# SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

*A proposition for Open ETCS*

**Dr. Marc ANTONI**

THE SNCF - INFRASTRUCTURE | ASSET MANAGEMENT & MAINTENANCE ENGINEERING

# SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

**Summary**:

1. **Problems to solve: safety and durability**
2. **Formal Method & software development**
3. **Interpretable requirement**
4. **Formal validation method**
5. **Tools developed by SNCF**
6. **Examples of use**

# SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

## FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

### Problem to solve

**The initial project** was to provide SNCF Infrastructure Manager and Rolling stock undertaker with an operating method guarantying:
→ the safety level, including the integration in the existing railway system
→ the easy refurbishment of the computerized safety relevant system (portability of the functional software to an over computerized target platform)
→ the taking under control of the operation costs

**Principle:**
proof of the functional requirement and industrial interpretation of the requirements
→ the proof covers equally the requirement and its real software implementation.

**Motivation for formal method**:
→**reduce costs** of all the life cycle (testing procedures, modifications…)
→**increase the safety level** (jump from "means" obligations to "results" obligation

## FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

### Problem to solve

**The traditional** development of computerized systems (critical or not) do not distinguish:

→ software relative to the “applicative rules” (requirement ⇔ functional)

➔ are the same for all the suppliers answering to a call to tender

→ software relating to the management of the material platform

➔ different for each supplier in regard of his hardware

**Without special precaution**:

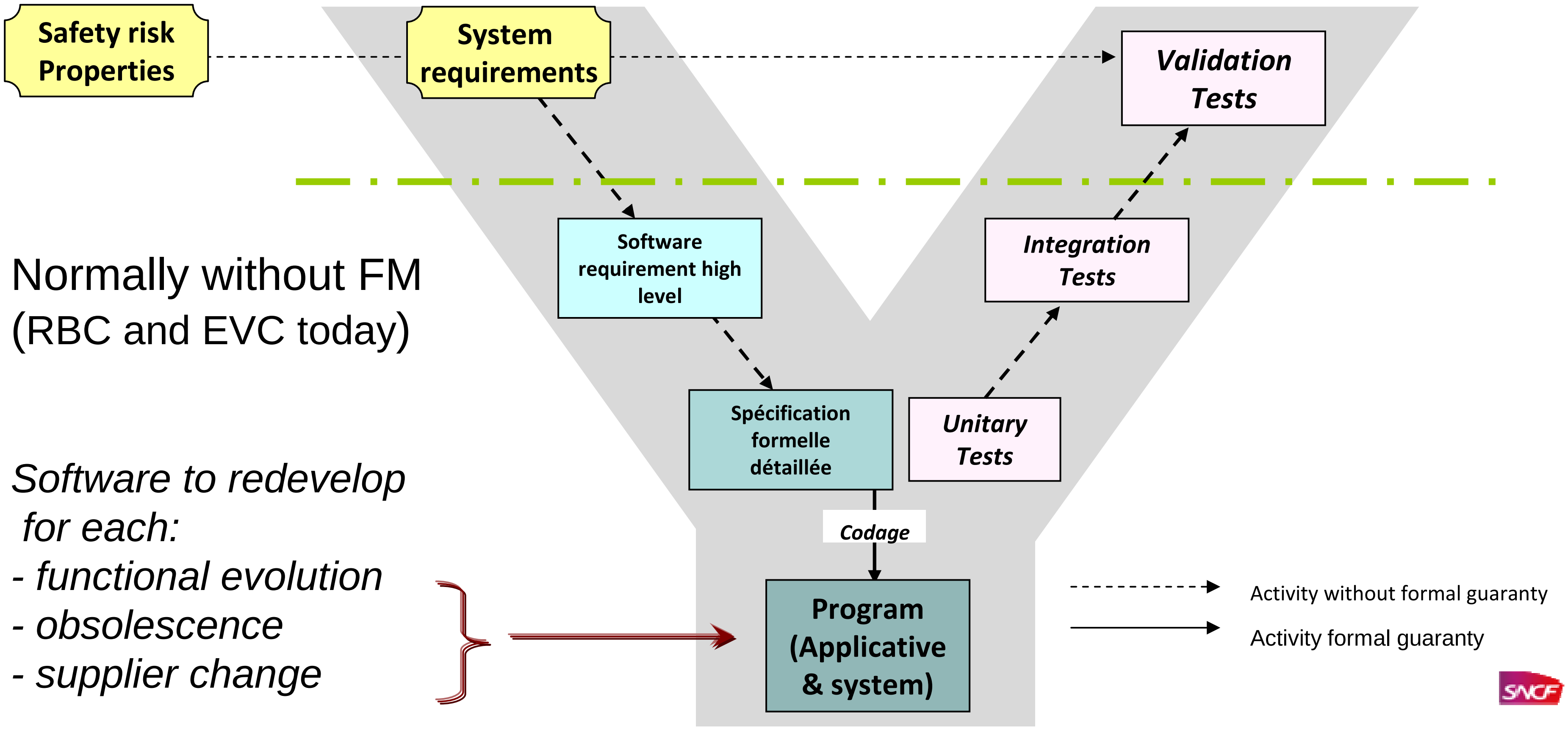
→ the applicative software is high connected with the hardware

→ all evolution of hardware or software will still lead to the redevelopment of the complete system… with the associated economic consequences

→ an platform obsolescence leads to the redevelopment of the applicative software… and to carry out associated safety and validation work

# SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

# FORMAL METHOD & SOFTWARE DEVELOPMENT



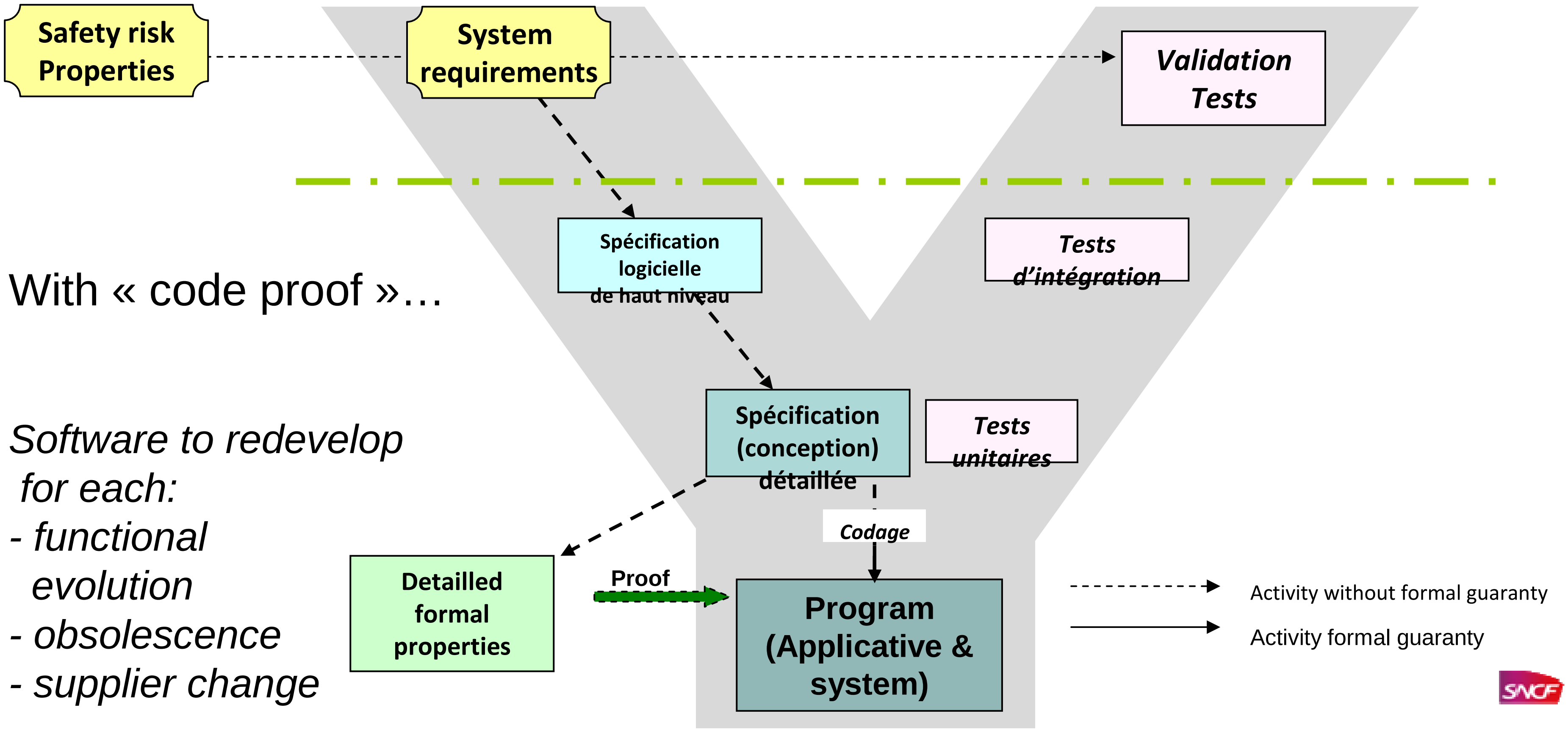

Normally without FM
(RBC and EVC today)

