CLEARSY
SYSTEM ENGINEERING

Métro de New York
CBTC Flushing

Vérification formelle système

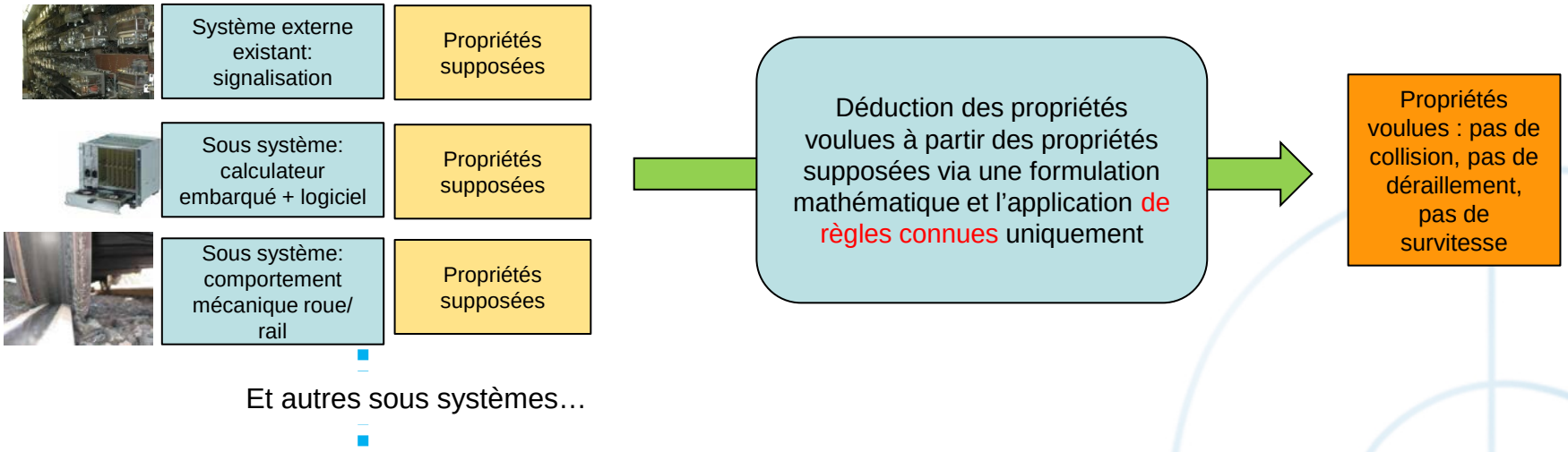
Auteurs:
Denis SABATIER
Lilian BURDY

Paris
Lyon
Aix

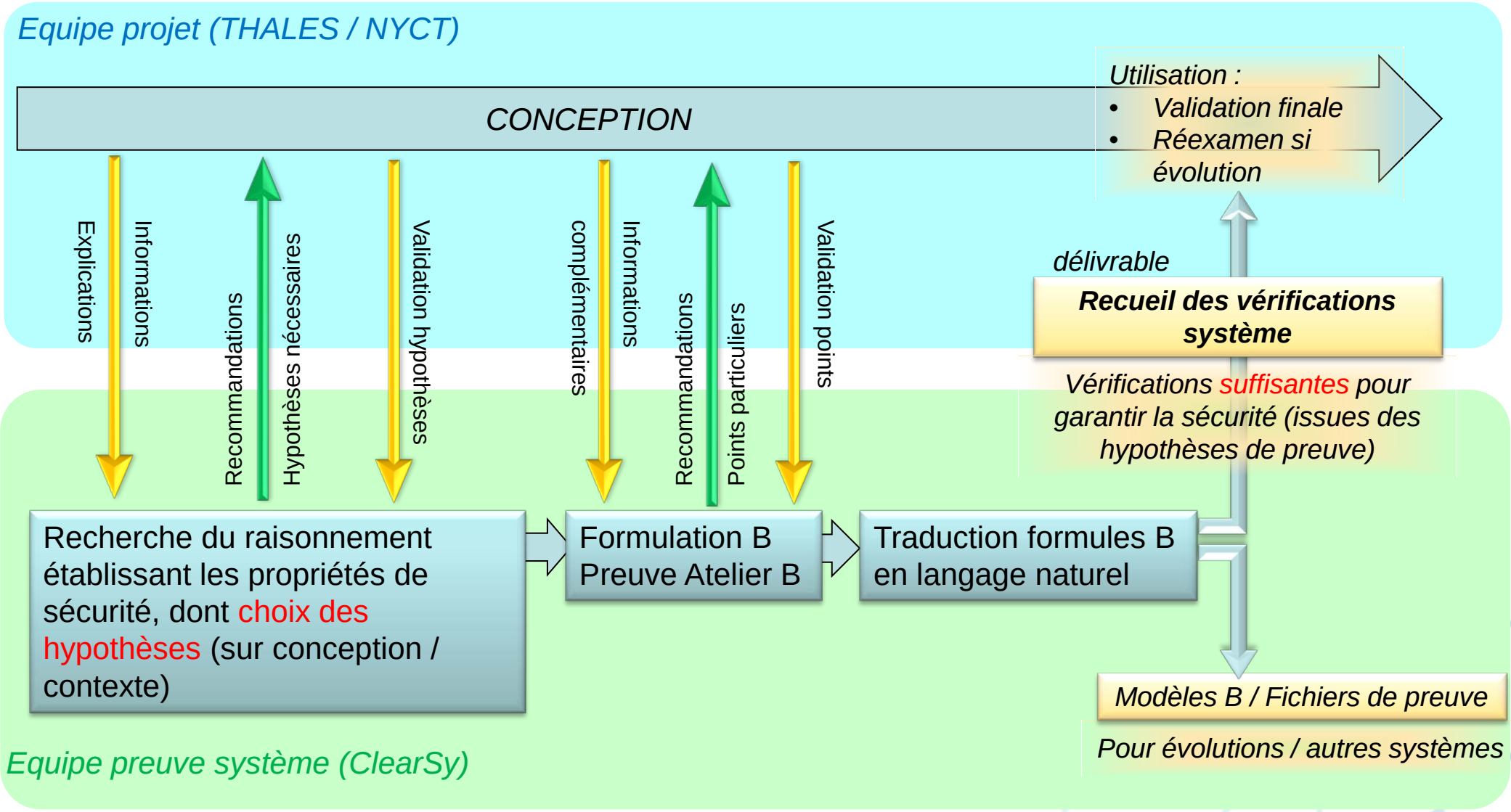
WWW.CLEARSY.COM

Qu'est qu'une « vérification formelle système »

- Cela consiste à démontrer les propriétés voulues d'un système à partir d'hypothèses et de règles bien définies
  - Niveau système : car les sous-systèmes sont représentés par des propriétés prises comme hypothèses
  - Formelle : car le raisonnement établissant les propriétés voulues à partir de ces propriétés de sous-systèmes ne devra employer que des règles mathématiques
  - Vérification : « building the system right » (la validation correspondant plutôt à un jugement, « building the right system »)

