CLEARSY

Safety Solutions Designer

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

FONDATION SCIENCES MATHÉMATIQUES DE PARIS

HORIZON MATHS 2024

PREUVE MATHÉMATIQUE ET SÛRETÉ LOGICIELLE

Formal Methods for Safety Critical Systems

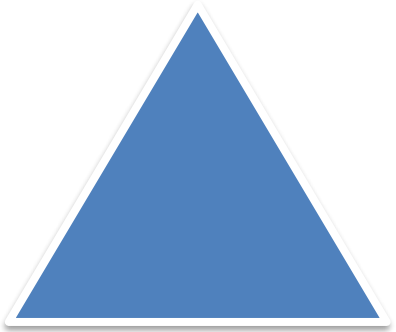
*« This presentation looks at the use of formal methods to ensure the safety and security of critical infrastructures. It explains **why formal methods are used**, through the prism of industrial activities and the experience feedback collected, particularly accidents that occur. »*

Thierry Lecomte
R&D Director

THIERRY.LECOMTE@CLEARSY.COM

Agenda

SAFETY

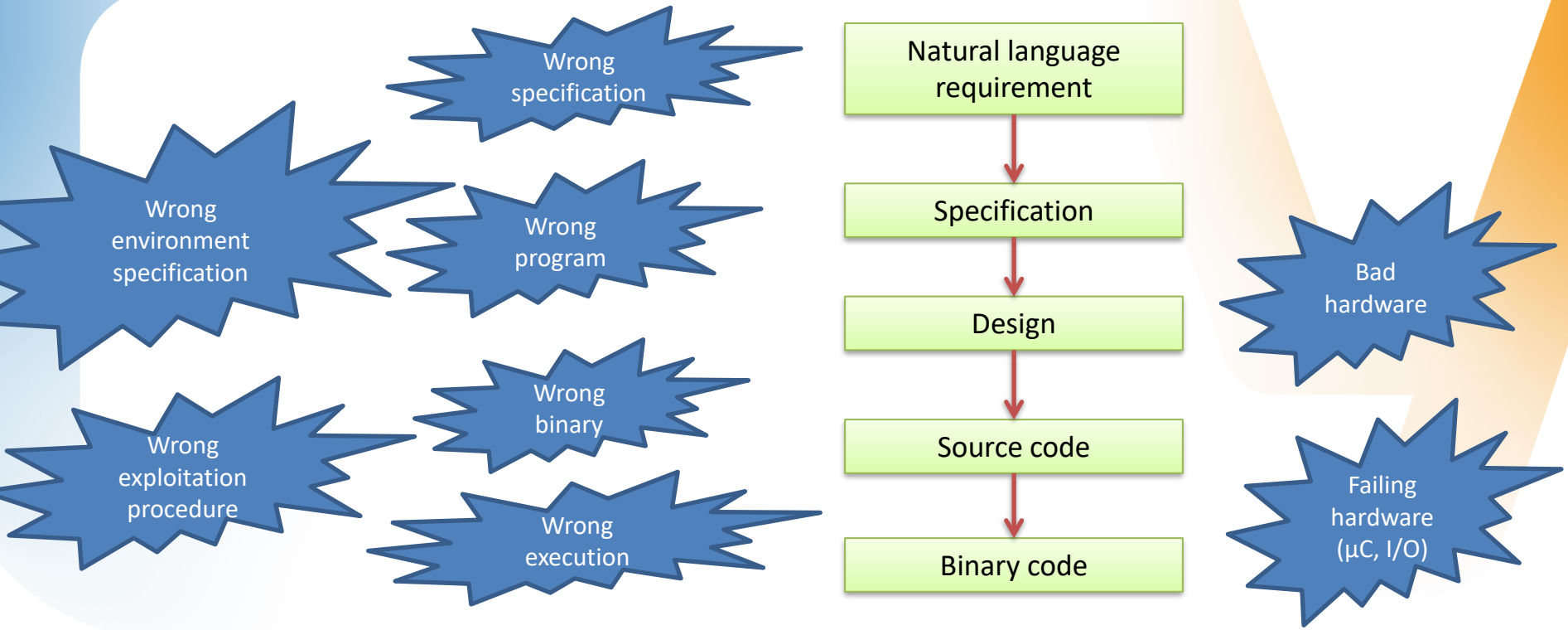


FORMAL METHODS

(MOSTLY) RAILWAY APPLICATIONS

EXPLAIN THE « WHY »

