Safety Demonstration and Software Development

Jean-Claude Laprie

10.4 52nd Meeting - Uphall - 28 June-2 July

Workshop on Achieving and Assessing Safety with Computing Systems: State of the Art and Challenges

Study performed for RATP (Régie Autonome des Transports Parisiens), the utility organisation for public transportation in Paris and region

Context: questionning on current software development approach, mathematically formal development by B-method

> Is it possible to demonstrate the same safety level without resorting to mathematically formal methods for developing safety-critical software?

- Keyword: demonstrate
- Attitude: no a priori, together with
  - Experience in fault tolerance rather than in formal approaches
  - Knowledge of current system approach (validation of the coded processor for speed control of SACEM A-line of Paris regional trains, PADRE protocol for consistency in bi-processor architectures for sidetrack equipments)

- Reminding the infeasability of quantifying reliability of safety-critical software

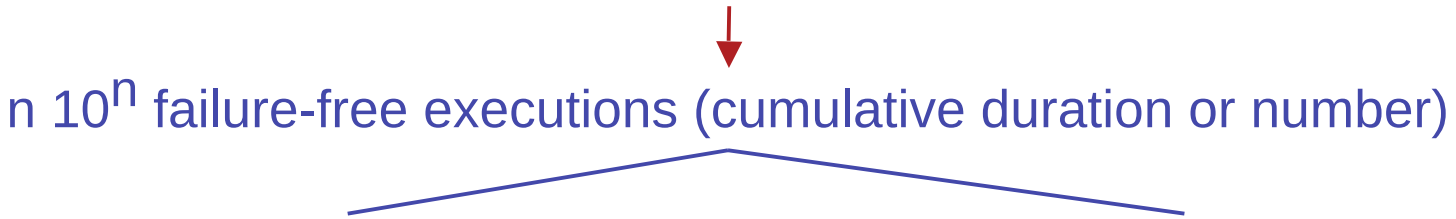
Development process for software reliability

System vision for safety demonstration

- Situating the current RATP approach
- Examining alternate approaches for safety-critical software development
- Coming back to RATP approach
  - Underlying assumptions: specification correctness, static data
  - Structure of development process
- Concluding recommendation: pursue mathematically formal development

Infeasability of quantifying reliability of safety-critical software

- Experimental demonstration of 10^{-n} failure rate (/h ou /demand)



$n\ 10^{n}$ failure-free executions (cumulative duration or number)

Validation

Failure-free tests
+
Input domain coverage

Operational life

$m\ 10^{m}$ failure-free exécutions, m<n

Failure rate ≤ 10^{-m}
Actual failure rate?

- Predictive evaluation
  - System from components
  - Reliability growth models
  - Incorporation of past experience

→ Inappropriate for safety-critical software

Situation of the current RATP approach

Civil avionics

Diversity, or dissimilarity, at several levels

- Hardware
- Software
- Control surfaces

Nuclear protection

In-depth defenses

- Differing technology barriers
- Delay before reaction

Railway signalling

[SIL 4 – Safety Integrity Level – of CENELEC EN 50126 standard]

End-to-end control

- Itinerary control
  - Safety net
- Speed control
  - Coded processor

Coded processor

- Execution flow signature embedding arithmetic code, computation cycle datation → quantifiable safety via probability of undetected error
- Assumptions
  - Absence of signature aliasing → Code length and key
  - Independence of failures of processor and of signature controler
  - Correctness of application software
    - Specification correctness
    - Absence of faults created during software development

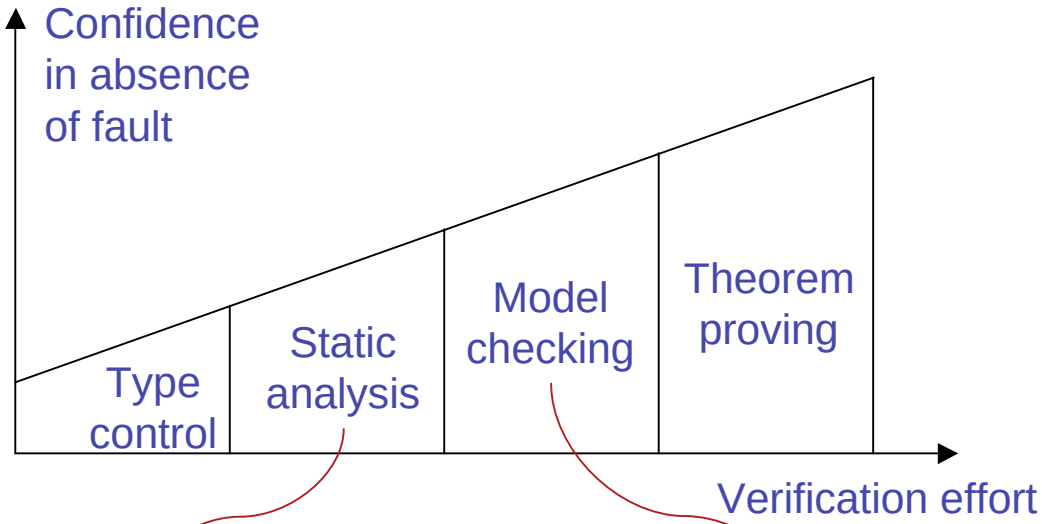
Mathematically formal development: fault prevention based on calculi in mathematical logic together with theorem proving

