Deriving safety properties of critical software from the system risk analysis

J.L. Boulanger]Φ, V. Delebarre] and S. Natkin]Φ

] **CESIR, 124 rue Vieille du Temple 75003-Paris France**
cesir@imaginet.fr

Φ **CEDRIC/CNAM 292 rue Saint Martin 75141 Paris Cedex 03 France**
boulang@cnam.fr

Summary

Safety properties of critical software are consequences of the application safety properties (i.e. the front collision of two trains must not occur), and of the system design choices. This paper presents the first results of a SNCF and CESIR join research project which purpose is to design a constructive and formal method to derive, at each design level, the safety properties of sub-systems from the System Preliminary Hazard Analysis. One of the goals of this method is to obtain, at the lowest level, properties of the safety software, which can be checked either by formal proof or by testing. The method relies on two concepts: the safety kernel, proposed by J. Rushby, and a generalization and formalization of the notion of "restrictivity", used in classical safe hardware design. An application to MAGGALY ([MAIRE 93]), the automatic pilot of Lyon unmanned Subway, is presented.

1. Introduction

1.1. Usual Process for safety critical systems demonstration

In safety applications, safety demonstration relies on two kinds of activities:

- At the system specification level, risk analysis is performed. Undesired events are identified by safety engineers and the risk analysis checks that system specifications are robust enough to avoid undesired events occurrence. A system safety framework describes the activities to be performed throughout the life cycle.
- Testing activities: dedicated system engineers specify a testing strategy, which is «safety oriented» and a specific team, performs testing tasks.

Once system level specifications and basic implementation concepts have been "validated" by risk analysts, the development is organized as for any other project. The safety testing strategy is destructive. There is a discontinuity in the safety activity process, so those safety objectives are not traceable easily. Also, the problem of testing completeness and stopping is difficult.

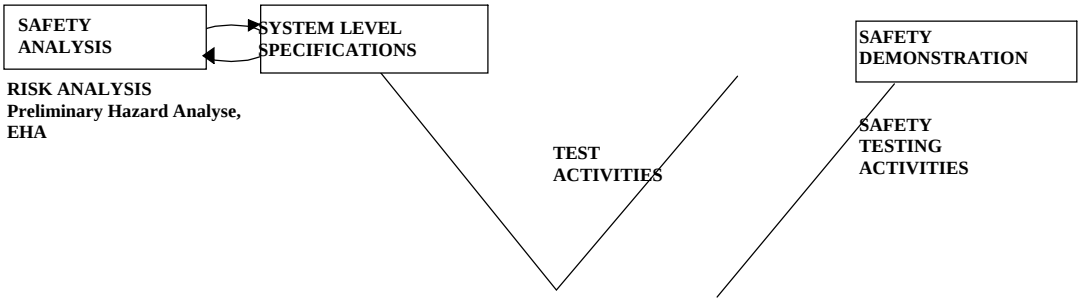


Fig.1: Classical life cycle.

1.2. Software formal process for safety critical systems

Considering the software life cycle (Fig.2), formal methods enable to integrate validation and verification activities within the design and development process. Refinement and proof control design and development steps. As a consequence, safety properties are traceable if they have been included in the formal model. The "missing link" of such an approach takes place between system analysis, risk analysis on one hand, formal modeling on the other hand. In fact, there is no traceability between safety objectives at the system level and properties, which have been proved during the software life cycle. As a consequence, there is no traceability between proven properties and safety testing strategy.

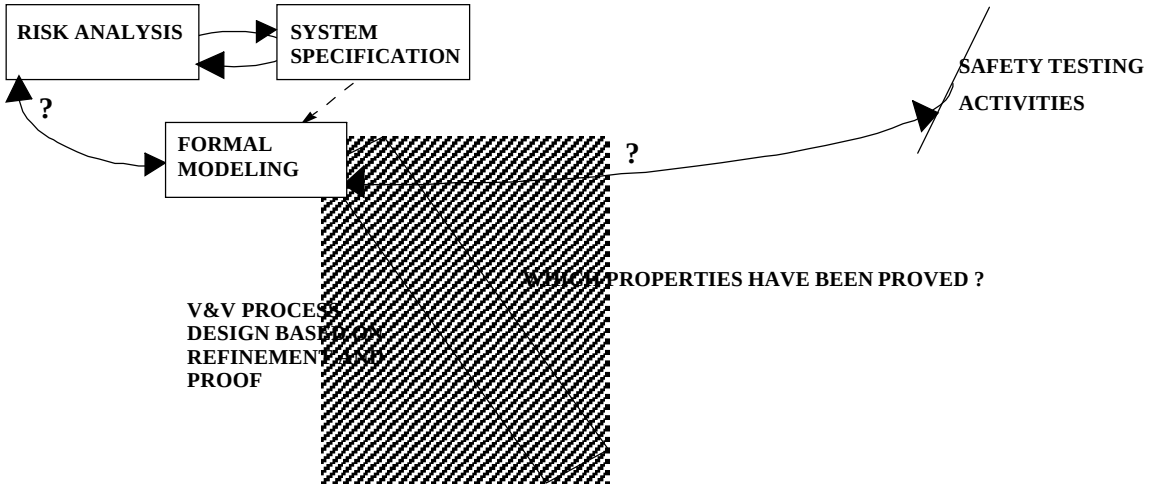


Fig.2: Software formal process for safety critical system.