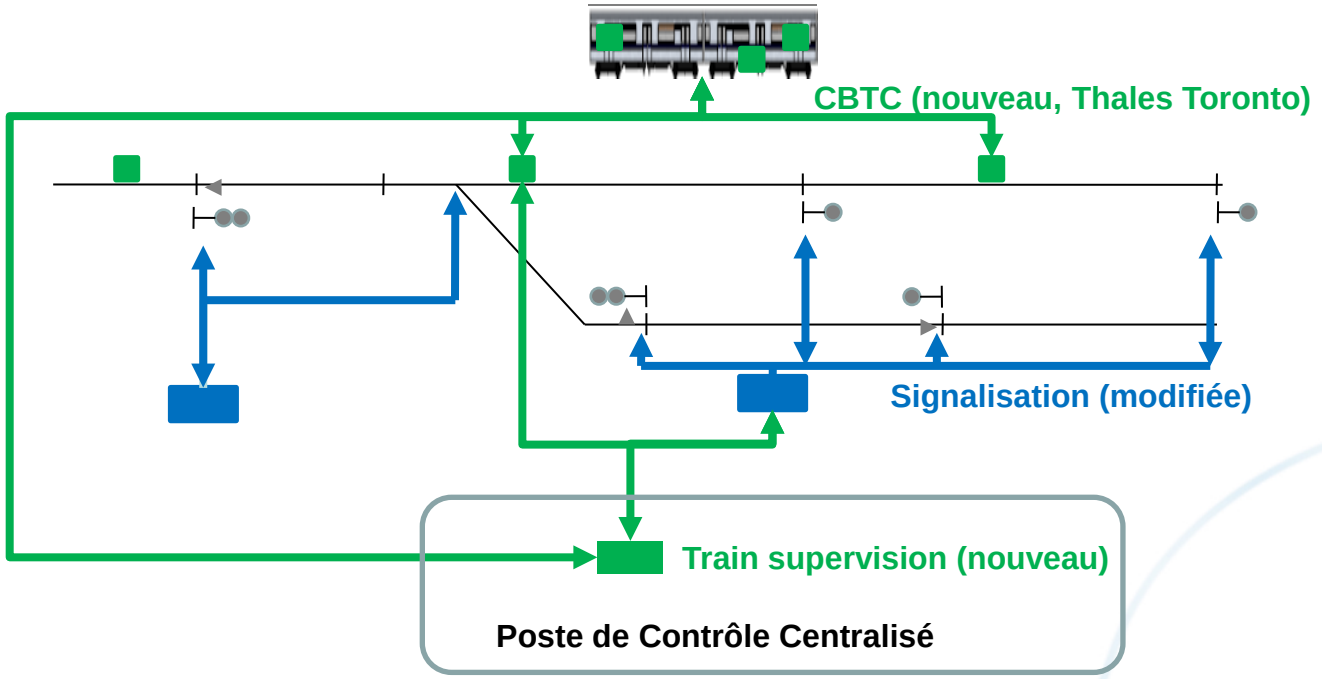


Vérification formelle système : processus employé sur Flushing

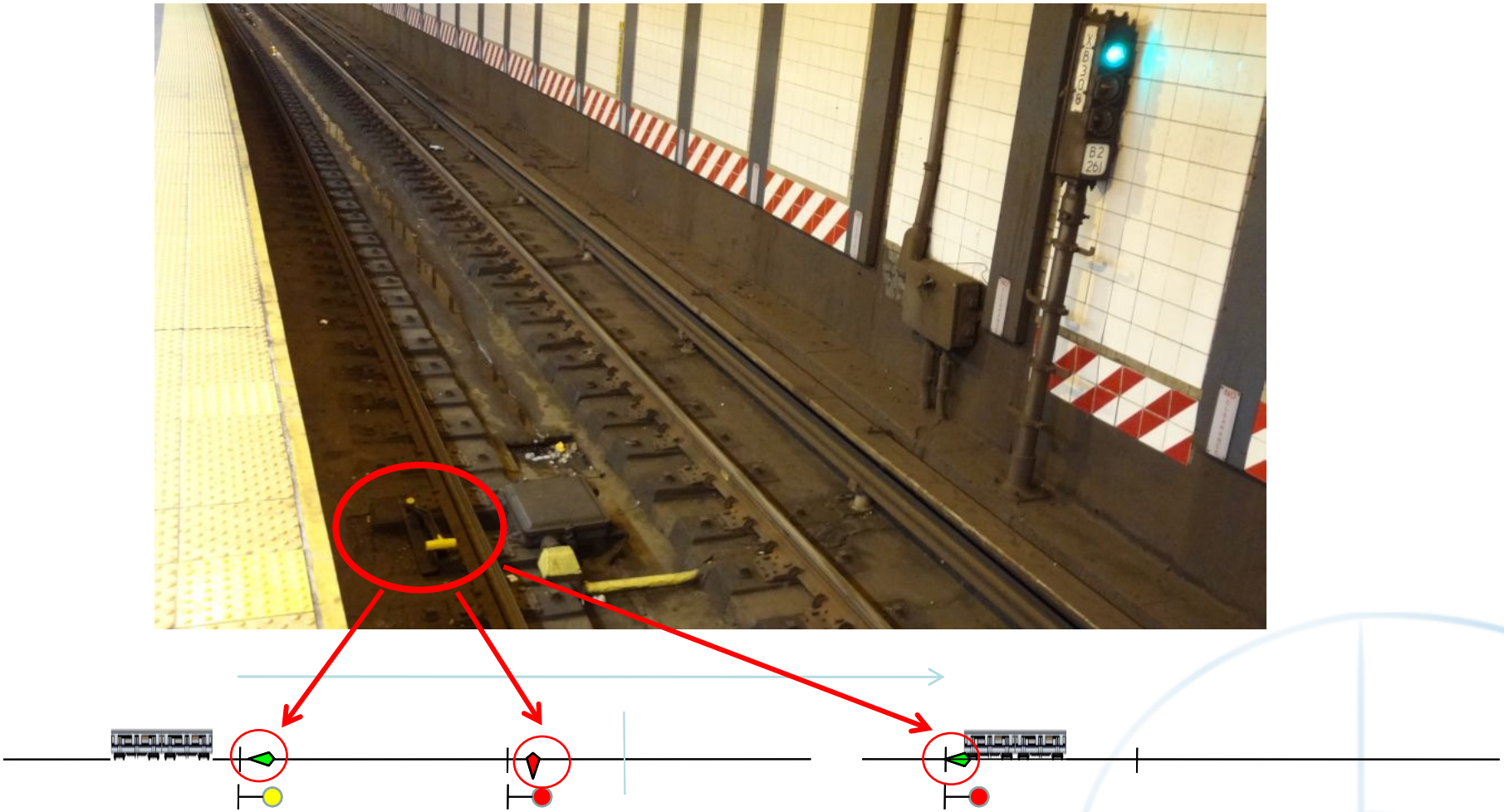


Line 7 CBTC: rôle & architecture

- CBTC = communication based train control
  - Un système avec des calculateurs au sol et embarqués
  - Qui pilote les trains (conducteur néanmoins présent)
  - Interfacé avec la signalisation (le système qui pilote les aiguilles / signaux)