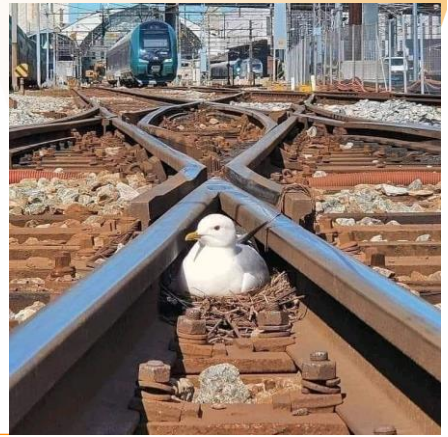
Safety is About Failing System



Standards for Safety Critical Systems

- When risk of injury / death
- Domain-specific standards
- Safety Integrity Level (*SIL*)
  - Level 3: 1 catastrophic failure every 100 years (10^{-7}/h)
  - Level 4: 1 catastrophic failure every 10 000 years (10^{-9}/h)
- FM highly recommended for SIL3-SIL4 (railways)
- FM mandatory for EAL6+/7 Common Criteria (Security)
- Recommandations (REX, best practices)
  - No definitive recipe to produce safe systems
  - Cover SW, HW and development process
  - Quality & correct development required
- **Safety by-design !** → Diversity, redundancy, etc.

- FDA: healthcare
- 26262: automotive
- EN5012{6,8,9}: railways
- IEC61508: industry
- DO178: aeronautics