A classical distinction in system modeling is that between safety and liveness properties. A safety property stipulates that specific "bad things" do not happen during execution while a liveness property stipulates that certain "good things" do happen (eventually).

1.3. Formal Methods for safety critical systems

"A system may be proved if and only if it has been designed to be proved". We think that formal methods (either model checking methods or re-writing based methods) are efficient and applicable for large safety critical systems if the system design makes the system provable. The objective of this study is to define a system design process, which includes the demonstration of safety property achievement. It is a constructive approach based on the Safety Kernel Concept ([Rushby 89]).

1.4. Safety Kernel

In security area, a security kernel is a set of components, whose **correction properties** ensure that **security properties** for the whole system are verified. In this paper, we are to point out how this approach is applicable to safety critical systems. Kernel based architectures are characterized by a set of second order properties: $\forall \alpha \in op^*, P(\alpha)$. Such properties state that, whatever operations or events (op*) which occur from the kernel environment, the systems properties are verified.

Such an approach implies the twofold design features:

- The safety properties of the system must be formalized in terms of kernel functions and related correction properties.
- No assumption is done for the system components, which are not within the kernel.

2. Definitions

2.1. Safety properties

States, walks and Properties

The behavior of a system is the set of walks that it can generate; a specification is likewise a set of walks. A property is a predicate on walk.

Behaviors, specifications and properties should be prefixed closed: if a given walk satisfies a specification, for example, so should all its prefixes.

Undesired event

For operational phases of a system, the system behavior is perceived as a safe behavior if undesirable events do not occur. Undesired events may be identified in the early stages of system analysis through risk analysis activities. It is based on system operational phases definition, system basic principles and system level specification. On the basis of undesired event identification, the set of safety properties (Pi) is defined:

"P_i: the undesired event E_i will **never** occur"

Safe states

The safety properties must be mapped onto the system behavior description. A random process on a state space S (discrete or continuous) may describe the system behavior. Let us note $\omega = ((X_1,t_1),(X_2,t_2),\ldots,(X_i,t_i),\ldots)$ a random walk in the states space, where X_i denotes the state which has been reached in i steps (transitions) and t_i is the instant time when the system enters state X_i. The system behavior is safe if no walk contains a state where a safety property is not verified.

Transitions are controlled by three kinds of events:

- Environment interactions (A is the set of associated transitions).
- Commands set by the control-command subsystem (C is the set of associated transitions).
- Recovery commands which can be set and controlled either by an operator or by the control command subsystem (R is the set of associated transitions).

A command $c \in C$ is defined by its local pre-conditions and post conditions. Pre-conditions are evaluated on variables which are monitored and post conditions set control variables.

The system is in a safe state $(X(t))$, $t \in [0,\infty]$ <=>

$P_{i,\ i \in [1,n]}$ set of safety properties, as identified in hazard analysis

$\exists$ a sequence of commands $c \in C / X(t)>$, $t \in [0,\infty]$, $\wedge$ $P_{i,\ i \in [1,n]}$ is verified,

where $X(t)>$ denotes successor states, reachable from $X(t)$ by transitions in A, C and R.

A state where an undesired event could occur is called dangerous : $\exists\ i \in [1..n]$ so that P_i is not satisfied in such a state.

Passive safe state

X is a **passive safe state** if $\forall\ t \in [0,\infty]$, $X(t)>>$ is a safe state, where $X(t)>>$ denotes successor states, reachable from X(t) by transition in A only.

For every system position, system analysis defines risks, which may occur, and corresponding passive safe states. For instance, if we consider trains on track way, and collision risk, the state where all trains are stopped is a passive safe state.

2.2. Safety Kernel

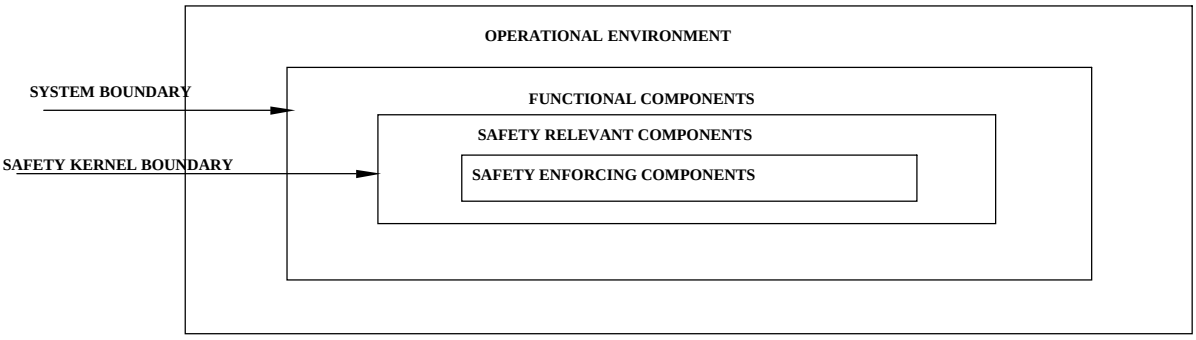


Fig.3: Component Layers.