- Consistency checking (invariant preservation)
- Refinement checking (correct refinement)

- Static data correctness

Alternate approaches for safety-critical software development

- Formal verification



Complements, or replaces, testing, esp. unit testing

Unavoidable simplifications of model and properties

Synchronous languages

❖ Software diversity

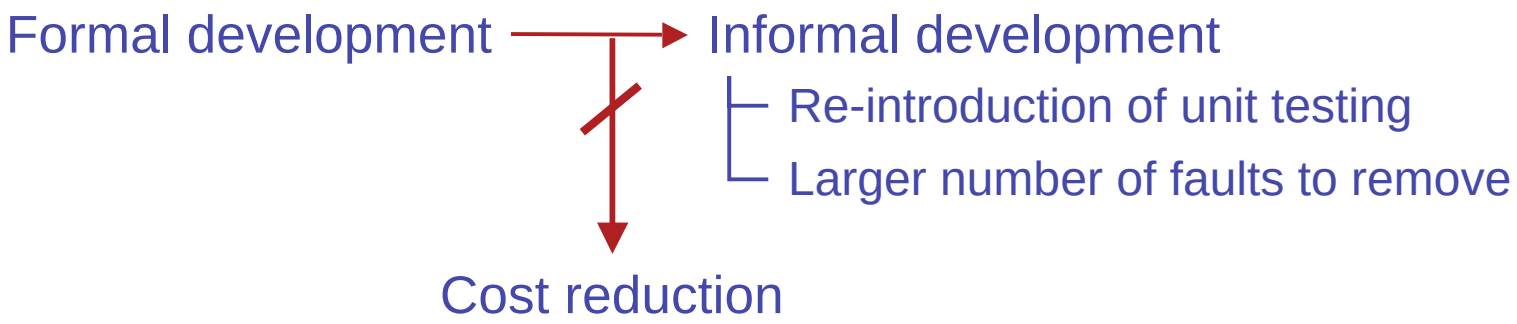
- ☞ Assumptions
  - (Specification correctness)
  - (Static data correctness)
  - Failure independence of hardware support to execution
    - ↓
    - Hardware diversity
  - Absence of common-mode residual fault → Version or variant failure independence
    - ↓ (lightning)
    - Not demonstrable at required level (although reliability improvement exhibited by all experiments and by operational data)
- ☞ Loss of end-to-end control
- ☞ Availability penalty

Flashback on RATP approach

- Specification correctness
  - Initial step informal in essence
    - ‘Semi-formal’ methods
      - Currently : SADT et ASA+
      - Explored : UML
        - Formalisation?
        - Link UML ➠ B
  - Safety specification
    - Check-lists
    - Models and simulations
      - On-going formalisations

- Static data correctness
  - Static data describe environment (track and station topology, location of sidetrack equipments)
  - Basic data, from which invariants are generated, in addition to some computation data
  - Achilles tendon of any control system, as
    - ✓ basic data can only be validated by reviews and inspections
    - ✓ Memory size for static data may, and usually does exceed memory size for programmes

- Structure of development process



Conclusion : recommendation to pursue mathematically formal development together with coded processor

- System recommendation
- Quantifiable safety demonstration based on assumptions weaker than other foreseen (foreseable?) approaches
- Adequacy of B method to RATP applications
  - Other applications
  - Past and current evolutions
  - Difficulties
    - ✓ Necessary mathematical culture
    - ✓ Tool limitations
- ☞ Another, independent, study reached same conclusion
- ☞ Contract for automating subway line #1 (most busy line of Paris subway) awarded to industrial proposal offering mathematically formal software development