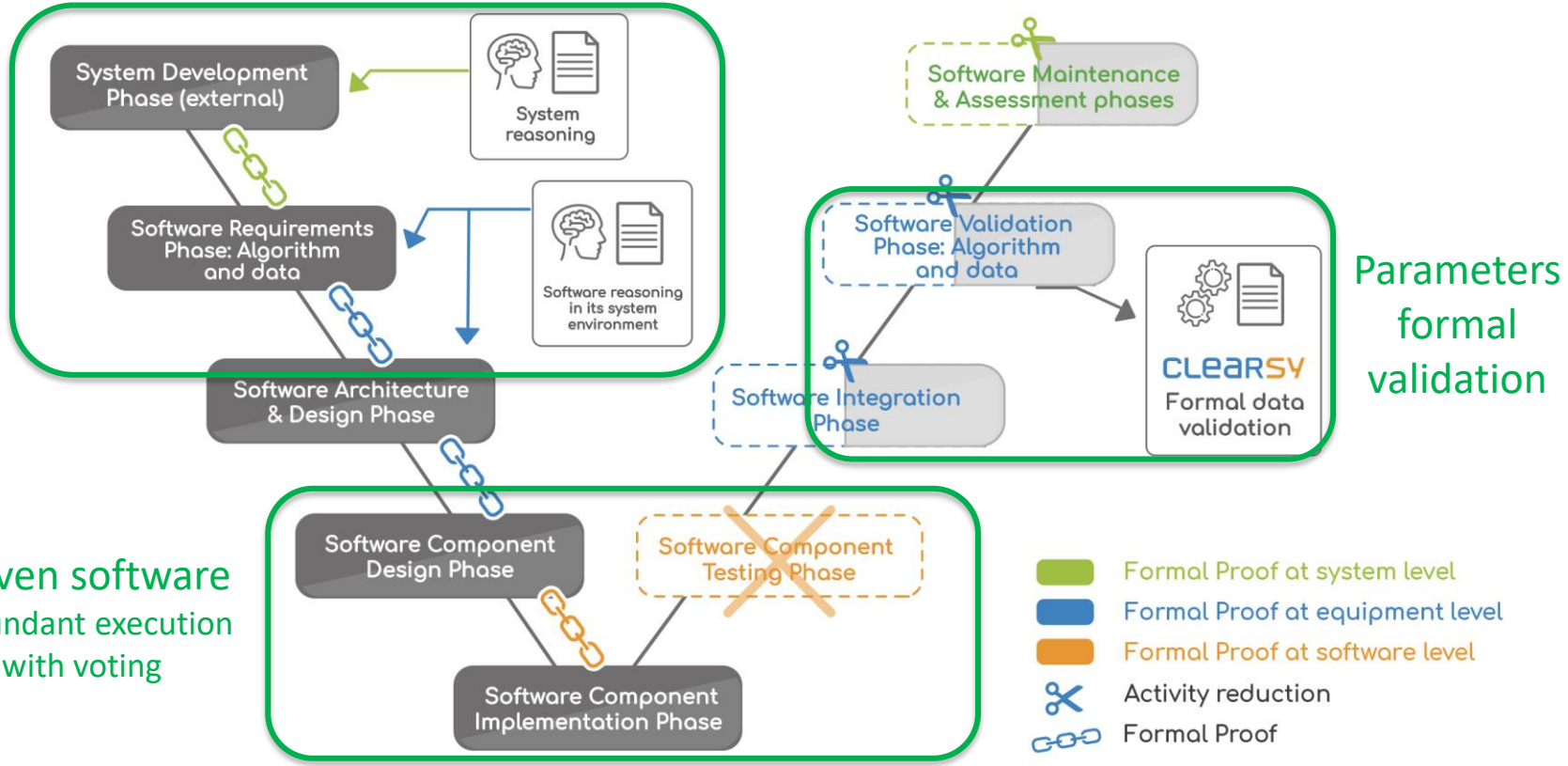
Demonstration Required

- Safety demonstration
  - Natural language document
  - Safety analysis (feared events)
  - How technical system is going to be safe, based on hypotheses and V&V elements

    Could be software code, electronic schematics, models, etc.
- Convince human expert that is not going to fail more frequently than expected