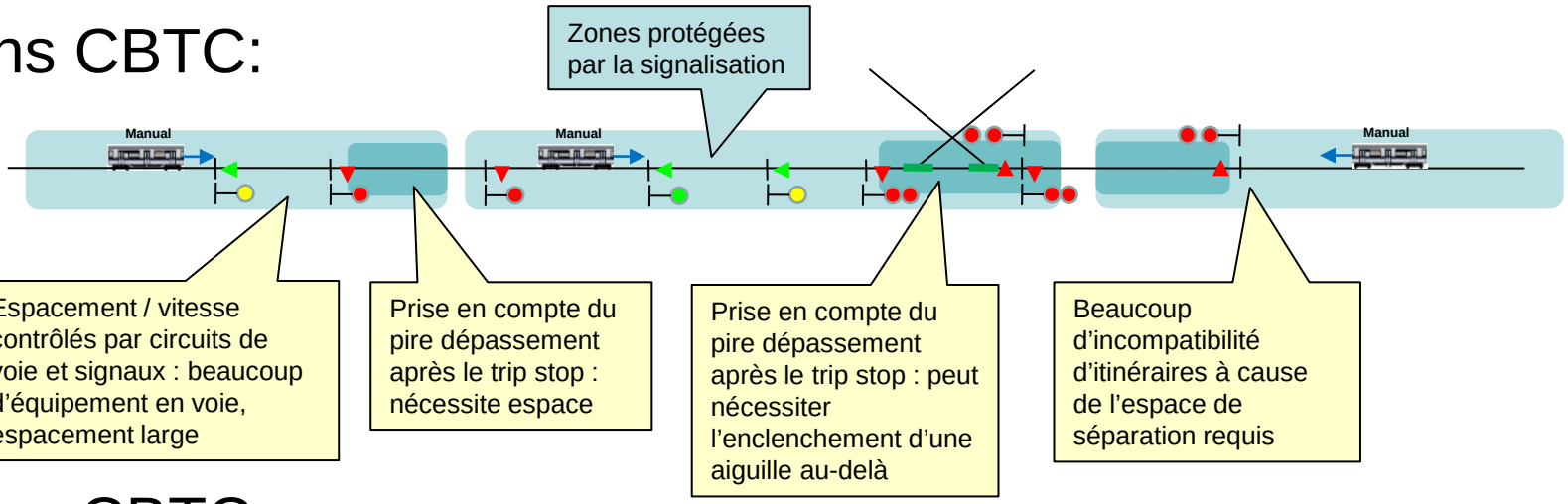


Une spécificité à New York: trip stop

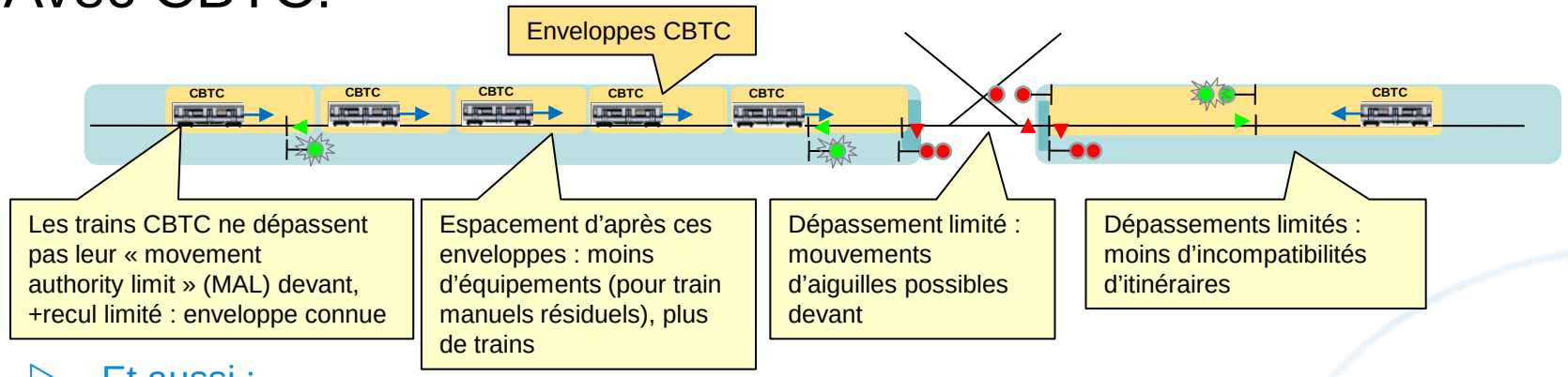


Line 7 CBTC: plus de possibilités avec la sécurité garantie

Sans CBTC:



Avec CBTC:



- Et aussi :
  - Possibilité d'insérer des signaux virtuels n'importe où pour manœuvres exceptionnelles, etc.
- Evidemment, les trains manuels doivent devenir minoritaires…

Line 7 CBTC : localisation des trains

- Les trains CBTC doivent déterminer et transmettre leur position:
  - Grâce à des balises de localisation disposées sur la voie
    - À maîtriser: portée des balises, délais, précision, diaphonies, installation & maintenance…
    - Pour une relocalisation d'un train délocalisé: il faut déterminer l'orientation
  - Grâce à la mesure du mouvement entre les balises
    - Capteurs de mouvement (roues phoniques, accéléromètres, attention aux glissements !)
      - Sur Flushing: 1 essieu libre et 1 essieu freiné, équipés de roues phoniques
      - Accéléromètres pour déterminer les glissements
    - Précision fondamentale pour performances, Connaissance de la précision fondamentale **pour la sécurité**
  - Grâce à la carte de la voie
    - Position des balises
    - Etat des aiguilles (reçu)
- Les trains CBTC doivent déterminer leur vitesse
- Communications radio