System organization

Definitions of §3.1 match exactly the safety kernel concept. The system is organized in three layers of components:

- Safety enforcing components ensure that system properties are verified if correctness properties are verified for these safety components,
- Safety relevant components are interface components, which ensure inputs of the safety kernel are independent from functional components,
- Functional components ensure functionality of the system is performed.

Following this organization, control commands are of two kinds: functional commands which are set and controlled by the functional components, safety kernel commands, set and controlled by the safety components.

System analysis activities consist in identifying different component layers and basics principles for each layer.

System behavior

States space of the system may be split in sub-spaces:

- Nominal states are safe states where transitions are due to environment interactions and functional commands.
- Hazardous states are safe states considering safety kernel commands :

  If we consider functional commands and environment interactions only, an undesired event R could occur, which means the system should enter a dangerous state, considering

safety kernel commands, the system always enters a passive safe state in a bounded amount of time: Always (not(R)).

- Passive safe states.
- Dangerous states where an undesired event can occur or not: POT(R) or POT(not (R)).

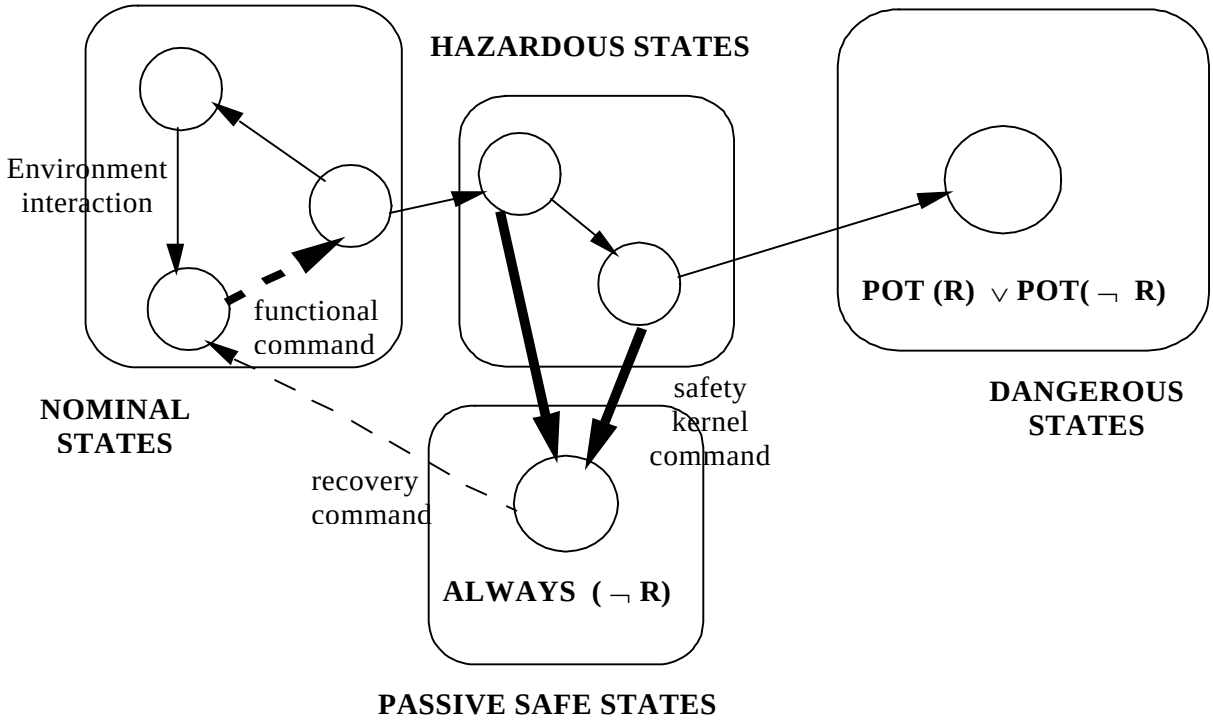


Fig.4: States space of the system S.

Safety based on the kernel design

For such architecture based on a safety kernel, the safety demonstration consists in the twofold demonstration goals:

- Each time the system enters a hazardous state, the control system (the safety kernel indeed) applies a safety kernel command, so that the system enters a passive safe state within a bounded amount of time.
- All hazardous states are detected.