*Software to redevelop for each:*
- *functional evolution*
- *obsolescence*
- *supplier change*

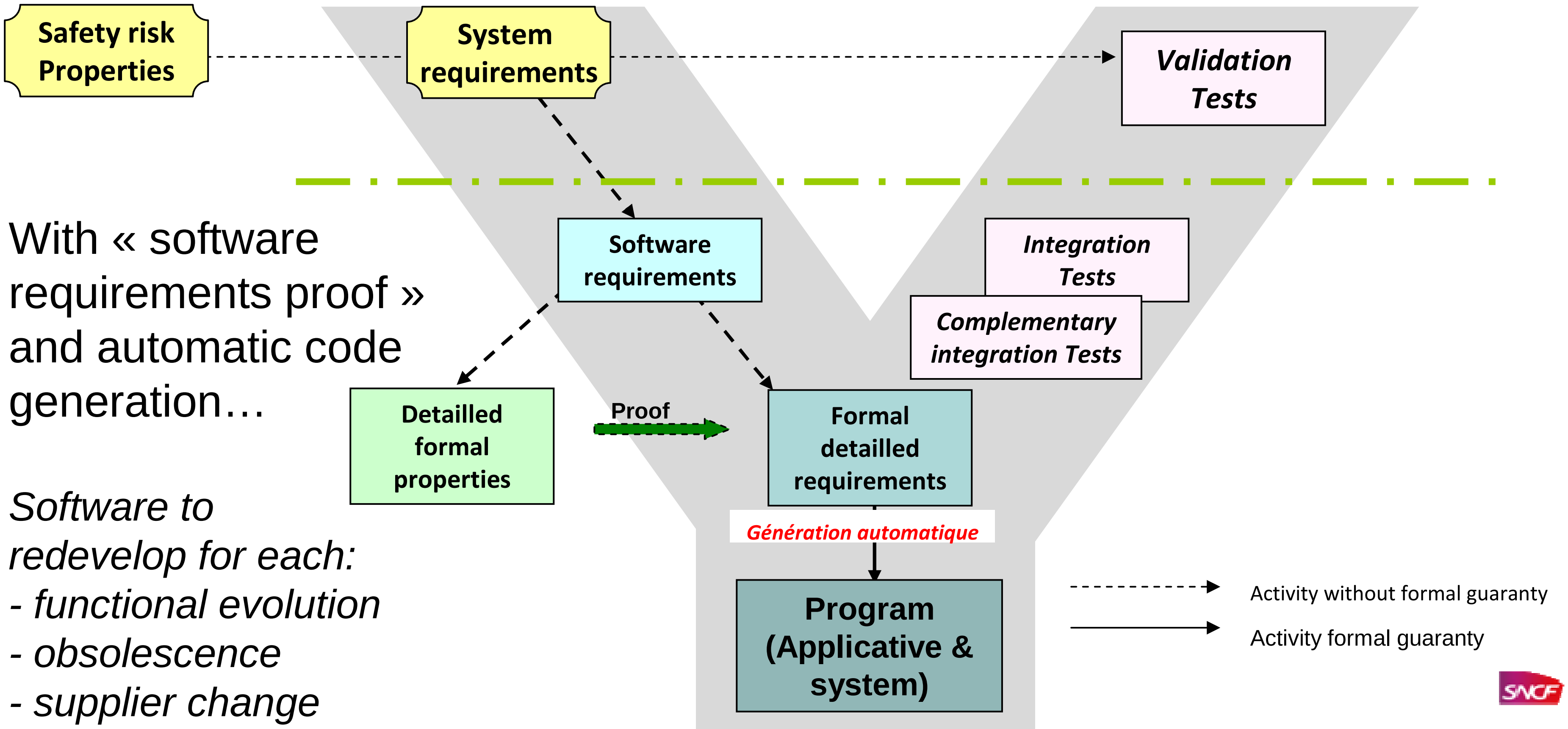
# FORMAL METHOD & SOFTWARE DEVELOPMENT

With « code proof »…

*Software to redevelop for each:*
- *functional evolution*
- *obsolescence*
- *supplier change*



# FORMAL METHOD & SOFTWARE DEVELOPMENT

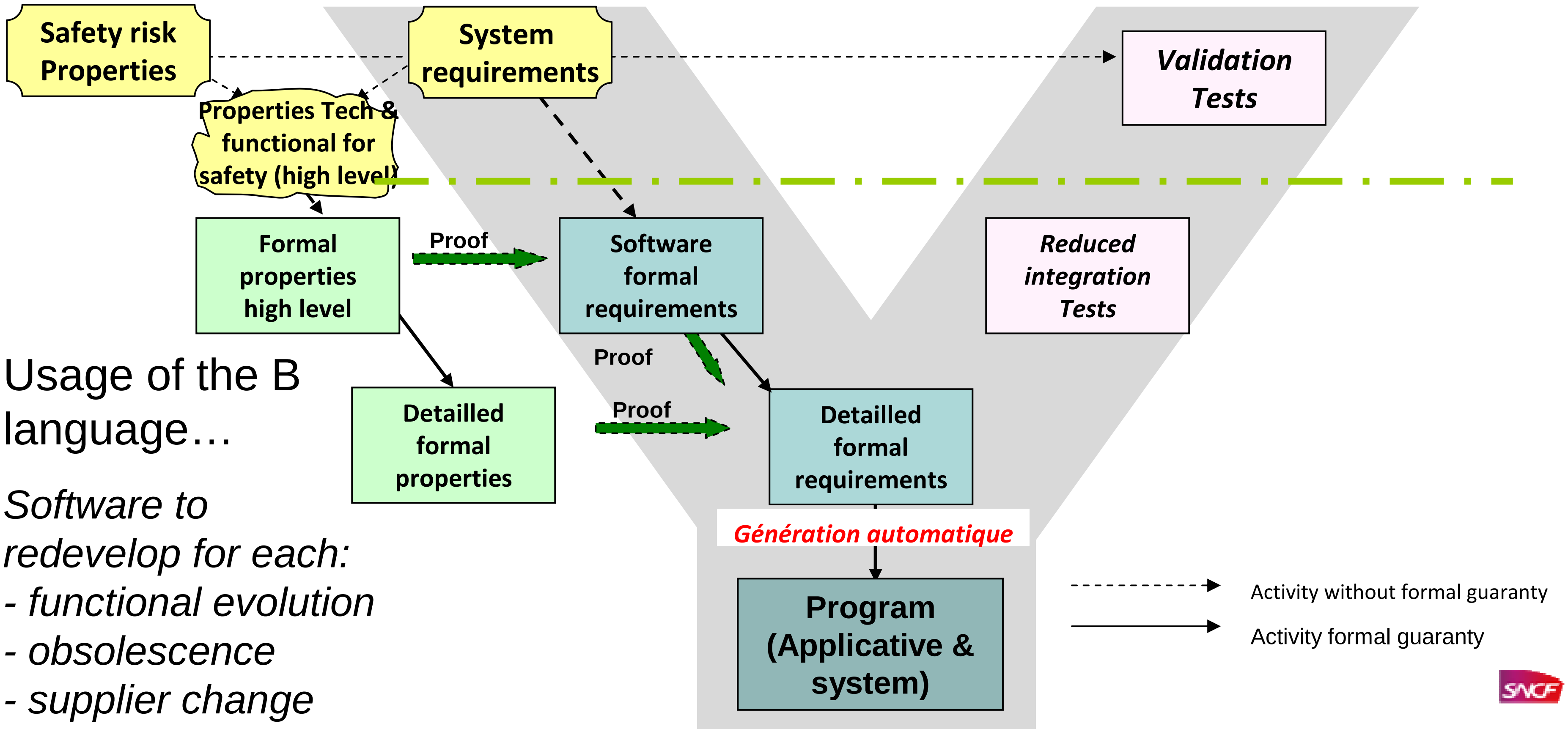



With « software requirements proof » and automatic code generation…

*Software to redevelop for each:*

- *functional evolution*
- *obsolescence*
- *supplier change*

# FORMAL METHOD & SOFTWARE DEVELOPMENT




Usage of the B language…

*Software to redevelop for each:*
- *functional evolution*
- *obsolescence*
- *supplier change*

## FORMAL METHOD & SOFTWARE DEVELOPMENT

# Classical approaches

- in fine :
  - No clear distinction between « formal requirements «  (extract from the SRS and national particularities)
  - A software grouping the “applicative” and the “system” (real time linked to the hardware…) aspect
  - The impossibility to modify the “applicative” side without to repeat the process,
  - The impossibility to modify the “hardware (platform)” side without to repeat the process
  - No taking into account of all the context of use of the system (EU and national)

# FORMAL METHOD & SOFTWARE DEVELOPMENT

## Proposed approaches

- Today in use by Thalès for SNCF and RATP, General Electric… :

  → To distinguish the WHY (common applicative rules or requirements for all suppliers) from the HOW (software related to the supplier hardware architecture & in charge of the real time interpretation or execution of the WHY) – during all the development cycle

  → To reach the previous targets:

  - Use of the same applicative software or functional requirements (given by a IM or Undertaker) of different suppliers platform, without modification…
  - Manage a platform obsolescence without to redevelop and reproof the applicative software or functional requirements…

# Proposed approache

- How to proceed?