Formal activities throught the V cycle

Explain why it is designed this way

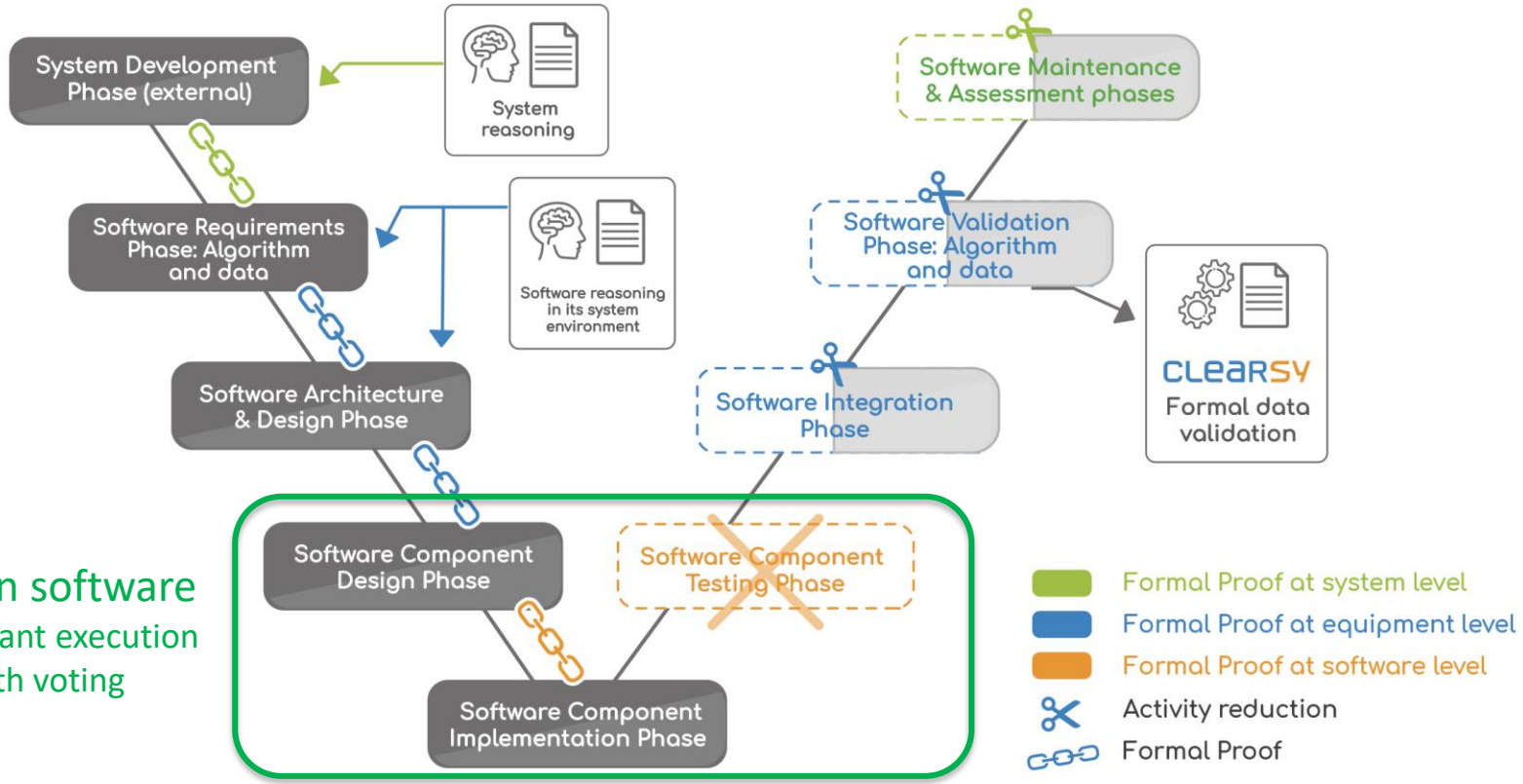


Parameters formal validation

Proven software

Redundant execution with voting

SOFTWARE DEVELOPMENT WITH B



Proven software

Redundant execution with voting

« Only inactive sequences can be added to the active sequences execution queue. »

```
activation_sequence = /* Activation d'une séquence non active */
PRE ¬(sequences = sequences_actives) THEN
    ANY sequ WHERE
        sequ ∈ sequences - sequences_actives
    THEN
        sequences_actives := sequences_actives ∪ {sequ}
    END
END;
```

```
activation_sequence = /* Activation d'une séquence non active */
VAR sequ IN
    sequ <-- indexSequenceInactive;
    activeSequence(sequ)
END;
```

```
void M0__activation_sequence(void)
{
    CTX__SEQUENCES sequ;

    sequence_manager__indexSequenceInactive(&sequ);
    sequence_manager__activeSequence(sequ);
}
```

```
0x01F970 FFFF 8B4C 2440 89C5 8D7D 0C8B 4110 89CE
0x01F980 83C6 0C8D 1485 0000 0000 8D42 0883 F807
0x01F990 7617 F7C7 0400 0000 740F 8B41 0C8D 7D10
0x01F9A0 83C6 0489 450C 8D42 04FC 89C1 C1E9 02F3
```

Natural language requirement

B Specification

Behaviour + properties

B Implementation

Behaviour + properties

C generated code

Binary code

« Only inactive sequences can be added to the active sequences execution queue. »

Natural language requirement

```
activation_sequence = /* Activation d'une séquence non active */
PRE ¬(sequences = sequences_actives) THEN
    ANY sequ WHERE
        sequ ∈ sequences - sequences_actives
    THEN
        sequences_actives := sequences_actives ∪ {sequ}
    END
END;
```

Cyclic software single-thread

```
activation_sequence = /* Activation d'une séquence non active */
VAR sequ IN
    sequ <-- indexSequenceInactive;
    activeSequence(sequ)
END;
```

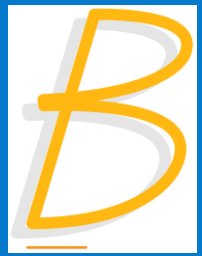
B Specification

Proof (coherence)

Proof (refinement)

B Implementation

Proof (coherence)



```
void M0__activation_sequence(void)
{
    CTX__SEQUENCES sequ;

    sequence_manager__indexSequenceInactive(&sequ);
    sequence_manager__activeSequence(sequ);
}
```

C generated code

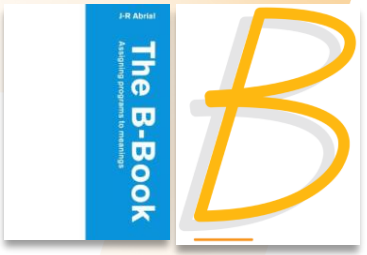
```
0x01F970 FFFF 8B4C 2440 89C5 8D7D 0C8B 4110 89CE
0x01F980 83C6 0C8D 1485 0000 0000 8D42 0883 F807
0x01F990 7617 F7C7 0400 0000 740F 8B41 0C8D 7D10
0x01F9A0 83C6 0489 450C 8D42 04FC 89C1 C1E9 02F3
```

Binary code

Software Formal Development

Atelier B IDE

- 198x: [initiator event] TCMS under development, late and with safety errors
- Atelier B initially developed by Alstom
- 1993-1998: Improved for Paris L14 Automatic Train Protection (ATP)
  - Emergency braking in case of danger (86 kloc B, 110 kloc Ada)
- 1998: V3.x certified
- 2001: Free tool (dual licence: community or with tight support)
- Used by ~30% radio-based control metro worldwide
- Used for Paris L1, L4, L13
- 2024: V2024.6 to be certified T2 EN50128 (code generation not included)
- To be used for Paris L15, L16, L17, L18
- 2001-2024: Also used marginally for smartcard certification up to EAL6+



References:

- *The B-book - Assigning Programs to Meanings*, Cambridge Press, 2001
- *The First Twenty-Five Years of Industrial Use of the B-Method*, FMICS, 2020

Software Formal Development

Atelier B Technology [C, C++, Prolog-like]

