RATP safety approach for railway signalling systems

ReSIST summer School 2007
Pierre CHARTIER

Summary

1. Introduction
2. Hardware fault detection
3. Software fault avoidance
Introduction

Global railway system

- Rolling stock
- Environment
- Signalling / Train control system
- Infrastructure
- Access / Evacuation
- Operation / Maintenance
- Station
- Track
- Power supply

Events to be feared at global system level

- Fire / explosion
- Derailment / overturning
- Panic
- Electrocution / burn
- Collision
- Individual accidents (fall, …)
- Others (terrorist attack, natural disaster, structure breaking, …)
**Introduction**

**Transport system overview (METEOR example)**

1. video in train
2. Inter-phone in train
3. video in platform
4. Inter-phone in platform
5. Platform screen doors
6. onboard equipment
7. transmission
8. track-vehicle communication
9. interlocking
10. trackside equipment
11. operation control center

**Railway signalling system**

Main protection against collision and derailment

- Safety critical mission

Historically 2 types of system

- Interlocking
- Automatic train protection

Main safety measure: stop all trains and power off the traction power supply
**Introduction**

**RATP technical evolution**

- Permanent ATP
- Metro
- 1970
- 1980
- 1990
- 2000
- 2010
- Manless interlocking
- CBTC
- OURAGAN
- Manless
- L1
- Fail-safe equipment
- Coded processor
- Safety-critical software
- B formal method
- SCADE
- Redundant Processors

---

**SATCEM – RER A**

Saturation of RER line A (80’s) → SATCEM project

- Objective:
  - To increase transport offer by raising train frequency

- But incompatibility between
  - train spacing reduction,
  - and traditional signaling
**SACEM – RER A**

Automatic Train Protection

› Control train spacing
› Control train speed
› Protect switching zones
› Switch between cab signal and trackside signalling

→ First safety-critical computing system in railways

---

**SACEM functions require the use of computers**

Two main concerns :

› Detection of errors due to hardware
  → coded processor
› Avoiding faults in software
  → formal methods
Hardware fault detection

Coded processor – main concepts

- Based on data and program encoding
- Encoding done automatically by specific tools
- Detect run-time errors
- If an error is detected, the hardware sets the system in a fail-safe state
Hardware fault detection

Coded processor onboard-wayside interface

Wayside
- Inputs encoding
- Encoded fixed data
- Transmitter
- Receiver
- Data processing
- Safety checked outputs

Onboard
- Inputs encoding
- Encoded fixed data
- Transmitter
- Receiver
- Data processing
- Safety checked outputs

Signalling and switching equipment

Train ahead

Fail safe power supply

Emergency braking

Cabsignal

Speed and distance detector

Transmitter

Receiver

Input encoding

Fail-safe clock

Output

Dynamic controller

Checking of date

Signature

Reference signature PROM

Microprocessor

Coded treatments

Dated signature

Information from the cabin
**Hardware fault detection**

**Coded processor – safety data encoding**

Data X = functional part X.F and coded part X.C

\[ X.F : N_F \text{ bits and X.C} : N_C \text{ bits} \]

Tasks for the computer

- Acquisition and coding of the fail-safe inputs
- Processing the coded data
- Conversion of coded data into fail-safe outputs
- Setting the system into restrictive state in case of failure

**Coded processor – detected errors (1)**

Differents kinds of errors:

- Arithmetical error
- Operator error
- Operand error
- Memory « non-refreshed » error
- Branch error
Coded processor – detected errors (2)

Components of the coded part

- Arithmetical error ==> remainder \( r_k \)
- Operator error ==> signature \( B_x \)
- Operand error ==> signature \( B_x \)
- memory « non refreshed » error ==> date D
- Branch error ==> compensation, tracer

Coded processor – architecture

Hardware fault detection

Energy supply for Vital outputs

Fail-safe Inputs

Fail-safe Outputs

Outputs checking

CES : Inputs encoding

CUC : Central Processing Unit (coded)

CSS : Outputs Conversion

CKD : contrôle

Safety Clock
**Hardware fault detection**

**Coded processor – Signature predetermination tool (OPS)**

The OPS protects from the compiler failures, therefore the compiler does not require specific qualification.