→ Write the functional requirement and safety properties & external postulate:
- in a formal way (not ambiguous and provable),
- in a real time interpretable and simulable way
- describing the helpful constraints to facilitated the formal proofs
- choosing a expressive language comprehensible by railway experts

→ Each suppliers define a platform able to:
- interpret in a non ambiguous way all functional specification
- respect the define interpretation rules
- manage in a safe way the platform

## Proposed approach

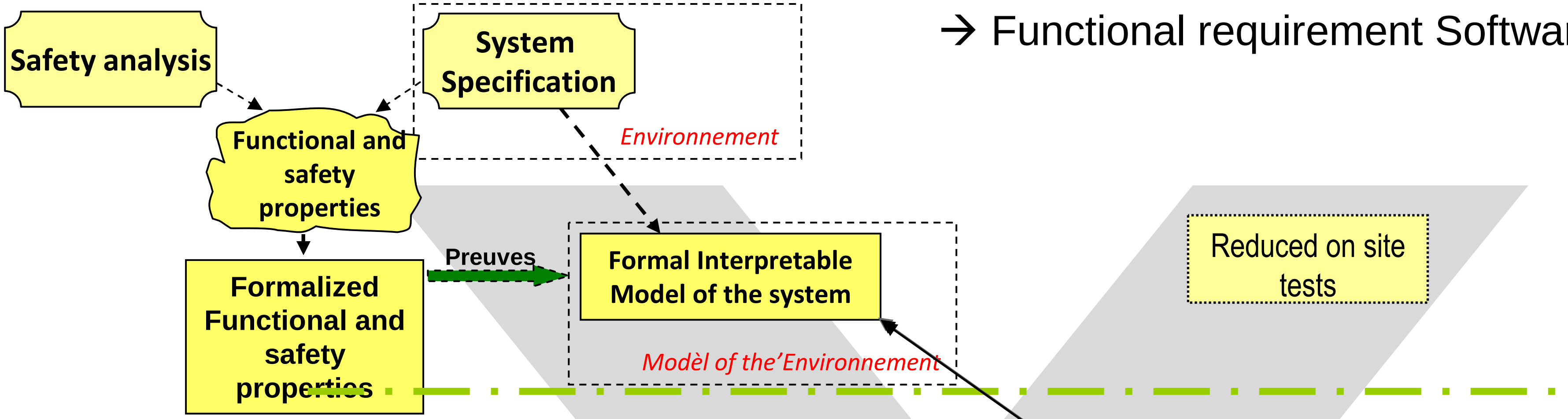
- How to proceed?

  → Two development process:
  - OpenECS group : write formally and prove the functional requirement in regard of each national environment
  - Supplier : to develop (with formal proof?) the platform

  → Around a Domain Specific Language (DSL) define by:
  - its semantic
  - its expressive representation
  - the writing rules
  - the interpretation rules

# FORMAL METHOD & SOFTWARE DEVELOPMENT

→ Functional requirement Software




**Permit to the asset manager to**:

- Share his requirement (EU and national) to all the suppliers
- validate the functional requirements
- prove the safety and the functional properties
- exhibit the unsafe situations

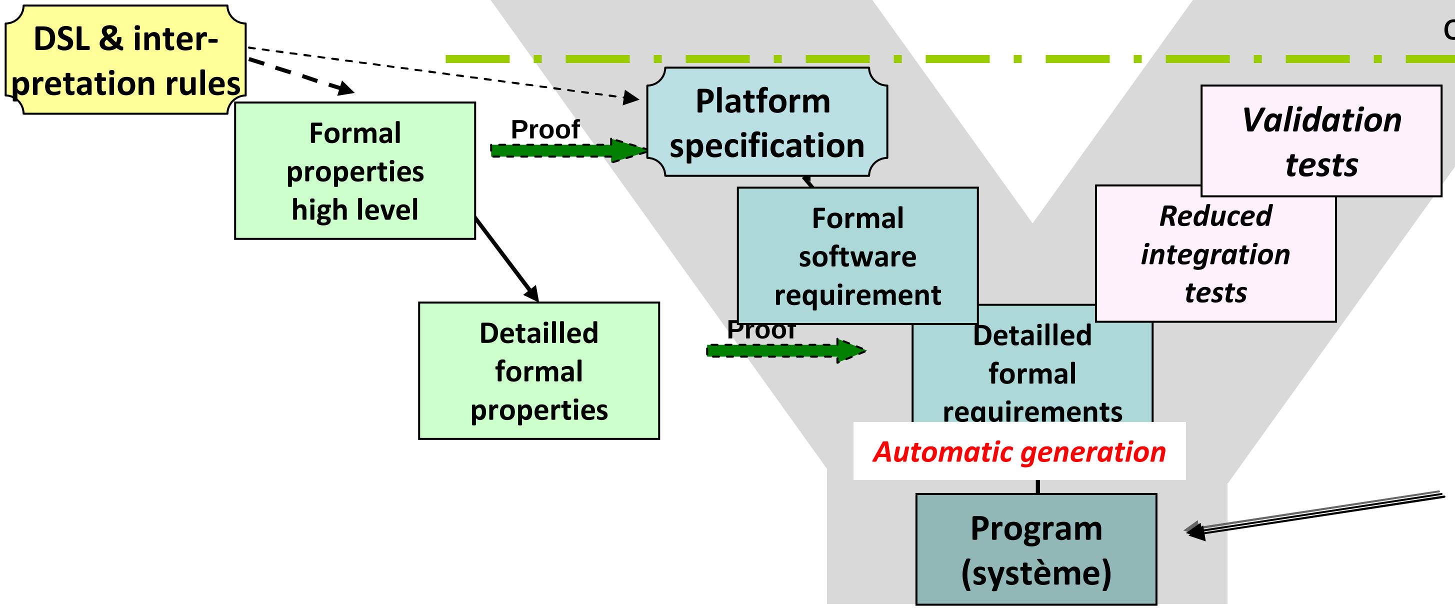
→ during all the life cycle (evolutions…)

*Software independent with obsolescence or supplier change*

# FORMAL METHOD & SOFTWARE DEVELOPMENT

**Permit to the supplier to:**

- Realize a platform reusable for many applications in different countries
- Prove the safety requirement linked with the DSL
- manage the obsolescence (different successive platform, possible engagement over many decades...)