Automatic refinement based on Siemens inference engine

2006

2006-2024

- Integrated into Atelier B
- Applications up to 500 kloc for train control (NY metro, CdG shuttle) and software engineering (interpreter, compiler)

Code generators:

2001-2024

2023

- Ada (proprietary)(product specific)
- C (generic, 32-bit MCU)(generation of Frama-C ACSL)
- Rust
- RIP: Instruction List, Ladder, LLVM, VHDL

```
RULE scalar_ini0
REFINES
    @a :: @b
WHEN
    ENUM(@b) &
    @c : @b
IMPLEMENTATION
    @a := @c
END;
```


References:

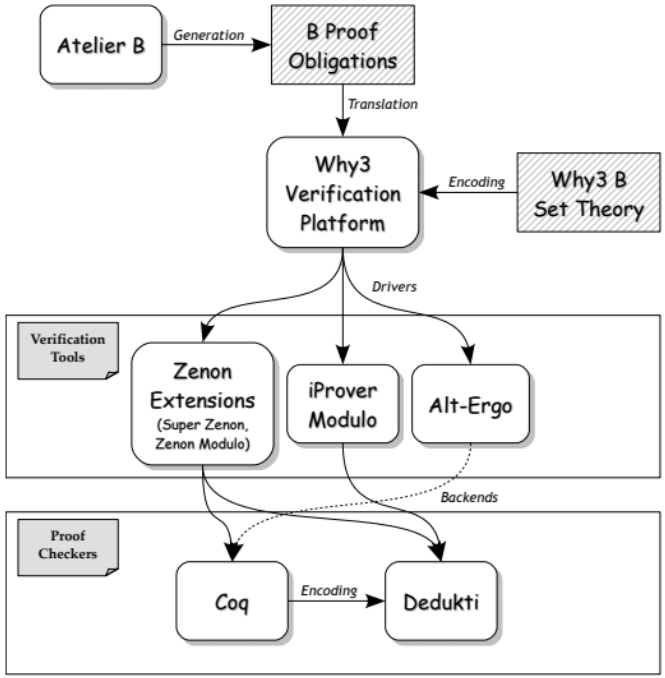
- *Applying a Formal Method in Industry: A 15-Year Trajectory*, FMICS, 2009
- *On B and Event-B: Principles, Success and Challenges*, ABZ, 2018
- *B2rust*, https://github.com/CLEARSY/b2rust

Software Formal Development

Atelier B Technology [C, C++, Prolog-like]

Specific proof tools developed

- **1998**
  - Main prover as an inference engine with using 2600 rules
  - Predicate prover to demonstrate 80% of the rules
  - Main prover stuck in 1998 (interactive demos could not survive prover improvement)
- **1998-2024**
  - Extension of interactive proof language, GUI
- **2008-2027**
  - Connexion with third party provers (Alt-Ergo, CVC3, iProver, Vampire, Z3, Zenon)
  - 500k proof obligations publicly available for benchmark
- **2022-2024**
  - Connexion with Generative AI for proof script generation



The BWare Platform for the Automated Verification of B Proof Obligations

References:

- ANR Projects Bware, BLASST, ICSPA - ECSEL Project AIDOaRT
- *Atelier B oPEn ResOurces*, https://github.com/CLEARSY/apero

Software Formal Development

- Atelier B Exploitation
  - No fatality in 25 years
  - Continuous improvement since 1998
  - Applications to automatic / autonomous mobility at large
  - Avoid 2x independent teams to develop SIL4 software
- Atelier B Dissemination
  - Continuous low frequency professional training
  - Internal training for volunteers and FM profiles
  - Continuous academic courses with CLEARSY Safety Platform
  - Downloads:
    - **4500** / teaching semester,
    - 1300 Atelier B Prover plug-in for Rodin platform

References:

- *Programming Handbook*, https://github.com/CLEARSY/CSSP-Programming-Handbook

CLEARSY Safety Platform [B]

Developing SIL3-SIL4 Functions

- [initiator event] Passengers pushed on Paris metro tracks. CLEARSY develop a safe platform screen-door controller

2005
- Demonstrator based on PLCs installed in L13

2006-2012
- System exported to America and Europe, wheel reinvented several times, system copied by competitors
- [Note] Safe computer usually are packaged with a train or as a LEGO where the safety has to be constructed

2016-2019
- LCHIP (project funded by BPI) to develop a generic safe computer with 2x 32-bit MCUs and programmed with B
- 4 instances of the same function with diverse compilation
- LCHIP experimented 3 years in worldwide universities