Practically, the problem is to specify components properties so that:

Let $X_H(t)$ be a hazardous state, entered at time t, always ($X_H(t)$ > is a passive safe state)

<=>

($\forall$ t $\in$ TIME, $\exists$ a safety command and a time amount τ, X_H (t) => X(t+τ) is a passive safe state).

Timing constraints τ depend on operational constraints and their specification is a part of safety analysis.

3. Case Study

3.1. Case Study description and objectives

This approach has been applied to collision risk in the context of a subway ground transportation system: MAGGALY ([MAIRE 93]). This system is a fully automated subway system. In this section, the risk, which is considered, is only the side collision between two trains.

We chose this example for the following reasons:

- It implies software, fail safe hardware and software parameters, so that we can point out various safety components in the kernel.

- It is quite simple.

3.2. System breakdown

The whole ground transportation system in its environment is composed of the following subsystems:

Rolling stock. This subsystem performs functions according to dynamic and cinematic on one hand and according to commands set by the control/monitoring system.

Control and Communication. It is broken down in two functional subsystems: one is devoted to operations control (trains description, communication with operators, ...) and the automatic train pilot (ATP). ATP is broken down in ground ATP and on board ATP. Finally ground functions of ATP are distributed among the topology whilst on board functions are replicated on trains.

Vital Signaling. This subsystem monitors trackside equipments such as points, train detectors, tracks circuits and wayside signals. It interacts with ATP for route setting: routes are defined at the ATP level and are set by the signaling system.

Civil engineering. This subsystem defines all the trackside equipments physical implantation, the topology (distances, slopes, ...). It interacts with ATP subsystems through parameters, called "topological parameters".

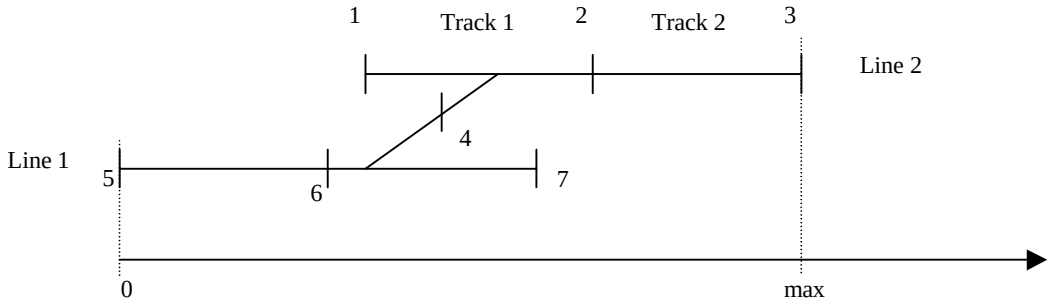


Fig.5: An example of track topology.

Rails are split into track segments where each track segment has an unique identification. Track circuits indicate whether or not a train is standing in the corresponding track section. We model a track topology by a graph in which the nodes represent particular point and the edges represent connectivity for the train traffic.

Let POSITION be the set of all possible train positions. An implementation of a position will be an ordered pair where the first projection is a line identifier 1 or 2 and the second projection is a natural number denoting a geographical position in this line. For a node n, the function *pos(n)* return the geographical position of the node n, for example *pos*(5) = 0.

Must be completed

A tracks tr is a position triple, example Track 1 = (1,4,2) and Track 2 = (2, 3, nil).

We can defines an operation pos_2_track which associate a track to any valid position.

pos_2_track : POSITION → TRACKS

To be define

Operators and users may be seen as a part of the transportation system. Interactions are managed by the Control and Communication subsystem. It is important to take into account end users and operators, since safety properties are related to them.

This system breakdown is a simplified one, sufficient for our case study.

3.3. Safety properties at the system level

Points are track segments connected to more than two tracks segments (POINTS ⊆ TRACK).

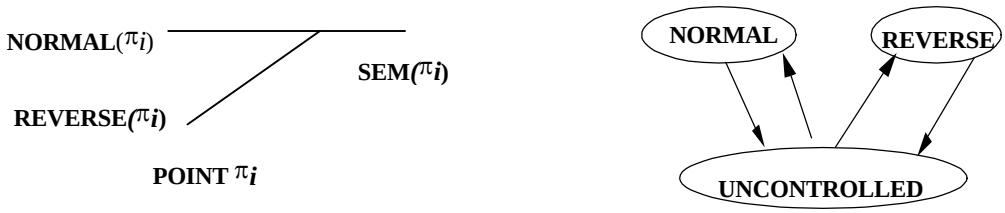


Fig.6: Static and Dynamic definition of points.

We consider the risk of "side collision", which may happen when two routes are convergent. Let πi be the point, *normal(πi)* and *reverse(πi)* are the controlled positions of point πi. We define *track_support(k,t)* the function which associates to any train k its occupancy on track tr at instant time t.

normal	: POINTS	→ POSITION
reverse	: POINTS	→ POSITION
sem	: POINTS	→ POSITION
track_support	: (TRAIN * TIME)	→ TRACKS

For a train k and an instant time t, we define the function *a(k,t)* which represent the geographical position of back axle of train k on the topology at instant time t.