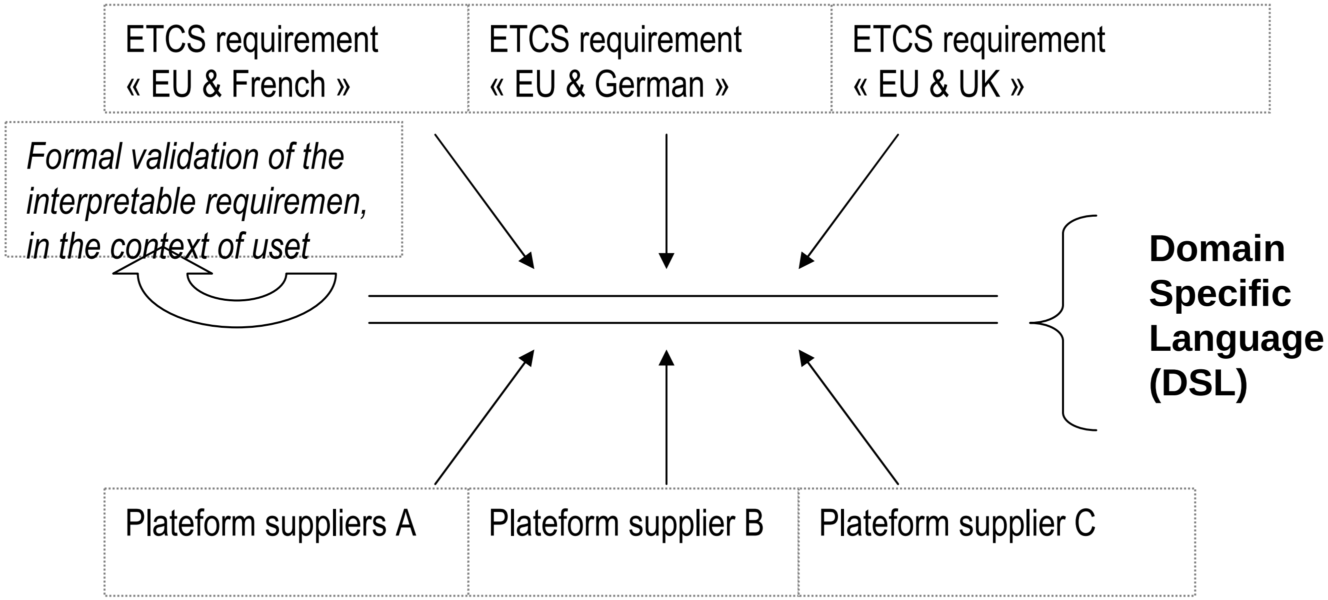



*Software independent of all functional evolution*

# FORMAL METHOD & SOFTWARE DEVELOPMENT

## Proposed approach

- The goal ➔ « a winner winner approach available on the long term »



## FORMAL METHOD & SOFTWARE DEVELOPMENT

# Proposed approach

- Safety outcome:

→formal method used from the highest functional level

→responsibility clarification in term of system safety between:
- the rolling stock manager / functional requirements & safety properties & environment postulates
- the supplier / realization of the products in respecting the right safety level

→possibility to management “softly” the main economic issues : functional evolutions, the obsolescence problems…

# SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

**Summary**:

1. **Problems to solve: safety and durability**
2. **Formal Method & software development**
3. **Interpretable requirement**
4. **Formal validation method**
5. **Tools developed by SNCF**
6. **Examples of use**

## INTERPRETABLE REQUIREMENTS

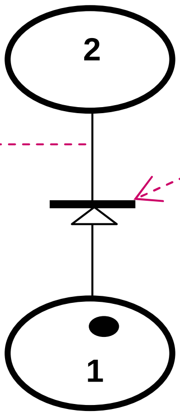
# Interpretable deterministic Petri Nets

→ AEFD language allows a deterministic functional specification and a deterministic interpretation of signalling functions (competing automats with constraints):

- The interpretation is realisable without indecision
- The interpretation is not dependant of the graphs reading order
- The interpretation is realizable in real time

AEFD Language

**Actions (sequential)** :
- Launch temporisation
- Value a real expression
- Activate a binary output
- send a message…




**Event** which starts transition :
TC_2005_free (or End of temporisation, or Valuation of an real expression…)

***Condition*** :
*TC_2002_free AND TC_2003_free*

# INTERPRETABLE REQUIREMENTS

## Interpretable deterministic Petri Nets

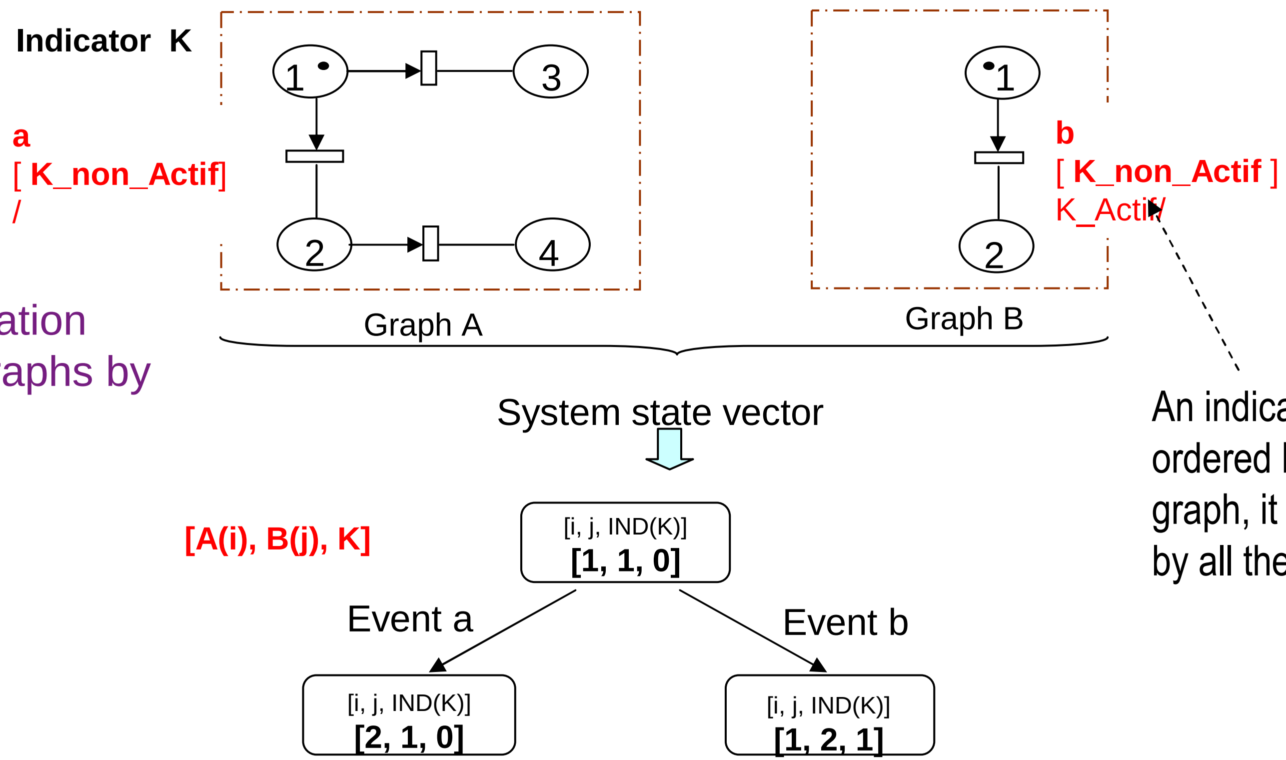
- AEFD definite language allows a deterministic functional specification and a deterministic interpretation of signalling functions:
  - The interpretation is realisable without indecision
  - The interpretation is not dependant of the graphs reading order
  - The interpretation is realizable in real time

*Selected notation in the textual interpretable file form*

```
…
Graph name
1
2
TC_2005_Libre OR FTP_TC2005_2 Event
TC_2002_Libre AND TC_2003_Libre AND
TC 2005_Libre AND FTP_TC2005_2 Condition
Signal_Open; DTP_Signal_1; Action
…
```

## INTERPRETABLE REQUIREMENTS

# Interpretable deterministic Petri Nets



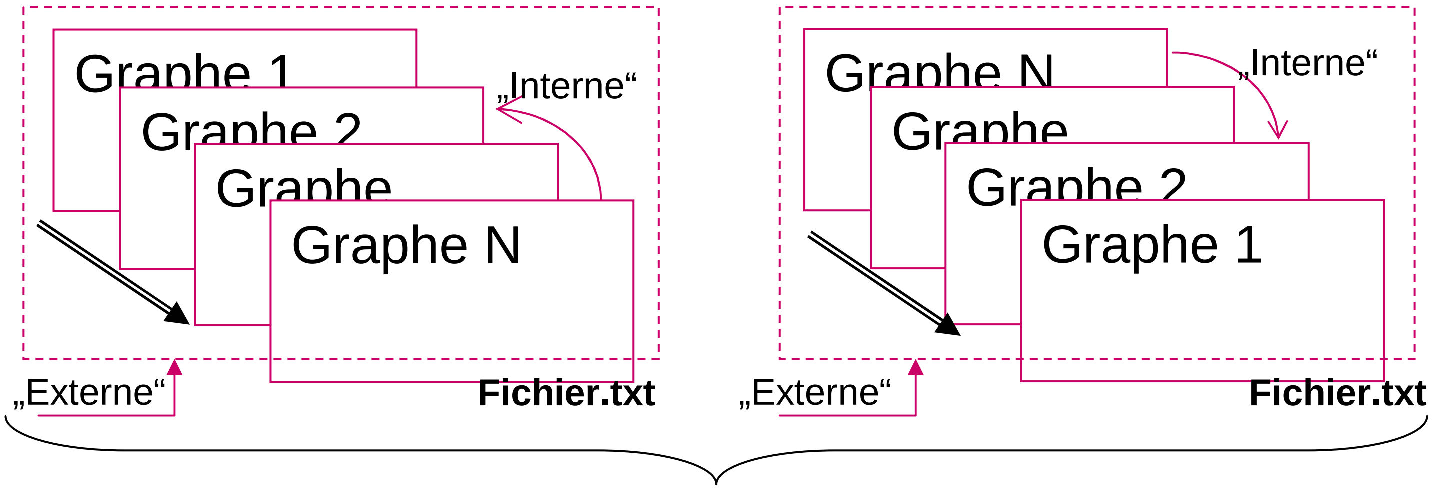

Communication between graphs by Indicators:

An indicator is ordered by only one graph, it can be read by all the graphs

## INTERPRETABLE REQUIREMENTS

# Interpretable deterministic Petri Nets

→ With the selected written mode, the Petri nets are interpretable in a deterministic way, without ambiguity and in real time




**An unique reachable, finished and countable system states**

## INTERPRETABLE REQUIREMENTS

→ The SNCF and RATP has jet define and put into service hundred of interlocking system designed:

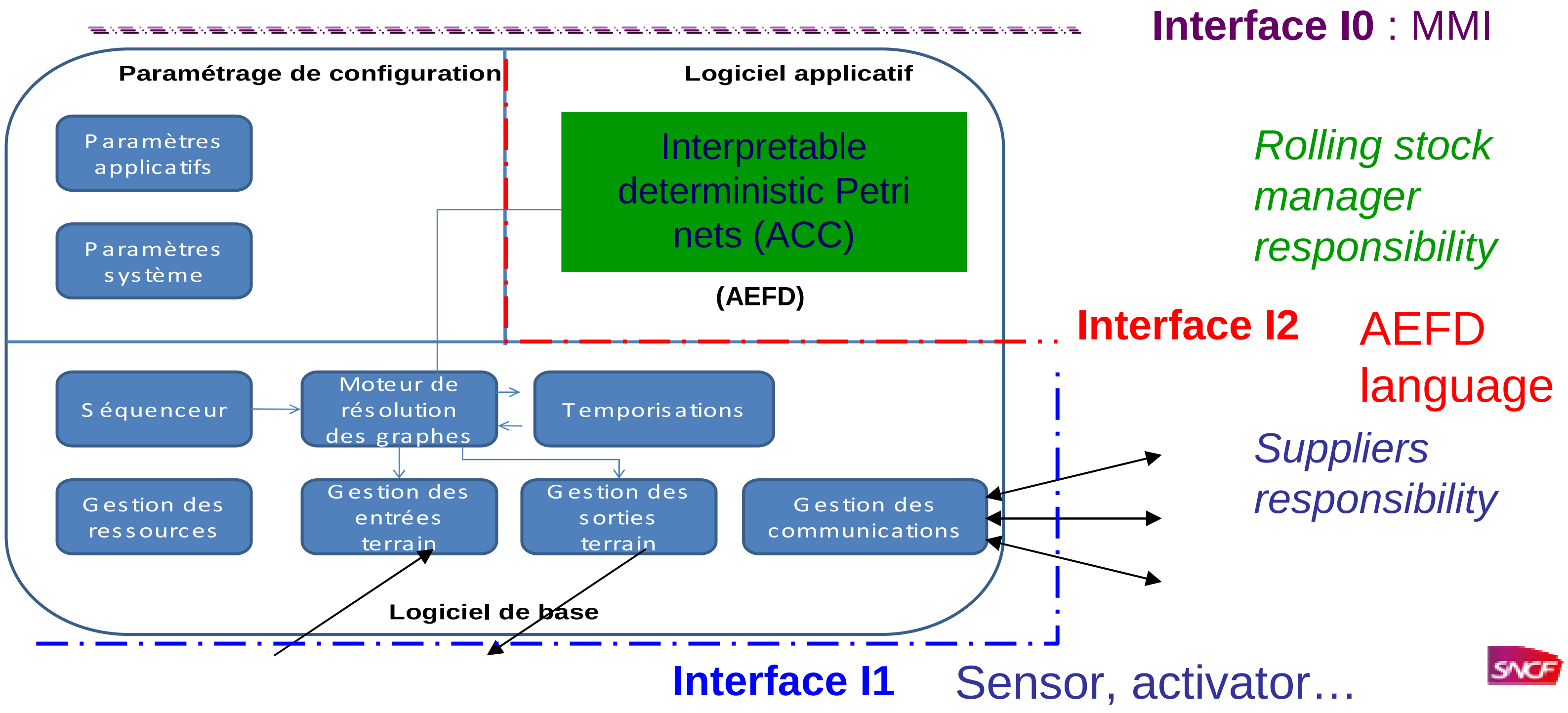
- To carry out a clear separation between « hardware & basic software » (*suppliers view*) and « functional software » (*infrastructure manager view*)
- To carry out clear interfaces between the computerized module and rest of the railway system
- To carry out the specification and the functional software with interpretable deterministic Petri nets (*interpreted in the target machine*)
- To reduce the safety demonstration costs and to allow a formal validation of the functional software in the real environment conditions of the interlocking system

  $\Rightarrow$ the method have to be applicable by signalling engineers

# INTERPRETABLE REQUIREMENTS

## → The architecture

use common functional interfaces for all the safety systems (for all the suppliers)



## SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

**Summary**:

1. **Problems to solve: safety and durability**
2. **Formal Method & software development**
3. **Interpretable requirement**
4. **Formal validation method**
5. **Tools developed by SNCF**
6. **Examples of use**

## FORMAL VALIDATION METHOD

→ Formal validation method:

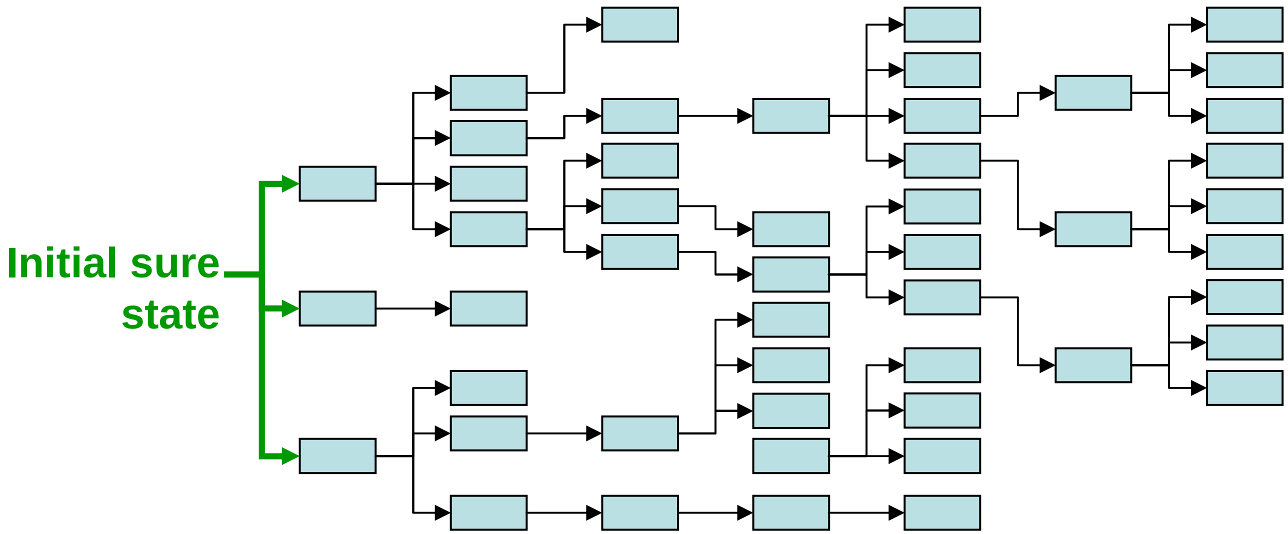
The proof is brought on the final interpreted final functional model (rolling stock manager vision)

The suggested method is a formal validation method

The method is applicable on the functionalities written with deterministic and interpretable Petri nets

# FORMAL VALIDATION METHOD

→ The functions written with deterministic and interpretable PN can be represented by an unique reachable system states:




Systematically system states research

*Post* (Initial_state)*

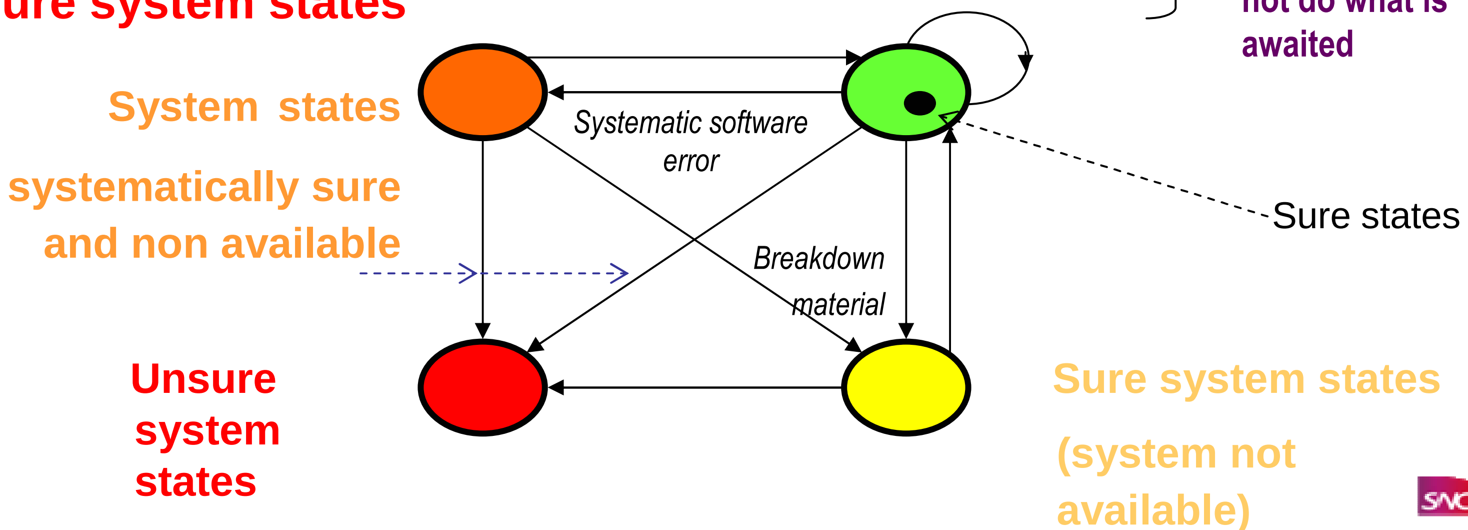
# FORMAL VALIDATION METHOD

→ Each state system can be associated with one with the 4 categories:

- System states sure and available
- System states sure but not available
- States system systematically reachable sure system states (not available)
- Unsure system states

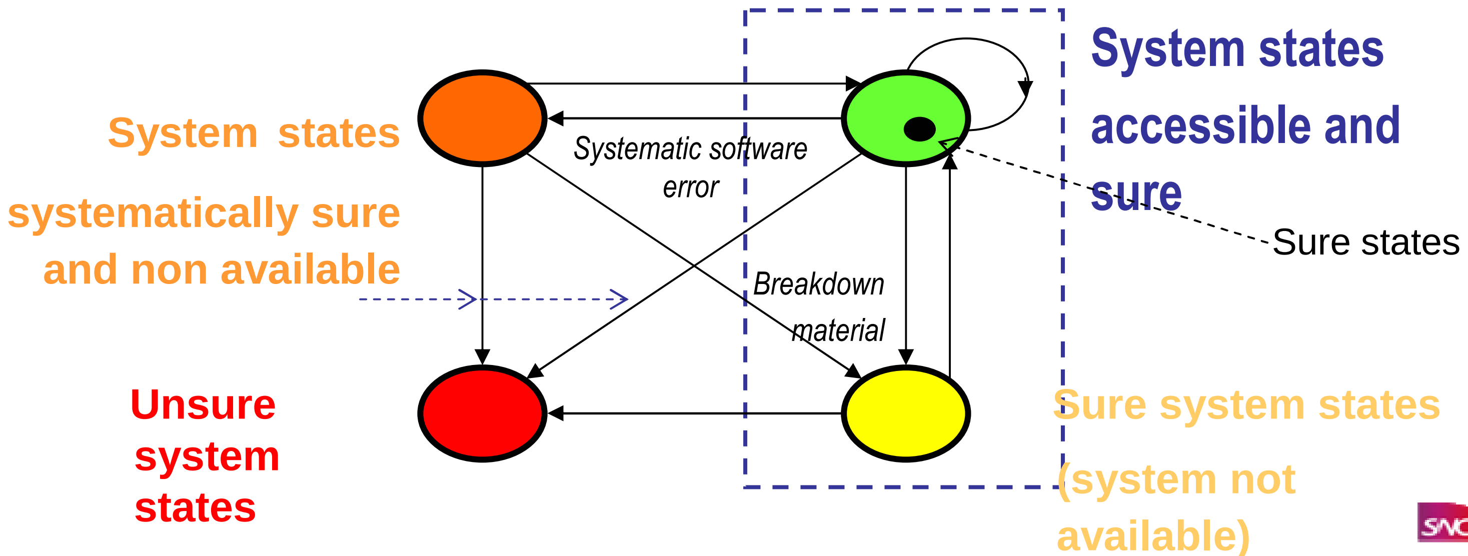
The system does what is awaited

The system doesn't not do what is awaited



## FORMAL VALIDATION METHOD

→ The safety properties must be written in order to be able to prove that no "sure but not available system state" (overabundant) or „unsure system state is reachable



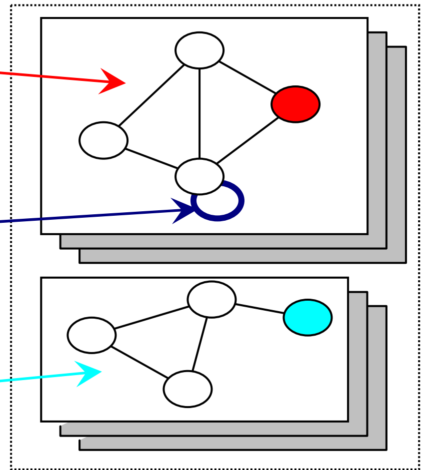
## FORMAL VALIDATION METHOD

→ The safety properties have to be written with « proof automats », by signalling engineers, in three stages:

Stage 1: description of the **safety properties or incompatibilities** they have to be ever respected by the railway system

Stage 2: description of the waited functionalities for the detection of « possible » **overabundant conditions**

Stage 3: **functional postulates** description (rules, environment…) limiting the validity field of the proof



Simple text file

## FORMAL VALIDATION METHOD

→ The proof can be accomplished in the following way with the use of the « functional graphs » and « proof graphs »:

$$\textbf{Post* } (\textit{Etat Initial}) \cap \textbf{ Unsafe States} = \phi \textbf{ ?}$$

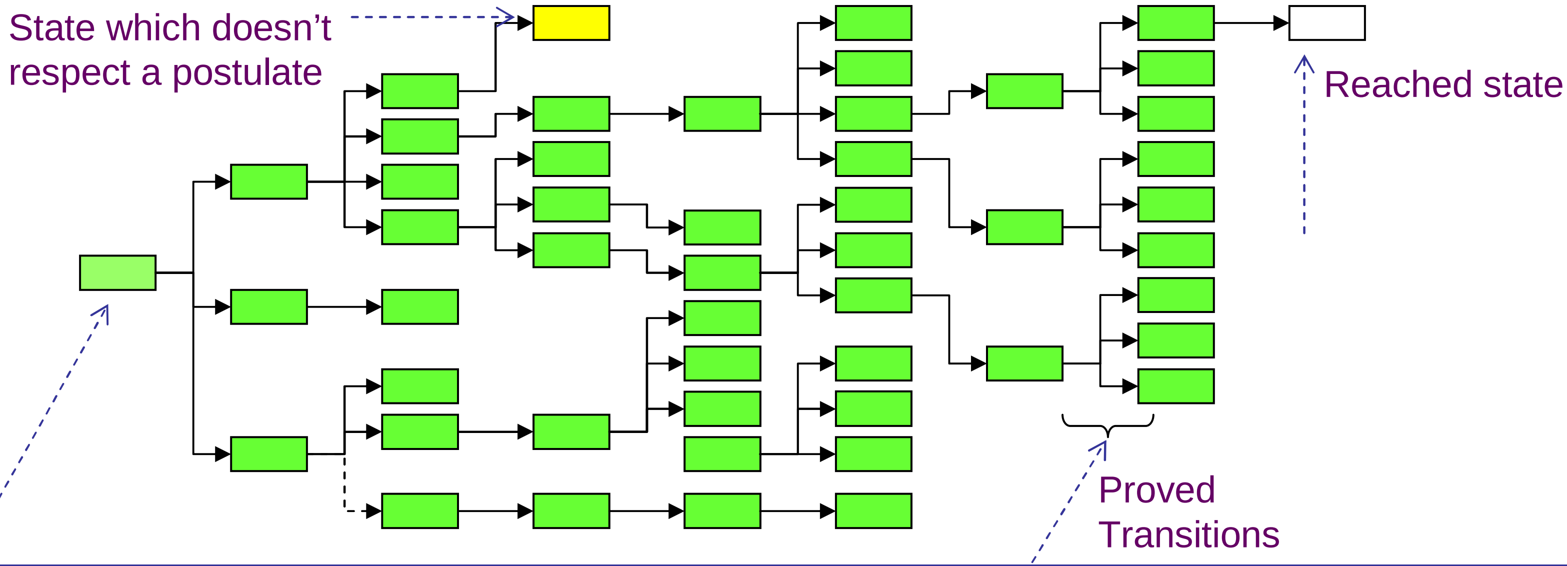
→ The proof principle is the following:

*«If a group of properties is true for a given system state, and that this group remains proved during a transition between system states, then the property is true in the new system state»*

This proof can be reproduced for every level of system states to the point of being applied by recurrence to all reachable system states. The initial state have to be safe.