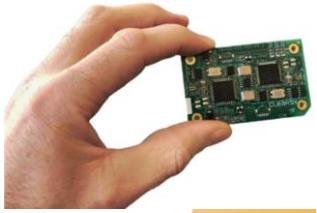
References:

- *The CLEARSY safety platform: 5 years of research, development and deployment*, SBMF, 2020

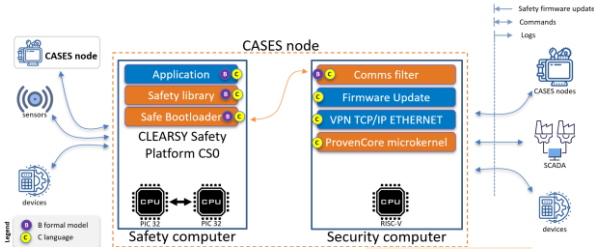
CLEARSY Safety Platform [B]

Developing SIL3-SIL4 Functions

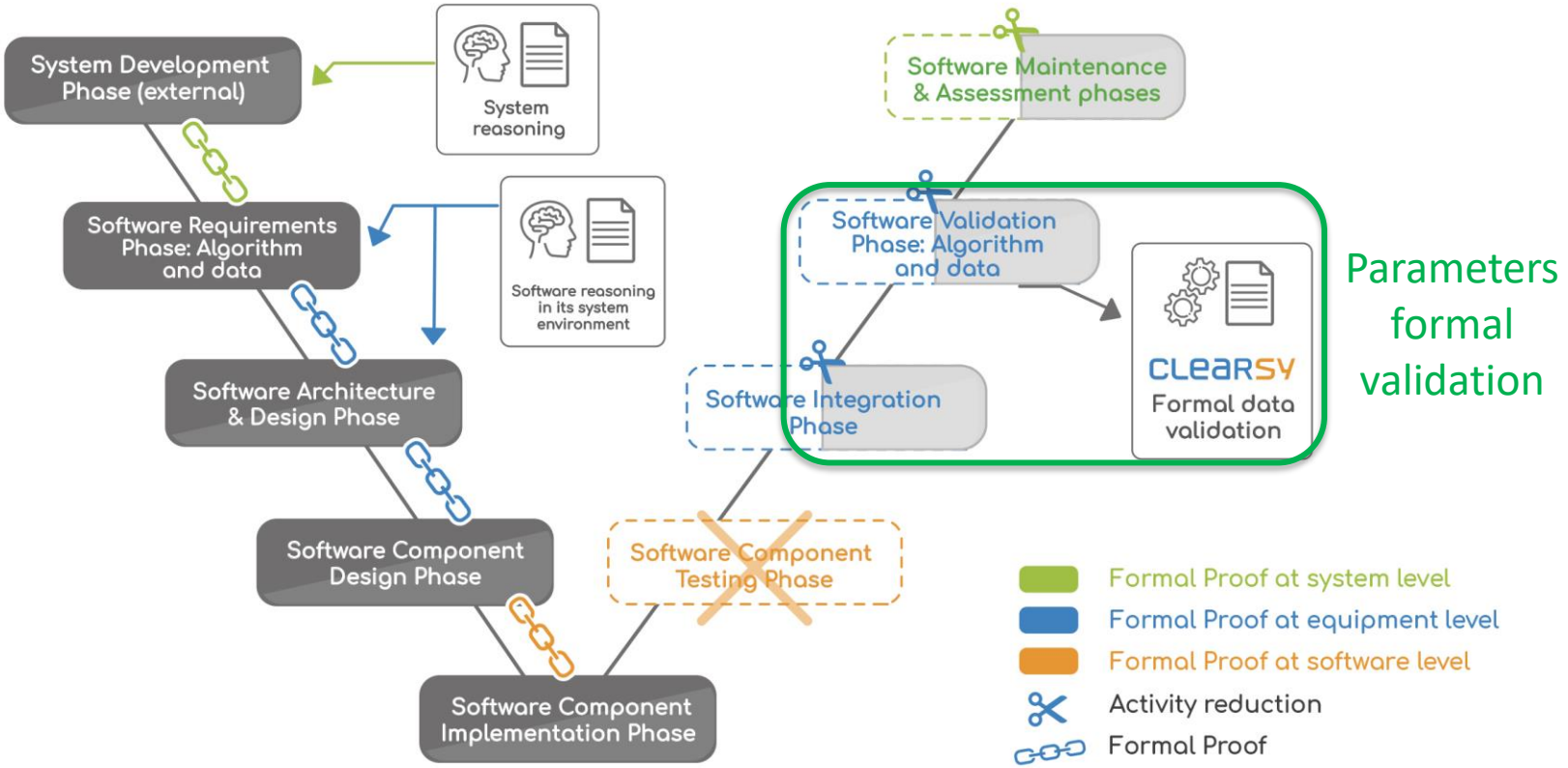
- CLEARSY Safety Platform developed in 1 year
- 2021 — SIL4 platform certificate obtained
- 2023 — Deployment in Australia (Brisbane) with some cybersecurity
- 2021-2025 — CASES (project funded by BPI) to develop a safe and secure computer.
- Applications for
  - Autonomous train (x3)(localisation)
  - Autonomous shuttle (low-level safety)
  - Communication (Thales)
  - Decision making (Naval Group)
  - Remote controlled mobile robot



