Advances in
Railway Control Systems Architectures
and
Related Challenges for Verification and Validation

Jan Peleska
University of Bremen and Verified Systems International GmbH
jp@verified.de
A Novel Distribution Paradigm.
Cloud-based Railway Control
Cloud-based Railway Control

• Siemens Mobility DS³ – Distributed Smart Safe System

• IXL, RBC and related functionality are moved into the cloud

• Functions run safely on standard HW, standard OS (Windows, Linux), and standard VMs

• Cloud servers communicate with track element controllers via high-speed backbone and Ethernet

• see Siemens Mobility publication [1]
Safety @ COTS multicore

- RBC (Radio Block Centre)
- IXL (Interlocking System)
- OCS (Occupation Control System)
- S&D
- Any SW

Track element controllers for points, signals, axle counters...

Source: see [1]
Motivation for this Architecture

- Excellent scalability
- Excellent performance through state-of-the-art servers and networks
- Significant availability improvements enabled by
  - Reconfigurable software allocation on different CPU cores and servers
  - Geographic distribution
Motivation for this Architecture

• **Cost reduction** enabled by
  
  • COTS operating systems and virtual machines
  
  • COTS hardware – virtualisation removes HW dependencies
  
  • Mixed SIL (Safety Integrity Levels) runnable on the same HW
  
  • Legacy software running in emulators on high-performance COTS servers
Challenges

• Ensure *fail-safe behaviour* on unsafe HW, OS, VM

• **Safe synchronisation** between geographically distributed components

• **Safe reconfiguration** during system operation

• Complexity is so high that **no complete formal overall model** of system behaviour and system architecture can be created
Design Characteristics

Source: see [1]
Design Characteristics

Create **fail-safe behaviour** using principles of the **coded monoprocessor**: A specific approach to **software diversity**

No specialised HW required, since **cloud servers can emulate coded monoprocessor hardware** and perform managed code execution

**Coded monoprocessor – recall.**

Use of coded data

\[ x \mapsto (x_f, x_c) \]

\[ x_c = A \cdot x_f + B_x + D_t \]

A : transformation factor
B : static signature
D : dynamic signature

Verification of redundant channel information

\[ z_c = A \cdot z_f + B_z + D_t \]

\[ (z_c - B_z - D_t) \mod A = 0 ? \]
Design Characteristics

- **Coded monoprocessor**

- **Strict cyclic processing**
- **Synchronisation** of redundant software components by logical clock
- **Memory scattering**
- **Coded data** and associated diverse transformation operations
- **Calculation of work flow digest** values
- **Dynamic data signatures** ensure use of the data at correct point in time
- Encryption with **complementary keys** ensures that data can only be used if all redundant components have calculated the equivalent result.
Design Characteristics

Increase **reliability** by means of n-modular redundancy and m-out-of-n voters.

Source: see [1]
Design Characteristics

Dynamic reconfiguration ...

... even across geographically distributed server farms

Source: see [1]
Cloud-based
Railway Control –
V&V Challenges
V&V Challenges: many different SW and system paradigms to be integrated

- Coded Monoprocessor
- Distributed Deployment
- SAFE PROTOCOLS
- Distributed Clock Synchronisation
- n-Modular Redundancy
- Publish-Subscribe Pattern
- HW Emulation
- Fail-safe Behaviour
- Generics
- Safety Software Patterns
- Legacy Software Execution
- Hard Real-Time Guarantees
- Security Mechanisms
- Dynamic Reconfiguration
- Virtual Machines
- Multicore Processing
V&V Solutions
Side Remark – why Models are so Important

- Formal models/specifications are highly recommended according to standard EN 50128

- We need them for
  - specification validation by model checking and simulation
  - automated code generation
  - automated model-based testing
  - enabling traceability between requirements, code, tests, and other V&V artefacts
Scenario Models

- **Coping with model complexity** – an approach adopted from the field of autonomous vehicles, see [2]
  
  - Identify scenarios
  
  - Develop collection of per-scenario models
  
  - Parameterised models specifying the required behaviour **for a specific operational situation**
Automated Model-based Testing

- Coping with large amount of test cases
  - Test case/test data generation and test procedure generation from models can be fully automated
  - Test suite execution may be parallelised by using cloud services
Complete Test Suites

• Coping with high test strength requirements
  • A black-box test suite is complete with respect to a given fault model if and only if
    • Every conforming SUT passes all test cases
    • Every non-conforming SUT inside the fault domain fails at least one test case
Complete Test Suites

• How can we cope with the size of complete test suites?
  