Formalization of side collision is the following:

$\forall$ t $\in$ TIME, $\forall$ k, j $\in$ Train,
Side_collision (k,j,t) $\Leftrightarrow$
$\exists$ πi $\in$ POINTS,
pos_2_track(normal (πi)) = track_support (k,t)
$\wedge$ pos_2_track(reverse (πi)) = track_support (j,t)
$\wedge$ pos (normal (πi)) $\leq$ a(k,t) $\leq$ pos (sem(πi))
$\wedge$ pos (reverse (πi)) $\leq$ a(j,t) $\leq$ pos (sem(πi))
$\wedge$ $\mid$ a(k,t) - a(j,t) $\mid$ $\leq$ C(πi) (TRACK GAUGE)

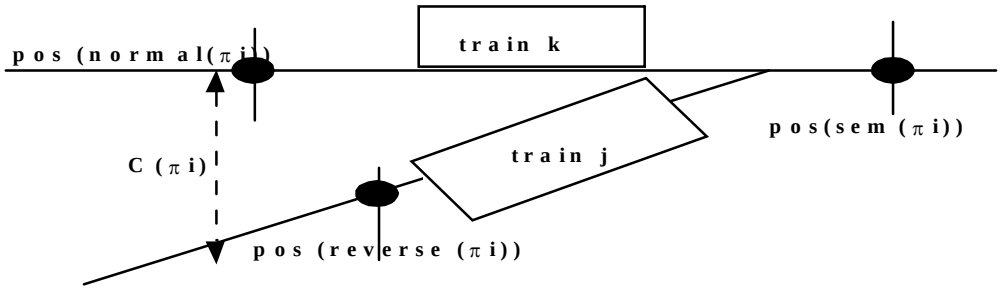


Fig.7: A side collision

The corresponding safety property is that the event *side_collision(k,j,t)* does not occur or may occur with a probability which is negligible < 10^{-9}.

3.4. Safety properties allocation onto system components

In this step, safety properties are distributed onto subsystem functions, so those safety kernel components are specified by their correctness properties. Allocation is based on a derivation process, according to system design (breakdown and principles). Once allocation is done, the whole set of basic properties, which have been identified, must be validated in terms of risk closure.

Restrictivity

Safety properties breakdown process relies on a "restrictivity assumption". We assume that every subsystem S_j (or component) has a restrictive state s_j, which is supposed to be reached by S_j in case of failure. This behavioral property is a very important constraint.
For every subsystem one must verify that:

$$\forall\, t \in \text{TIME},\ X(t) = (e1, e2, ..., sj, ...el) \text{ where } s_j \text{ is a "restrictive state" of subsystem } S_j$$

$$<=>$$

$$\exists \text{ command } (S_k, k \neq j) / X(t)> , t \in [0,\infty] \ \wedge \ \text{Pi is verified}$$

$$i \in [1,n]$$

This property means that S_j is "fail safe". For instance if the signaling fails (whatever the failure semantic is considered), all signals are considered to be red (not permissive) and all points are supposed to be uncontrolled. This restrictive assumption is done by other subsystems, which operate appropriate commands. Usually, the whole system reaches a passive safe state, through this assumption because all subsystems reach their respective restrictive states.

Safety properties identification and allocation

Vital Signaling

The vital signaling system implements signals, which are used for points protection. POINTS is the set of points controlled by the Vital Signaling subsystem. SIGNALS is the set of signals which are controlled by the Vital Signaling Subsystem. Any signal has two states: permissive (green) or not permissive (red). Let us introduce the function *green(s,t)* which returns true if the signal s at time t is green. For one point there are three signals, called *f_normal*, *f_reverse* and *f_sem*. Signaling subsystem must ensure that if the point is in normal position (resp reverse) only f_normal signal (resp f_reverse) is permissive, or that no signal is permissive. The state where all signals are red is a passive safe state for signaling.

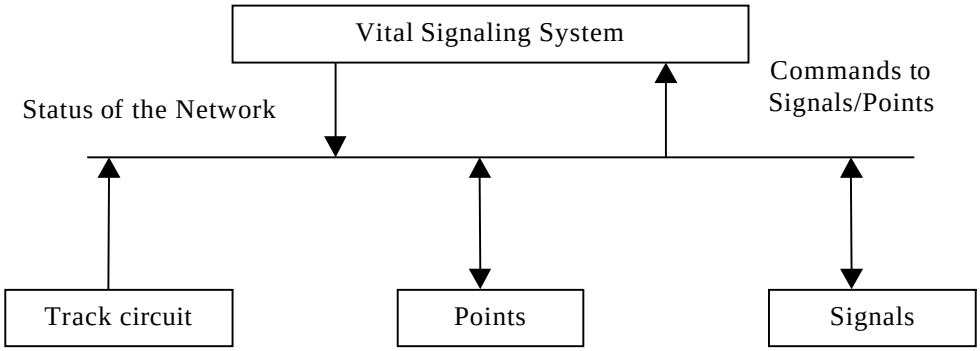


Fig.8: Architecture of Signaling Control.