**Hardware fault detection**

**Coded processor – safety outputs**

Setting of safety outputs

- rereading of outputs
**Hardware fault detection**

*Coded processor – safety integrity level*

Theoretical result:

- Coded processor alone $\rightarrow 10^{-12} \text{ h}^{-1}$
- Including transmission between onboard and trackside equipment $\rightarrow 10^{-9} \text{ h}^{-1}$

**Software fault avoidance**
Software fault avoidance

**ATP role**

![ATP role diagram](image)

- ATP’s command of emergency braking
- Speed limitation
- Emergency braking
- Cab signalling
- Movement authority limit

**Communication-based train control (CBTC) systems**

- Power supply
- Interlocking equipment
- Platform equipment
- Operation control center
- I/O interface
- Zone controller section A
- Zone controller section B
- Maintenance system
- Data communication system (LAN, radio, inductive-loop...)
- Localisation system
- Onboard equipment
- Cab signal
- Rolling stock
Software fault avoidance

**CBTC operation**

- train B movement authority limit
- speed profile
- zone controller
- train A position
- virtual block
- track circuit

**Communication-based train control (CBTC) systems**

- Automatic Train Protection (ATP)
- Automatic Train Operation (ATO)
- Automatic Train Supervision (ATS)
1988 SACEM - First safety software in railways
   ‣ Usual (unformal) software specification issues
      – lack of global approach with the system designer point of view
      – ambiguous, not legible, not coherent, not complete
   ‣ Validation issues
      – no certitude that the functional tests are sufficient
1998 First run of the subway line 14 Météor
The B method is used to obtain:
   ‣ a reliable and exact software design from specifications to runtime code

**Formal methods**

**B formal method**

**Goal**
   ‣ To get a software which meets completely its functional specification by construction

**Application fields**
   ‣ Sequential code with no interruptions (real time aspects, low level softwares, operating kernels are not taken into account)

**Large spectrum language**
   ‣ Unified framework and continuous process from specification to code
B formal method

High level language
  ‣ Abstract operators for specification needs
  ‣ Concrete instructions similar to ADA or C one’s
Model oriented approach
  ‣ Software = data + properties + operations
Refinement process
  ‣ Translation of the abstract machines into concrete modules, and finally into code
Proof obligations
  ‣ Conditions to check to ensure a strict mathematical construction of softwares

Software fault avoidance

B formal method – examples of safety properties

  ‣ Only equipped train which is located and in automatic mode can have a target.
  ‣ The trains locations computed by the SWE must be correct with the actual trains locations on the line.
**Software fault avoidance**

**B development process**

1. Unformal Requirements
2. **B formal specification**
3. **B refinement 1**
4. **B refinement n**
5. **B implementation**
6. Code

- Formal re-expression
- PROOF
- PROOF
- PROOF
- PROOF
- Automatic and Manual code translation

**B verification process**

1. Unformal Requirements
2. **B formal specification**
3. **B refinement 1**
4. **B refinement n**
5. **B implementation**
6. Code

- Functionnal tests
- Integration tests
- Module tests

- B proof obligations = exhaustive testing
Software fault avoidance

B validation process

Unformal Requirements

Formal re-expression

All requirements are traced within the B model

Proof

Proof activities are checked

Only for Manual translation

Code is exhaustively compared with B implementation

B formal specification

B refinement 1

B refinement n

B implementation

Software fault avoidance

B industrialisation

AtelierB©: An industrial tool to specify, refine, implement and prove B models

Statistics about Météor B model

- 1150 B components
- 115 000 lines of B code
- 27 800 Proof Obligations (all proved)
- 86 000 lines of « safe » ADA code
**Software fault avoidance**

**B today in railway industry**

Used by two railway leaders: SIEMENS and ALSTOM

Recent projects:

- Canarsie Line (New-York),
- North East Line (Singapour)