Generic ECU for Safety Critical C&C App



DATA VALIDATION WITH B MATH LANGUAGE



Parameters formal validation

Parameter Formal Validation [B math language]

Need to Avoid Boring Activities for Safety

- Data validation to manually check 10k-100k trackside safety-related values changing every day, against 1k-2k rules

199x

- [initiator event] Hard-coded C++ software to check embedded constant parameters difficult to maintain and adapt to other lines

2003-2024

- Development of several tools using B mathematical language to model data and to check compliance with model-checkers

2006-2008

- [confirmation event] Industrial dataset, supposed to contain 3 errors, fully verified by ProB model-checker, revealed 4 errors

2019

- [confirmation event] TGV overspeed over a switch in La Milesse due to errors not detected during human data validation

References:

- *Formally Checking Large Data Sets in the Railways*, ICFEM, 2012
- *ProB*, https://prob.hhu.de/

Parameter Formal Validation

▶ BEA-TT supports FM

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

BEA-TT
Bureau d'enquêtes sur les accidents de transport terrestre

2021

"Given the difficulty of controlling the growing quantity of parameter data, the use of validation algorithms is essential. The use of innovative formal methods, based on advanced mathematical concepts, is one answer."

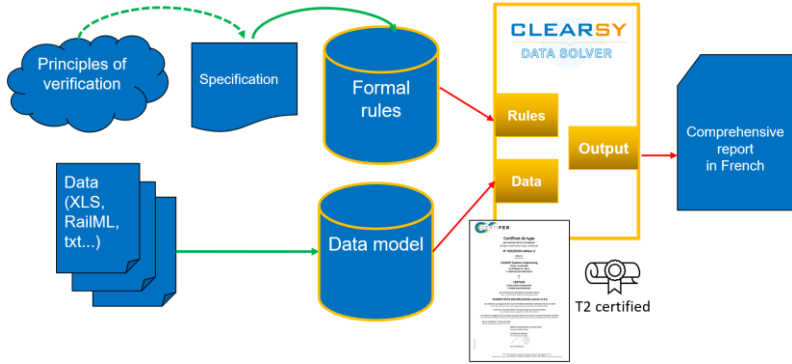
▶ Formal IDE and Services

2006-2024 ▷ Modelling and Verification tools adapted and integrated to industrial development cycles

