.

- *Configuration Data*: graphical editor or XML input (e.g. exported from CAD)
- *Generic Application*: a railway tailored language with familiar concepts, notions such as Route, Signal, Point, etc.
- Visualize erroneous situations



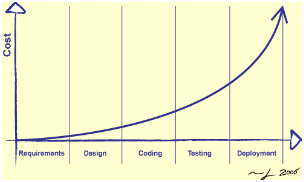
→ mathematical artefacts are generated

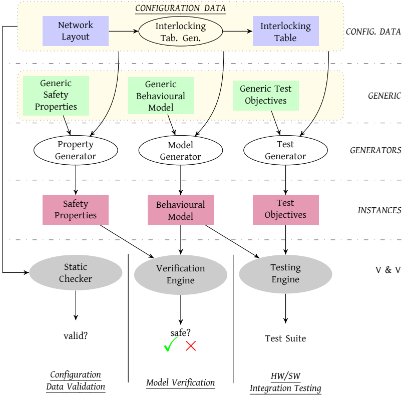
→ minimal training is required

Discover errors efficiently and early

Errors are revealed *as early as possible* by a 3-step V&V

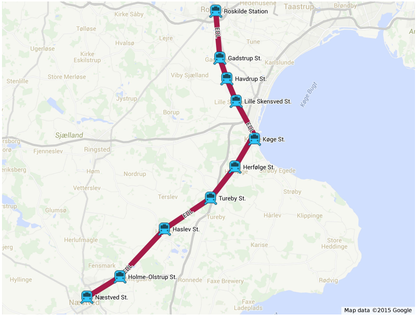
1. *Configuration Data Validation*: e.g., route protection, conflict routes are correct.
2. *Model Verification*: safety requirements are verified on the designs
3. *HW/SW Integration Testing*: implementation conforms to the formal model





Scalable

- Tackle the *state explosion problem* by using advanced verification techniques.
- Verified safety requirements for the Early Deployment Line (EDL): 8 stations (largest: Køge), one interlocking.
- No other research group has been able to formally verify an interlocking system of this size.



Conclusion

- Interlocking systems: SIL4 $\rightarrow$ efficient safety verification is crucial
- Formal methods are strongly recommended by CENELEC for SIL4
  $\rightarrow$ Issues: not easy to use, state explosion
- A method for verification of safe requirements for interlocking systems
  1. Formal
  2. Easy to use
  3. Automated
  4. Discover errors efficiently and early
  5. Scalable (was successfully applied to the Early Deployment Line)
- Related work: advanced state-of-the-art by the size of verifiable interlocking models.
- Future work:
  - Push the size of verifiable interlocking models even further
  - Technology transfer to industry

Questions?