  • Take generic parameters into account by using symbolic methods [3], [4]
  
  • Reduce test suites size by building equivalence classes [5]
  
  • Reduce test suite size further by enforcing completeness only for safety-related or mission-critical requirements [6]
Remaining Challenge.
Completeness & Consistency of Scenario Models

- Even if all scenarios have been tested by means of complete test suites:
  - **How do we ensure that the collection of scenario models is consistent and describes all relevant system behaviours?**

- New research field, involves
  - **Machine learning**
Autonomous Trains (Rolling Stock)
Autonomous Trains (Rolling Stock)

• Driving rolling stock trains without human train engine drivers has many advantages, in particular

• Freight trains can be “parked” anywhere to let passenger trains bound to fixed time tables pass, without having to consider rest periods for the train engine driver
Autonomous Trains V&V

• Why does V&V for autonomous trains require more effort than V&V for manual train control?
Consider First V&V for Conventional Train Control With Human Train Engine Driver

Train Control Computer

Fixed set of well-defined non-evolving behaviours

Train Engine Driver

Initially trained behaviour — continuously evolving, due to practical experience
Consider First V&V for Conventional Train Control With Human Train Engine Driver

V&V and Certification applies here

Train Control Computer

Fixed set of well-defined non-evolving behaviours

Train Engine Driver

Initially trained behaviour — continuously evolving, due to practical experience
Autonomous Trains V&V

Train Control Computer

- Fixed set of well-defined non-evolving behaviours
- Initially trained behaviour — continuously evolving, due to practical experience
Autonomous Trains V&V

V&V and Certification applies here

- Fixed set of well-defined non-evolving behaviours
- Initially trained behaviour — continuously evolving, due to practical experience
Consequences of High V&V Workload for Autonomous Trains

- A considerable portion of tests needs to be executed in the cloud, with very many tests running in parallel.

- To obtain certification credit for tests in the cloud, these tests need to run in an emulation environment that reflects the true HW target platform in a trustworthy way.
  - Again, we need emulators.
  - The research fields related to building trustworthy emulators are:
    - HW/SW Codesign
    - Virtual Prototypes [7]
Conclusion
Conclusion

• Cloud-based architecture for railway control systems has been presented
  • Based on the DS$^3$ system by Siemens Mobility GmbH

• V&V issues have been analysed

• Feasible modelling approach can be based on scenarios

• Test strategies with full fault coverage may be used to prove correct implementation of safety-relevant requirements with acceptable effort
Main Challenges for the Future

• Invent validation methods to check completeness and consistency of scenario collections – based on machine learning

• **Tool qualification for trustworthy emulators** (research field Virtual Prototypes [7])
  
  • Needed for
    
    • Execution of legacy IXL software in the cloud
    
    • Execution of trustworthy tests in the cloud
Further Reading

1. Sonja Steffens. Safety@COTS Multicore, Distributed Smart Safe System DS³. Siemens Mobility GmbH 2018, available under https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwizhzdvTdjiAhVQr6QKHU5QDQUQFjAAe5QIcBRAC&url=https%3A%2F%2Fsmartrail40.ch%2Fservice%2Fdownload.asp%3Fmem%3D0%26path%3D%2525Cdownload%2525Cdownloads%2525C2018%252011%252013%2520Innovationstr%2520ETCS%2520Stellwerk_smartrail%25204.0.pdf&usg=AOvVaw23cALWR65rwvLr7jpjvt11


3. Jan Peleska: Model-based avionic systems testing for the airbus family. ETS 2018: 1-10


7. Mehran Goli, Rolf Drechsler: Scalable Simulation-Based Verification of SystemC-Based Virtual Prototypes. DSD 2019: 522-529