Line 7 CBTC : contrôle d'énergie

- Les trains CBTC garantissent une « movement authority limit » (MAL) devant :
  - Transmise par les ZC (Zone Controller)
  - Quand un train a accepté un MAL : il ne doit jamais le dépasser
    - Tant qu'aucun MAL au-delà ne le remplace
- Contrôle d'énergie : prédiction sécuritaire du freinage en pire cas
  - Ainsi les trains déclenchent le freinage d'urgence alors que celui-ci les arrêtera encore avant le MAL
  - Mais ne pas déclencher trop tôt : fondamental pour de bonne performances !
- Principes :
  - Décélération minimale garantie sur voie à plat (avec pires pannes sur les freins, pires conditions d'adhérence)
  - A partir de la détermination sécuritaire de la position et de la vitesse
    - Calcul des distances et **des pentes**
    - **Les pentes** sont très importantes (elles peuvent doubler la distance d'arrêt)
    - Utilisation de lois mécaniques pour prédire le freinage en pente à partir du freinage sur plat
    - Attention à l'énergie cinétique stockée dans les masses tournantes
    - La masse des passagers n'est pas connue
  - Prise en compte du délai d'établissement du freinage d'urgence
    - Phase d'accélération résiduelle, phase de coasting (et **pentes** pendant ces phases…)
  - *Optimiser les performances tout en restant sécuritaire est l'enjeu ici…*

Propriétés de sécurité cibles choisies

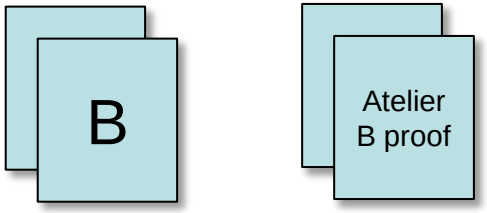
- La propriété principale choisie :
  - **À tout instant et pour chaque train, on peut définir une zone de protection PZ telle que :**
    - Le train est entièrement dans PZ, *et restera dans PZ grâce à son freinage seul si PZ n'évolue plus*
      - Ce dernier point est un peu subtil, voir plus loin…
    - PZ contient seulement des aiguilles verrouillées et pas d'autre obstacle que le train lui-même
    - Les PZs sont d'intersection vide deux à deux
- La propriété de « non-survitesse » est aussi requise (plus facile à formuler)
  - Pour les déraillements par survitesse
  - Et la preuve des PZ en aura besoin (dans la propriété « les trains restent dans leur PZ »)
- Nous avons ainsi quelque chose de bien défini à prouver
  - Si on réussi à définir de telles PZ à tout instant et dans tous les cas
    - En décrivant toutes les évolutions de PZ
    - De telle sorte de les propriétés précédentes soient conservées,
    - En s'appuyant sur des choses qui correspondent à la conception et aux conditions réelles
  - Alors OK.

Produits finaux obtenus

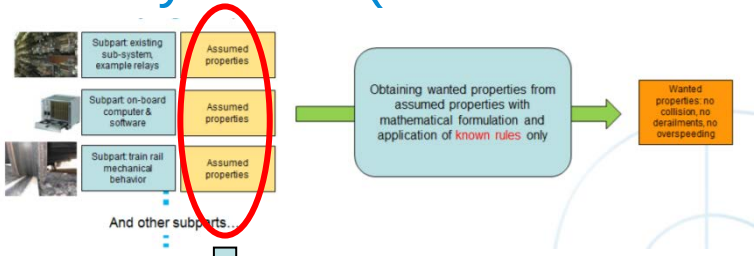
- A la fin du processus :

  Recueil des vérifications système (en Anglais)

Modèles B *Fichiers de preuve*



- Recueil des vérifications système (= recueil des hypothèses) : le produit principal



Explication en Anglais de **tout** ce qui a été nécessaire pour la preuve : de la conception du CBTC jusqu'aux lois mécaniques ou aux procédures d'exploitation…

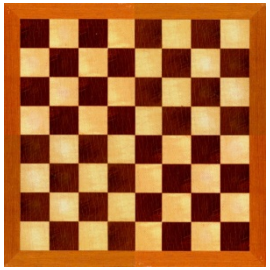
La preuve est vérifiable, même sans méthode formelle

- Très souvent : la conception et la sécurité sont “fermés”
  - S’appuyant sur des avis d’expert
    - Conclusions finales disponibles, mais raisonnement pas complètement disponible
  - Conception : les détails importants & et les « reasons why » sont souvent connus de peu de personnes
    - Avec une impossibilité de comprendre sans **tout** voir…
- L’idée ici : la preuve est vérifiable
  - Comme une preuve mathématique traditionnelle : « tout le monde peut la lire et personne n’a trouvé de faille »
    - Ici : la logique est très simple en général, ce sont les hypothèses qui sont importantes
      - Tout ce qui est nécessaire à la preuve est baptisé hypothèse…
  - En connaissant les hypothèses (grâce au recueil des hypothèses), avec des indications sur le mécanisme de raisonnement, le lecteur peut refaire la preuve dans son cheminement
    - Indications sur le mécanisme de raisonnement : proof path § dans le recueil des hypothèses
    - Avec l’Atelier-B : on obtient une validation de la formulation complète et de la preuve, mais cela n’empêche pas de chercher une preuve claire et facilement lisible

Raisonnement avec des règles et des hypothèses définies: le damier de Dijkstra

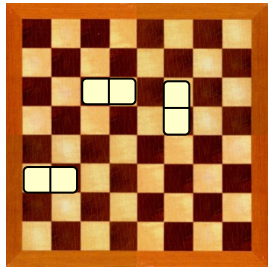
- L'exemple célèbre de Dijkstra :
  - Conception:

Un damier



Des dominos

Un placement des dominos alignés avec les cases

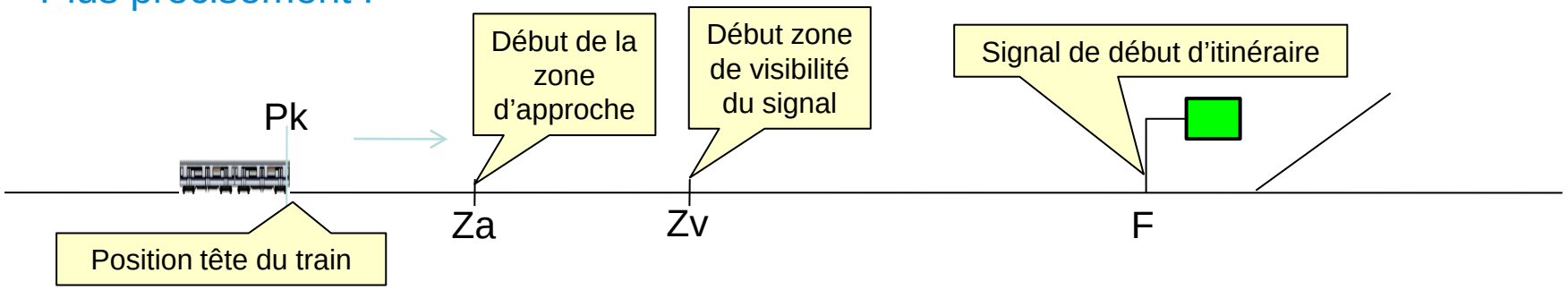


  - Propriété : les dominos ne couvriront jamais tout le damier sauf exactement la case bas / gauche et la case haut / droit

  - La clef : si B et W sont le nombre de cases black / white non couvertes, B = W tout le temps
    - *Car placer chaque domino consomme toujours un blanc et un noir*
    - Donc atteindre un état où W = 2 et B = 0 est impossible
- Ce n'est pas un avis d'expert fermé…
  - C'est parce que l'unique opération produit B:=B-1, W:=W-1 ce qui conserve trivialement B = W
  - Raisonnement mathématique très simple, tout est dans la formulation des hypothèses
    - Si on définit le damier / les dominos / la géométrie du placement avec les propriétés clefs (évolutions B et W), évident !
  - C'est bien quelque chose que tout le monde peut lire et vérifier (et qui ne procède pas par multitude de cas…)

Raisonnement avec des règles et des hypothèses définies: la destruction d'urgence

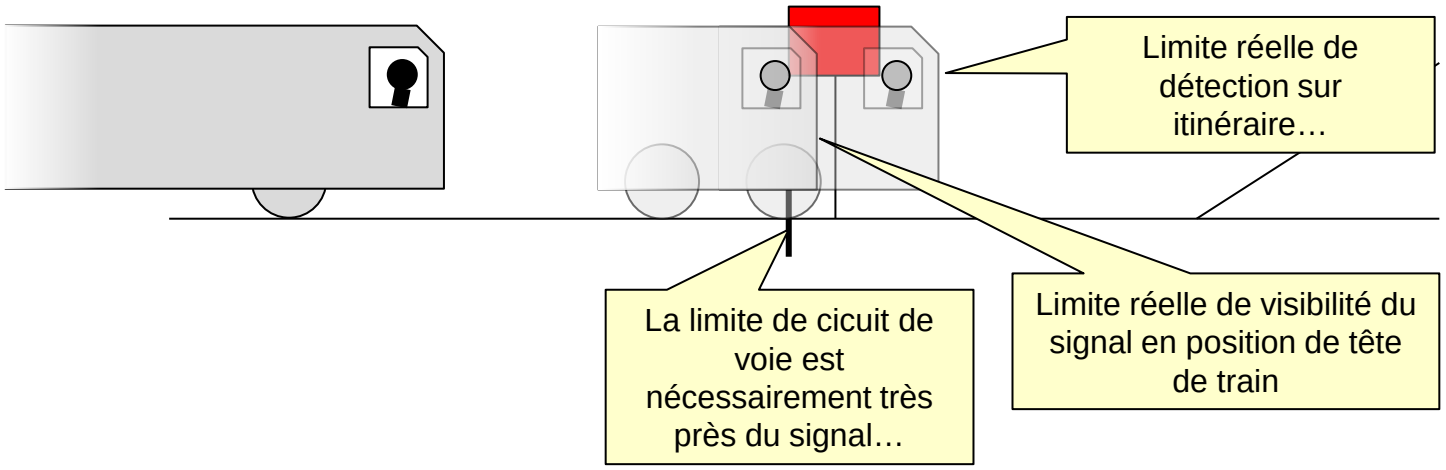
- Destruction d'urgence d'un itinéraire (sans trip stops / CBTC) :
  - Si la zone d'approche est occupée, attendre un délai T avant de désenclencher l'itinéraire (et les aiguilles)
  - Ceci garantit qu'un train ne sera jamais sur un itinéraire détruit (propriété cible)
- Plus précisément :



  - Hypothèses :
    - Si F est rouge et visible du train (Pk>Zv), le train stoppe en moins de Ts (délai) et Ds (distance). Cela inclut le temps de réaction du conducteur… Et le train reste arrêté ensuite.
      - Ts et Ds sont plus petits que T, Za ou Zv
    - Si le train est au-delà de F, l'itinéraire reste enclenché (train détecté)
    - Si le train est au-delà de Za, le délai de destruction d'urgence T est appliqué
    - Le train arrive de la gauche et avance sans sauts (Pk est continu et croissant)
  - Propriété : si Pk>F, alors l'itinéraire est resté enclenché
- Le raisonnement est maintenant possible… Et simple !