At any time t and for all track circuit tr, (tr $\in$ TRACK). *free(tr,t)* is a boolean function which indicates if a train is detected or not, on the track circuit tr.

The corresponding safety property is described hereafter:

$\forall\, t \in \text{TIME}, \forall\, \pi i \in \text{POINTS}$

Normal_position $(\pi i, t) \Rightarrow \quad \neg(\ \text{green}(\text{f\_normal}(\pi i), t) \wedge\ \text{green}(\text{f\_sem}(\pi i),\ t)) \wedge \neg\ \text{green}(\text{f\_reverse}(\pi i), t)$

Reverse_position $(\pi i, t) \Rightarrow \quad \neg(\ \text{green}(\text{f\_reverse}(\pi i), t) \wedge \quad \text{green}(\text{f\_sem}(\pi i), t)) \ \wedge \neg\text{green}(\text{f\_normal}(\pi i), t)$

$\vee \quad (\ \neg\text{green}(\text{f\_normal}(\pi i), t) \ \wedge \neg\text{green}(\text{f\_reverse}(\pi i), t) \wedge \neg\text{green}(\text{f\_sem}(\pi i), t))$

$\forall\, t \in \text{TIME}, \forall\, \pi i \in \text{POINTS}, s \in \{\ \text{f\_reverse}(\pi i), \text{f\_sem}(\pi i), \text{f\_normal}(\pi i)\}$

$\text{green}(s,t) = \text{True} \Rightarrow \exists\, \sigma \in \text{TIME}, \forall\, t' \in [t-\sigma\ ,\ t] \mid \text{green}(s,t') = \text{False} \wedge \text{free}(\pi i, t') = \text{True}$

Civil Engineering

Signals are implemented in such a spacial position, that side collision is impossible if the spacial positions of trains are in respect with signals:

$\forall t \in TIME , \forall k, j \in TRAINS, \forall \pi i \in POINTS,$

$a(k,t) \leq pos(f\_normal(\pi i)) \vee a(j,t) \leq pos(f\_reverse(\pi i)) \Rightarrow \neg (side\_collision(k,j))$

Ground ATP

Few functions are necessary for this case study. In the following, we describe only the properties, which must be satisfied by these functions.

Ground localisation

Train localization is performed by ground ATP. At any time t and for any train k, train localization function computes *E_ground(loc(k,t)) and E_ground(direction(k,t))*. Properties of this function are localization freshness and alarm setting if the localization cannot be computed.

$\forall k \in TRAIN, \forall t \in TIME,$

$E\_ground(loc(k,t)) = nil \Rightarrow Alarm\_localisation(t) = true$

Next function

Trains relative positions are also managed by ground ATP. Let's note **Next** the function which associates to train k, the identity of the nearest train in a given direction: Next : (TRAIN, DIRECTION, TIME) -> TRAIN

$\forall t \in TIME, \forall k \in TRAIN,$

$\sigma = E\_ground(direction(k,t)),\ Next(k,\sigma,t) = n, \neg\exists j \in TRAIN \mid (n \neq j) \wedge (a(k,t) < a(j,t) < a(n,t),t))$

Target function

Third function is the **target** function, which associates to any train k, at each time t, the maximal position that can be reached by the train. This position depends on trains position (Next function), signals states (permissive or not), obstacles existence.

$\forall t \in TIME, \forall k \in TRAIN , Target(k,t) < \min_{x \in Next(k, E\_ground(direction(k,t)),t)} \{ |a(x,t) - a(k,t)| \}$

On board ATP

We have identified two safety functions and related properties useful for side collision avoidance: the first one is the train localization loc(k,t) and the second one is the speed control function. Properties are the following: for the localization function, either the localization is fresh and precise enough or the emergency braking command is set; the speed control safety function monitors the acceleration control variable γ, ensuring that the target cannot be passed or that the emergency braking command is set.

Localization function

$\forall t \in TIME , \forall k \in TRAIN,$

$loc(k,t) \Rightarrow$

$\{$

$\quad | E\_board(a(k,t)) - a(k,t) | \leq \varepsilon a$

$\wedge \quad | E\_board(speed(k,t)) - speed(k,t) | \leq \varepsilon s$

$\wedge \quad | E\_board(direction(k,t)) - direction(k,t) | = 0$

$\}$

$\vee \quad Alarm\_Emergency(k,t) = true$

Speed Control Function

In a system representing train speeds in miles per hour, a sub range type SPEED = 0..Max_speed is introduced.

$\forall$ t $\in$TIME , $\forall$k $\in$ TRAIN,
1/2 (γt^2) + speed(k,t)t < Target (k,t) - a(k,t)
$\vee$ Alarm_Emergency(k,t) = true

where γ is set by the functional ATP and |Gam_Emergency| > γ.