## FORMAL VALIDATION METHOD

→ The basic principle is:




| | | | | | | |
|---|---|---|---|---|---|---|
| Initial safe state | **AND** | All the possible transitions are known | **AND** | All the reachable transitions are proved | ⇒ | All the reachable system states a safe |

# SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

<u>Summary</u>:

1. Problems to solve: safety and durability
2. Formal Method & software development
3. Interpretable requirement
4. Formal validation method
5. Tools developed by SNCF
6. Examples of use

## TOOLS DEVELOPPED BY SNCF

**Appropriated tools were developed by SNCF Infra** to accomplish:

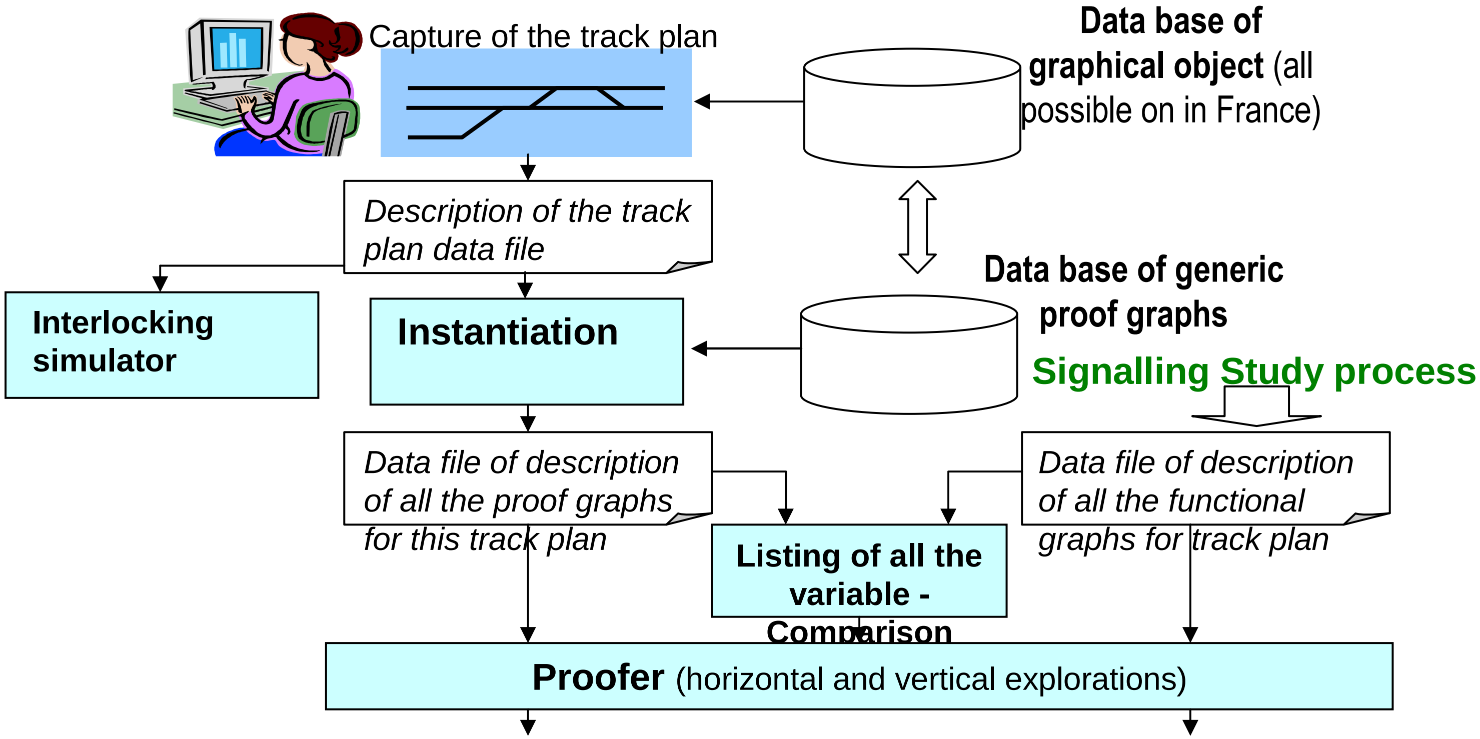
- Automatic definition of the safety properties and the postulates describing the conditions of use,
- Formal writing of these properties in order make the proof,
- Definition of the initial system state in which all the safety property are true,
- Evaluation of the safety properties by recurrence for each transition between system states. The safety properties are evaluated until all safety properties are true, otherwise the proof is stopped.

⇒**Their application possible by persons without special mathematical education but only a good signalling knowledge**

⇒**Their application leads to a significant reduction of the validation costs and delays** .