Projects size has increased more than twofold

---

**Interlocking system**

Track circuit

Route V1 to V2Q
Signals M (V2), H, KR, C
Switches 101, 201
Signal M (V1)
**Software fault avoidance**

**Relay interlocking**

Main technology on RATP network
- Fail-safe relays
- Man-machine interface with button/switch control panel

**Increasing cost and expensive reconfiguration**

**Electronic interlocking**

- Configured interlocking graphs
- Graph interpreter real-time engine
- Electronic interlocking
- Wayside equipments
- Generic signalling rules
- Site configuration
- Operator
- Off-line
**Interlocking validation**

Issue: how to be convinced that any combination of generic graphs for any site configuration is safe?

- Heavy testing for both supplier and RATP on site configuration

To reduce test effort for next interlocking sites, formal proof of safety properties has been considered.

---

**Interlocking proof**

- Derailment
- Collision
- Generic feared events

17 refined properties
- Site configuration
- Interlocking graphs

Configured properties
- Configured interlocking graphs

Proof engine

Validation
Software fault avoidance

**Interlocking proof process (1)**

1. **Translator 1**
   - Configured graphs
   - TECLA
   - Configuration + properties

2. **Translator 2**
   - Configured graphs
   - TECLA
   - Configuration + properties

---

**Interlocking proof process (2)**

1. **Proof engine**
   - Proof OK/KO
   - ProofLog
   - Properties OK/KO
   - Proof OK/KO

2. **Proof engine**
   - Proof OK/KO
   - ProofLog
   - Equivalence system
   - Proof OK/KO

Translation certification

Proof certification
**Software fault avoidance**

**Interlocking proof**

The proof engine (from Prover Technology) is based on combination of SAT techniques and other automatic proof techniques.

Work in progress

- Feasibility is established
- Complete proof of a real interlocking configuration is expected in a few months

**Apparition of SCADE tools in railway industry**

For a few years, SCADE has found favour with railway industry

- fitted for designing command-control systems
- reduces development cost
- facilitates communication between specialist engineers and software engineers
Software fault avoidance

SCADE brief overview

‣ based on a declarative synchronous language Lustre, encapsulated in graphical representation
‣ Software = variables + equations
‣ Time is discrete \((\text{var}_n)_N\)
clocks, temporal operators (pre, when, …)
‣ Equations between inputs and outputs
\[\text{out}_n = \Phi(\text{in}_n, \ldots, \text{in}_{n-p}, \text{var}_n, \ldots, \text{var}_{n-q})\]

Software fault avoidance

SCADE proof process (1)
Software fault avoidance

SCADE proof process (2)

Example of safety property:

- Two distinct trains must not cross their movement authority limit

Work in progress

- Feasibility on a real site configuration
- System requirements specification coverage
- Method to complete proof when safety properties are not totally proved
Software fault avoidance

**Formal methods ...**

- reduce drastically test effort
- provide a high level of quality and safety for software
- are applicable to industrial software projects
- but have to take more into account the practical aspects for using them (cost, competence, …)

---

**RATP renewal program: software development methods**

<table>
<thead>
<tr>
<th></th>
<th>OURAGAN CBTC</th>
<th>Manless CBTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>B method</td>
<td><img src="L3" alt="B method OURAGAN CBTC" /></td>
<td><img src="L1" alt="B method Manless CBTC" /></td>
</tr>
<tr>
<td>Coded processor</td>
<td>![Coded processor OURAGAN CBTC](L5 WaySide Equipement)</td>
<td><img src="L1" alt="Coded processor Manless CBTC" /></td>
</tr>
<tr>
<td>SCADE</td>
<td><img src="L5" alt="SCADE OURAGAN CBTC" /></td>
<td><img src="L3" alt="SCADE Manless CBTC" /></td>
</tr>
<tr>
<td>Redundant processors</td>
<td><img src="L5" alt="Redundant processors OURAGAN CBTC" /></td>
<td><img src="L13" alt="Redundant processors Manless CBTC" /></td>
</tr>
</tbody>
</table>