Raisonnement avec des règles et des hypothèses définies: la destruction d'urgence

- Une fois formulées (hypothèses / propriétés cibles), les choses semblent simples…
- Mais il faut examiner avec attention le sens des hypothèses dans la réalité :



  - L'hypothèse « itinéraire conservé si Pk>F » est légèrement fausse (en fait : si Pk>F+delta1)
  - L'hypothèse « la visibilité du signal va jusqu'à F » est légèrement fausse (en fait : fin à F-delta2)
  - Problème si destruction lorsque le train reste entre F-delta2, F+delta1
    - Certains CBTC délocalisent un train qui reste trop longtemps dans cette position…
- Les hypothèses formulées et bien définies peuvent et doivent être confrontées à la réalité
  - Grâce à la précision de leur définition

Raisonnement avec des règles et des hypothèses définies: la destruction d'urgence

- En fait, prouver « si Pk>F l'itinéraire est conservé » est très simple (hormis le piège précédent):
  - En suppose partir d'un état où le signal est vert et le train avant Za, Zv
  - Soit t0 l'instant de destruction
  - Si le train est avant Zv à t0, le train stoppe avant Zv+Ds (avant le signal), Pk ne sera jamais plus grand que F
  - Si à t0 le train est après Zv: le train s'arrête avant t0+Ts, donc avant t0+T (itinéraire pas encore détruit)
    - Si le train est stoppé après F: itinéraire jamais détruit
    - Si le train est stoppé avant F: Pk ne dépasse jamais F
- On n'utilise pas de mathématiques complexes
  - Bien qu'on emploie les outils formels pour s'obliger à une définition totalement formelle et pour garantir la preuve
  - Si des mathématiques complexes sont nécessaires :
    - Généralement : signifie qu'on tente de re-prouver les résultats scientifiques employés dans la conception… => Non.
- L'action la plus importante : exiger que les propriétés soient obtenues d'hypothèses bien définies via des règles logiques uniquement conduit à :
  - Des hypothèses bien définies (donc vérifiables)
  - Un "know-why" connu et maîtrisé

Différentes sortes d'hypothèses

- Hypothèses sur la conception du CBTC:
  - Hypothèses sur la conception Software
  - Hypothèses sur la conception Hardware
- Hypothèses de contexte : tout le reste
  - Hypothèses sur les systèmes externes (exemple : la signalisation)
    - Hypothèses sur les **propriétés de comportement global** seulement
      - Ces propriétés pourraient être prouvées, mais seulement en rentrant dans la conception du système externe
  - Hypothèses sur le comportement des trains ou des personnes
    - Conséquences de lois physiques et de probabilités
      - Par exemple : les deux essieux équipés de roues phoniques ne glissent pas ensemble
        - Car 1 libre, 1 seulement freiné. OK, mais…
        - Preuve faite dans cette hypothèse (même si le CBTC traite ce cas…)
  - Lois physiques connues
    - Introduites dans la preuve en tant qu'hypothèses
- *Dans ce contexte tout est baptisé « hypothèse »…*