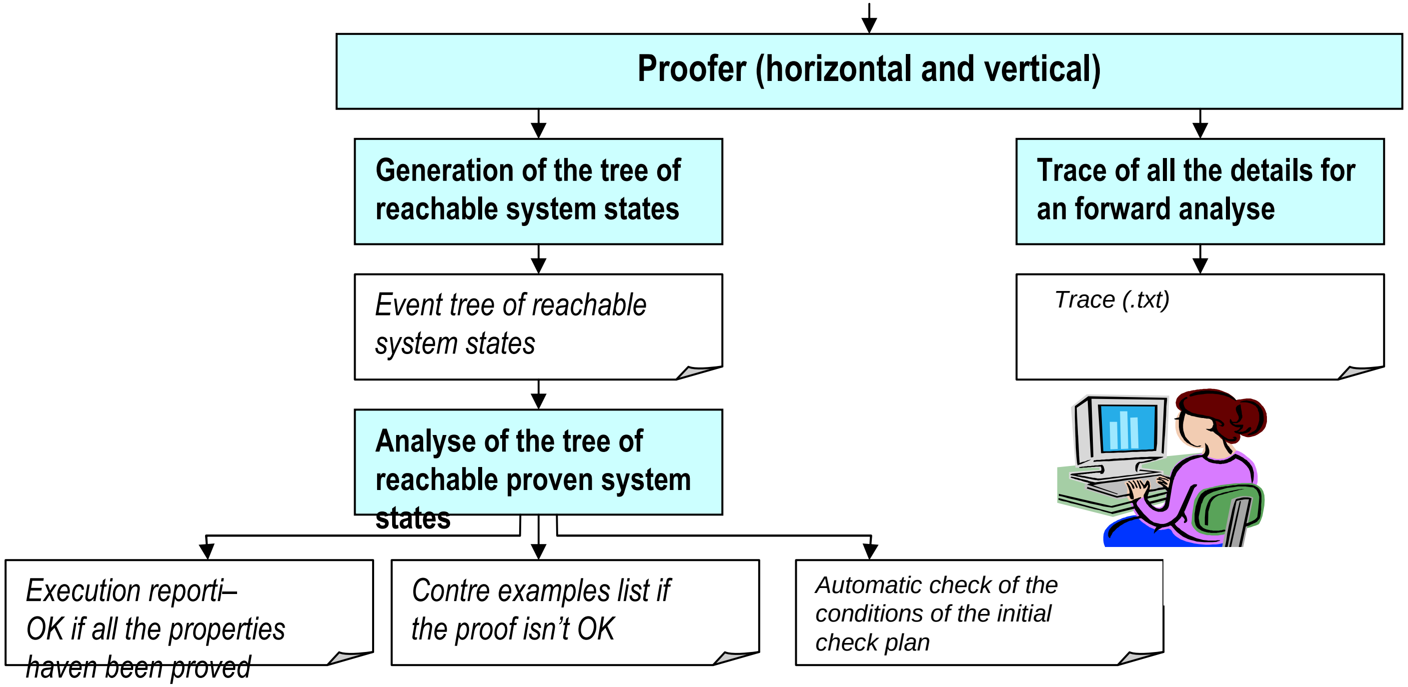
# TOOLS DEVELOPPED BY SNCF

- **Formal validation process - Step 1**



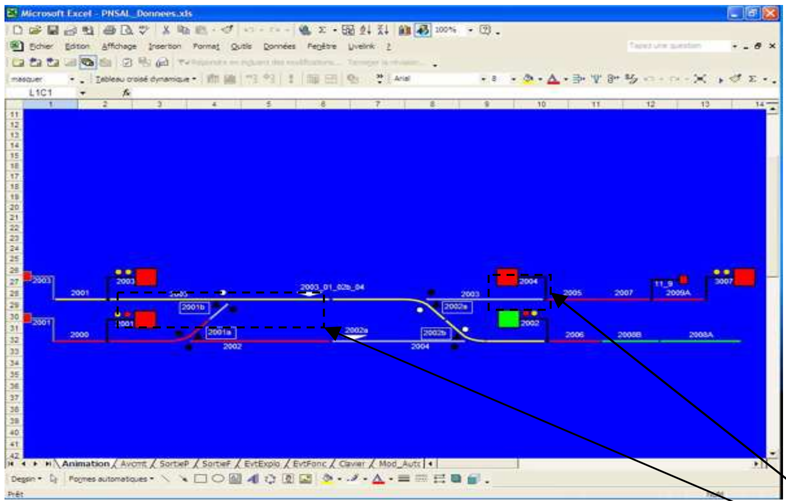
# TOOLS DEVELOPPED BY SNCF

- **Formal validation process - Step 2**



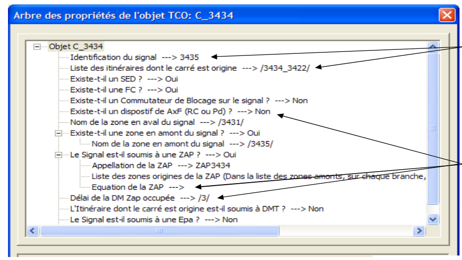
# TOOLS DEVELOPPED BY SNCF

- **Track plan example and safety properties instantiation**



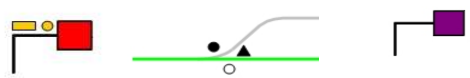
**Capture of the track plan by topological association of graphical object**

Saisie des paramètres : instanciation des objets




Elément ou listes d'éléments permettant d'instancier les Automates de preuve

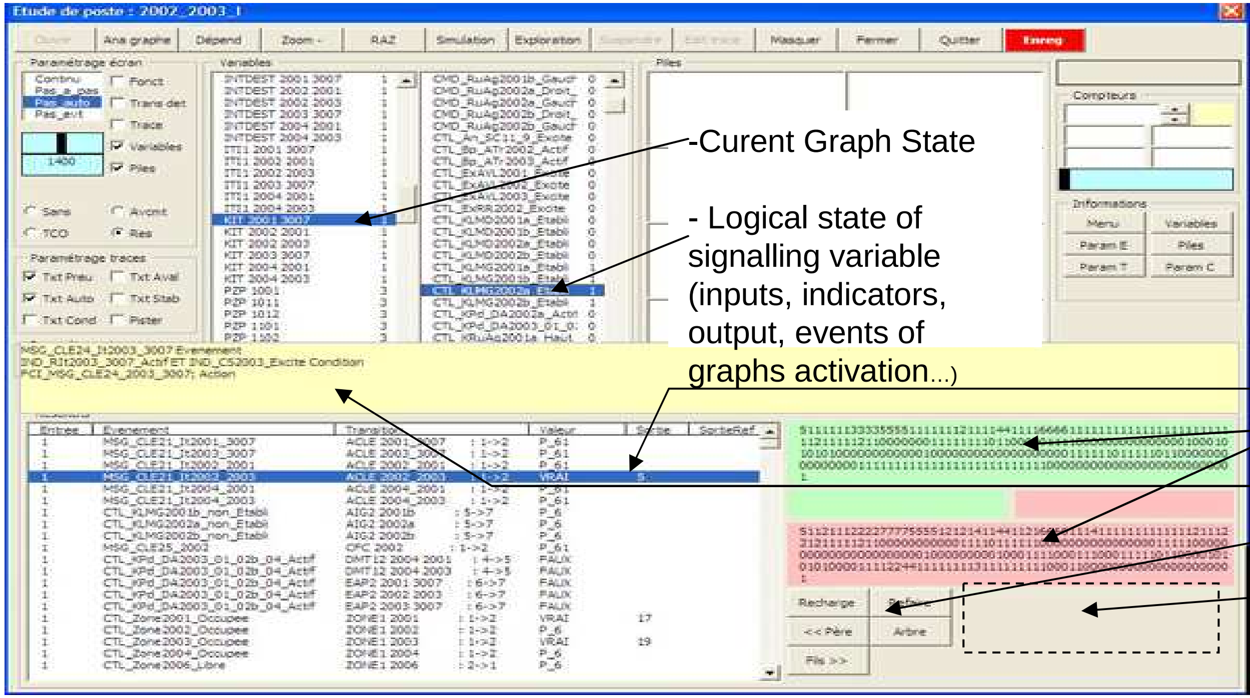
Nombre de réponses demandent une réflexion préalable de l'essayeur (élaboration du cahier d'essais)



Graphical Objects topological laid out and instantiate: automatically or manually by the signalling engineer in charge of the proof:
- Signal object,
- Switch object…

# Application - Formal validation tools chain

- **Proof tool view**




Control screen of the Proof tool

System state change selected (blue)

System State Vector before the selected transition

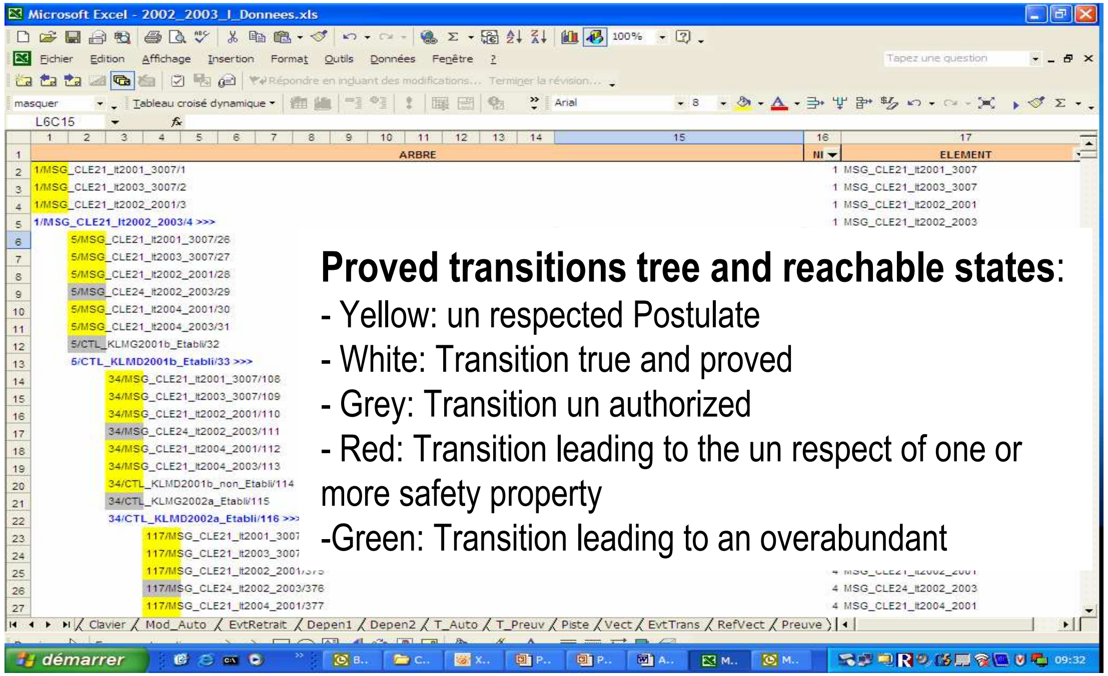
System state Vector after the selected transition

Details of the transition

Screen button

# Application - Formal validation tools chain

- **Reachable states tree tool view**



**Proved transitions tree and reachable states:**

- Yellow: un respected Postulate
- White: Transition true and proved
- Grey: Transition un authorized
- Red: Transition leading to the un respect of one or more safety property
- Green: Transition leading to an overabundant

(1) To carry out the vivacity check

(2) To carry out the execution report

(3) To presenter the results with ergonomic manner

(4) To carry out the tree of the transitions tree

## SNCF FORMAL VALIDATION METHOD OF INTERPRETED REQUIREMENTS

<u>**Summary**</u>:

1. **Problems to solve: safety and durability**
2. **Formal Method & software development**
3. **Interpretable requirement**
4. **Formal validation method**
5. **Tools developed by SNCF**
6. **Examples of use**

# Thanks for your attention

**Contacts:**
The SNCF Infrastructure
Dr. Marc ANTONI
18 rue de Dunkerque
75018 PARIS

(+33) 6 29 91 77 43 - marc.antoni@sncf.fr

# Any questions?