2019 ▷ Certified T2 EN50128

2006-2024 ▷ Continuous support of ProB academic team

2023 ▷ Biggest B machine analysed: 10 Mloc

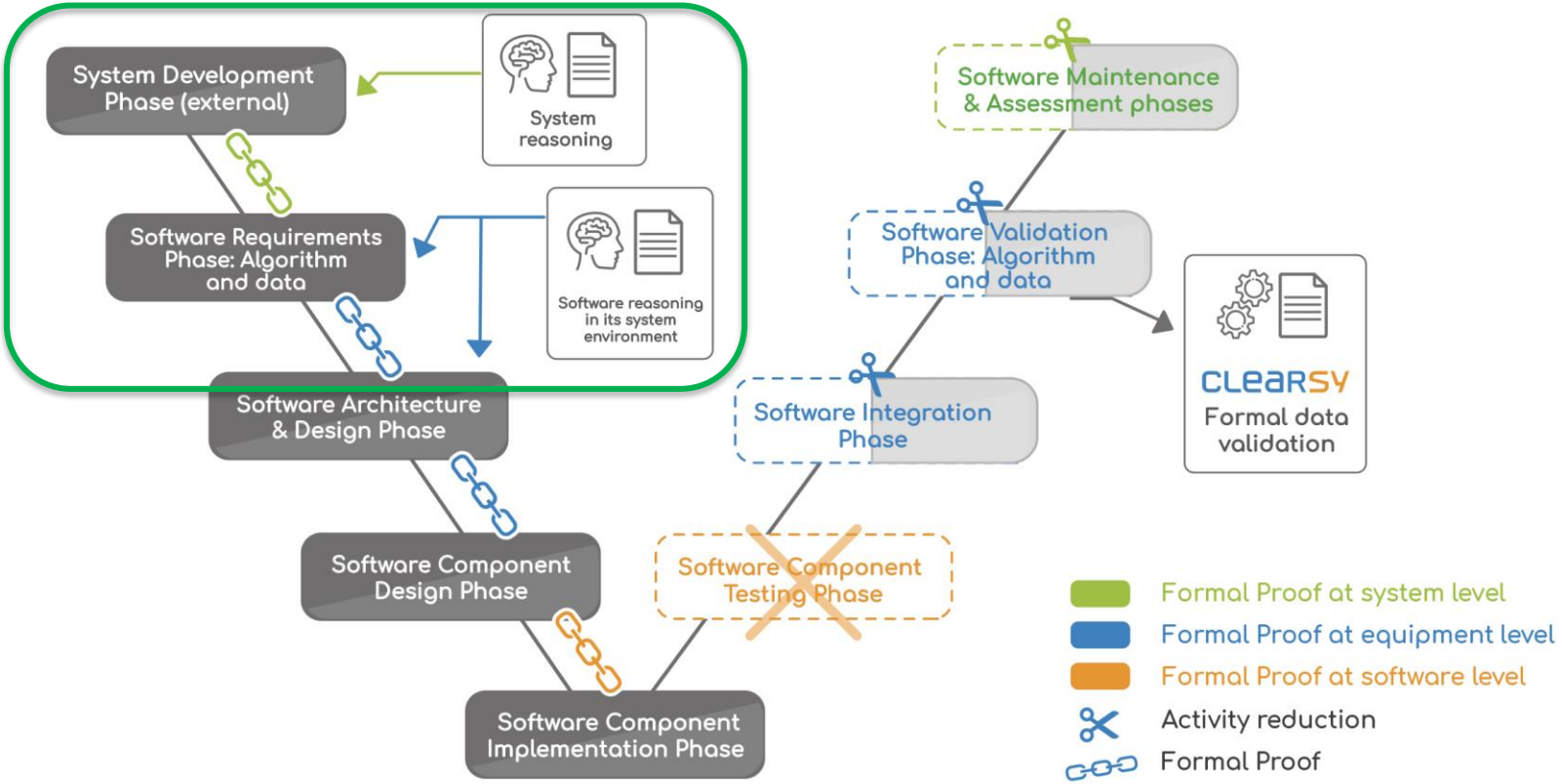


References:

- *https://www.bea-tt.developpement-durable.gouv.fr/rapport-d-enquete-sur-la-survitesse-d-un-tgv-le-22-a1077.html*
- *ProB*, https://prob.hhu.de/

SYSTEM PROOF WITH EVENT-B

Explain why it is designed this way



System Level Safety Proof [Event-B]

System safety assessment

- Legacy systems to evolve over time
- Existing non satisfactory structural modelling

2010-2012

- [initiator event] 2-year modelling of the safety reasoning for MTA network
  - Simpler models
  - 200-page self-sufficient natural language report
- Applied to railway systems worldwide, including Paris L3, L5, L9, L6, L11

2012-2023

- [confirmation event] **several errors detected** on follow-up projects on operated networks

2023-2025

- ERTMS Hybrid 3 first implementation in Marseille-Vintimiglia line. FM mandatory
- Connexion with formal data validation (exported constraints).

References:

- *Formal Proofs for the NYCT Line 7 (Flushing) Modernization Project*, ABZ, 2012
- *Safety Analysis of a CBTC System: A Rigorous Approach with Event-B*, RSSR, 2017

Conclusion

- Why do we use formal methods ? (i.e. B, Event-B, ProB, Frama-C)
  - We are more efficient, more competitive, more flexible
  - Enhance the safety demonstration (clarity, test vs proof)
  - Help us to keep things under control
  - We find problems on existing systems / never implemented specs
- What perspective ?
  - Problem not yet « solved »: incidents, accidents still happen
  - FM requirement appears in call for tender
  - Applied also in non-safety related domains
  - Room for improvement, contribution to SotA
  - Human central, ability to handle application domains “teachable” ?

CLEARSY
Safety Solutions Designer

FONDATION SCIENCES MATHÉMATIQUES DE PARIS

HORIZON MATHS 2024

PREUVE MATHÉMATIQUE ET SÛRETÉ LOGICIELLE

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

Thank you
For your attention

MOOC
massive open online course

THIERRY.LECOMTE@CLEARSY.COM

https://mooc.imd.ufrn.br/