Rolling Stock

Rolling stock applies commands set by on board ATP. In respect of collision avoidance the only one function is speed application. The safety property to be specified by Rolling stock is:

$\forall$ t $\in$TIME , $\forall$k $\in$ TRAIN,
Alarm_Emergency (k,t)
=> $\exists$ $\tau\in$TIME | speed (k, t+τ) =0

Interfaces properties

These are directly derived from the restrictivity assumption.

Vital Signaling

Vital signaling is a fail safe subsystem which ensures that points positions and signals physical states are always compatible.

Degraded states are managed as follows:

- If the control position is not known (or not fresh enough), it is supposed to be uncontrolled,
- When a signal fails its physical state is set to "not permissive state" (red).

Civil Engineering

Topological parameters must be correct, that is compliant with geographical implementation and with safety analysis.

ATP subsystem

For the ATP subsystem (both ground and on board components), we have identified the following set of properties:

- If a train k is near a point area, and if a point is uncontrolled in this area, *target (k)* < points control position.
- If the location of a train is not known, (e.g. message is not received or is incorrect), ground ATP sets *E_ground (loc(k,t))* to nil value.
- If *target(k,t)* is not known (e.g. message is not received or not correct), on board ATP sets it to nil value.

Timing constraints

They are part of safety properties. End to end properties specify the elapsed time between the occurrence of an event and the entrance into a safe state.

Risk closure

Safety properties must be validated, using semi formal or formal methods. The demonstration relies on complementary techniques:

- Logical trees in order to state the demonstration scheme; in particular goals and sub-goals must be identified, at least informally.
- States diagrams and behavioral properties analysis (temporal logic, liveness, timing constraints).
- Formal proof.

Hereafter is the framework of the demonstration. The root property is broken down in a logical combination of properties. But the formal description of properties and the actual demonstration of

risk closure are in progress. The statement to be demonstrated is "For a given point πi, and for any instant time there can be only one train on the *track_support* which contains the point".

From Civil Engineering properties, signals implementations are compliant with points positions and trains gauge constraints. Signals implementations ensure that if trains are behind the signals, there is no collision between them. This is a static property. From Vital Signaling property, we can say that at most one signal is permissive so that only one train can enter the point area. While the permissive signal remains permissive, other signals are not permissive.

ATP estimates the localization of train, its direction, and ground ATP asks for routes using pointπi. We can suppose two trains ask for convergent routes, since vital signaling ensures only one route may be opened. If the localization of a train is not fresh enough, the train is supposed to be lost and all trains are stopped in a bounded amount of time.

If trains are localized at time t, targets are allocated to trains (in a bounded amount of time) and these targets are compliant with signals physical states (states as known at time t). So that, either trains are all stopped or only one train has a target beyond the permissive signal in the point area, which is considered. Other trains targets are in respect of red signals. Any train which is localized at time t is controlled so that the target cannot be overtaken (speed control function). This implies that trains movements are compliant with targets. So that only one train enters the point area.

4. Formal Process for safety critical systems

In this paper, we address two questions:

- It is possible to design a system so that the safety property demonstration is traceable throughout the whole design process?
- To which extent can formal methods (model checking, formal proof) be applied for the safety property verification in the context of large and complex systems?

Our approach is exploratory and, since our work is still going on, we have only partial results. Nevertheless, we pointed out that it is possible to design a system so that only a few components are concerned with safety, and that only safety properties are to be proved.

The safety demonstration we propose is organized in five steps, throughout the system design process:

- In the system analysis stages, risks and safety properties are identified. Passive safe states must be identified and the system behavior is analyzed so that it is proved that all risks are covered by the safety properties and related passive safe states.
- Safety properties are allocated onto subsystems, and components according to design principles. In this step we have to prove that system safety properties are implied by subsystems properties and by interface properties. Interface properties rely on restrictivity assumption.
- Third step is the formal specification of the safety functions for each subsystem. It defines safety components. This step is validated if safety properties are implied by safety functions correctness.
- Fourth step demonstrates that the safety functions properties remain true for all environment operations and in case of failures.
- Fifth step is the implementation of the safety functions. Safety component development relies on various "safety technologies": formal methods for software fail safe technologies for hardware.

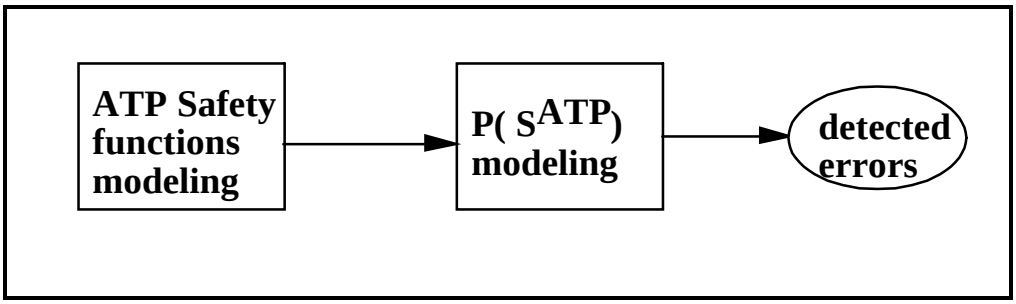
Our case study will focus on ATP subsystem specification and verification, for side collision avoidance. In practice, our future work will cover steps 2 to 4 of the process:

- Formal description and verification of the safety properties identified in section 3.

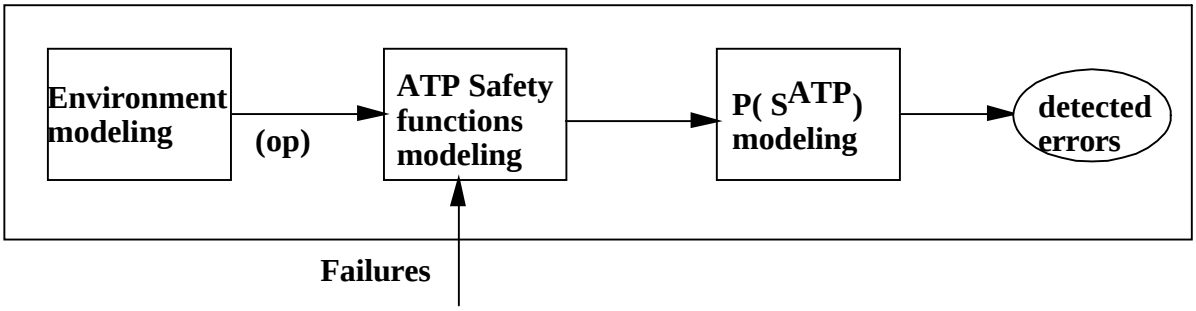
$$\forall \text{ Pi}, \quad \bigwedge_{S^j \text{ Subsystem}} P(S^j) \quad \Rightarrow \text{Pi}$$

  In particular, we shall identify $P(S^{ATP})$, the set of safety properties to be satisfied by the ATP subsystem,

- Formal specification of the subsystem ATP. The related verification activity is to demonstrate that formal specification properties => P(S^{ATP}). Whatever the modeling technique, which will be use, the safety properties must be included in the model so that verification or proof can evaluate safety properties.



- In the fourth step of the process, we shall demonstrate that ATP is a kernel. We shall take into account environment operations (degraded modes for instance) and failures in the model.



The method describes in this paper was capable to enable a safety evaluation of the ATP at ITSEC ([ITSEC 90] and [ITSEC 91]) assurance-correcteness level E4 (or E5 or E6) requirements.

References

[Abrial 96] J.R Abrial, "The B Book – Assigning programs to meanings", Cambridge University Press, ISBN 0-521-49619-5, September 1996.

[Auclair 96] J.P. Auclair, F. Baranowski, P. Caspi, S. Natkin, "Principes d'organisation pour la réalisation de logiciels critiques", AFCET 1996.

[Boulanger 95] H. Waeselynck, J-L. Boulanger, The Role of Testing in the B Formal Development Process, ISSRE 95, Toulouse October 1995

[Boulanger 97] Boulanger Jean-Louis - Delebarre Véronique - Natkin Stéphane. "Validation des spécifications et génération de tests de sécurité s'appuyant sur un modèle formel d'un système de transport ferroviaire", Décembre 1997, Rapport de recherche CEDRIC No 97-17.

[Behm 93] P. Behm, “Application d'une méthode formelle aux logiciels sécuritaires ferroviaires”, Atelier Logiciel Temps Réel, 17-19 Novembre 1993.

[Delebarre 96] V.Delebarre, S. Natkin, "Conception et evaluation de logiciels critiques: De la sécurité logique à la sécurité physique", Congrès sur l’application des méthodes formelles, ENSIMAG, Grenoble, Janvier 1996.

[Guihot 90] G. Guihot, C. Hennebert, "SACEM software validation", in Proc. 12th IEEE-ACM International Conference on Software Engineering, March 1990.

[ITSEC 90] E. E. Community, "Information Technology Security Evaluation Criteria (ITSEC)", Technical report, 1990.

[ITSEC 91] “Information Technology Security Evaluation Criteria”, CEC Publication, June 1991.

[ITSEM 93] “Information Technology Security Evaluation Manual”, CEC Publication, September 1993.

[MAIRE 93] A. Maire, "Présentation du système MAGGALY", Symposium international sur l'innovation technologique dans les transports guidés, ITIG'93, Lille, septembre 1993.

[Rushby 89] J. Rushby, "Kernel for safety", in Safe and Secure Computing Systems, Blackwell Scientific Publications, London, 1989.

[Rushby 93] J. Rushby, "Formal methods and the certification of critical systems", SRI Research report, November 1993.

GLOSSARY

ATC	:	Automatic Train Control
ATP	:	Automatic Train Protection
MAGGALY	:	Métro Automatique à Grand GAbarit de la région LYonnaise.