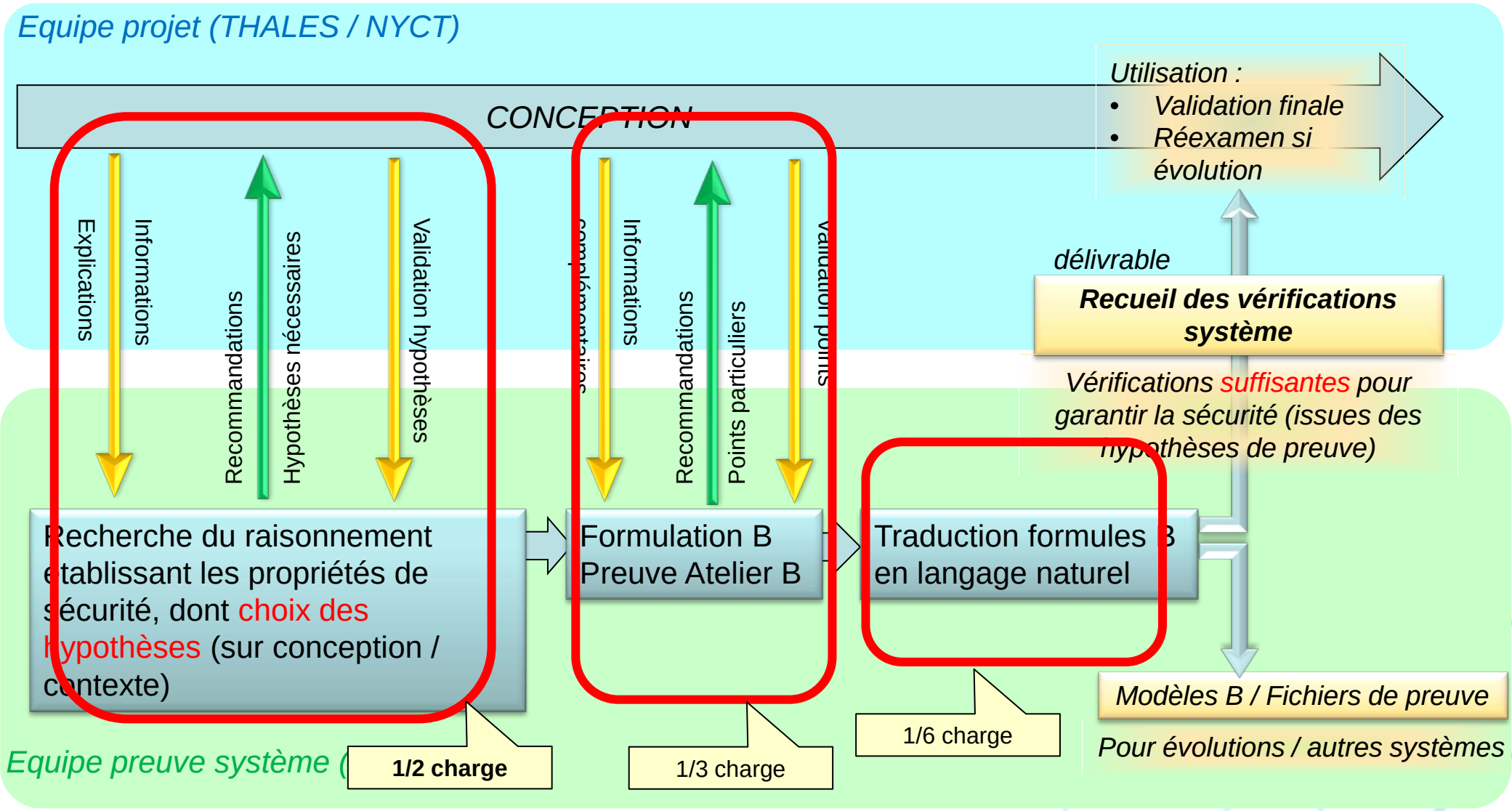
Méthode : choix des hypothèses

- Faire le choix des hypothèses et trouver le raisonnement de preuve sont deux processus très liés
  - Hypothèses réalistes correspondant à la conception et aux conditions réelles : savoir d'expert
    - Dans l'exemple précédent : zone de visibilité / zone de détection
  - Comprendre « pourquoi ça marche » c'est refaire le raisonnement du concepteur (à nouveau : savoir d'expert)
    - Exemple: dimensionnement Za, Zv et T
- La communication avec les experts est donc fondamentale
  - Ne pas réinventer
  - L'équipe de preuve doit apporter rigueur et bonne définition à des éléments existants

Trouver pourquoi ça marche : méthodologie

- Méthodologie pour « trouver le bon raisonnement » :
  - “animer” le système dans des scénarios, en cherchant à briser la propriété (dans notre exemple : en cherchant les collisions de trains)
    - Résoudre les détails manquants pour ces animations
      - On sélectionne ainsi les détails qui comptent (sans toute la complexité de conception)
  - Comprendre pourquoi on n’arrive pas aux collisions
    - Trouver les propriétés intermédiaires qui les interdisent
    - Et les hypothèses pour éliminer les scénarios de collisions considérés comme irréalistes
  - Démontrer ces propriétés intermédiaires menant aux propriétés voulues
- **Cela nécessite une phase de « preuve en langage naturel »**
  - Dans laquelle la priorité est à la communication experts / concepteurs
    - Pour trouver le bon raisonnement
    - Pour trouver les hypothèses réalistes
    - Dans le but de formuler mais sans y passer du temps à cette étape
- Nota : la méthode formelle est utilisée pour s’obliger à tout formuler et garantir le raisonnement, *et non pas pour trouver le bon raisonnement*

Global méthodologie globale



« how-to » pour la phase de preuve en langage naturel

- Ne pas essayer de lire toute la documentation d'abord
  - Exemple: expliqué en terme de schémas à relais / noms de relais, l'exemple de la destruction d'urgence peut être *très* compliqué…
- La communication avec les concepteurs est fondamentale
  - S'appuyer seulement sur les documents n'est pas assez rapide !
  - Et les hypothèses doivent être choisies avec les concepteurs
- Utiliser un format « léger » de documents temporaires pour communiquer
  - Schémas, textes courts
  - Réunions (téléconférences pour éviter les temps de déplacement)
    - Décrire une compréhension précise et demander confirmation est un processus très efficace
    - Même s'il faut confirmer par écrit ensuite, **le volume d'information échangé par oral est toujours bien plus important**
- *La volonté de construire une preuve formelle doit être la motivation*

L'apport de la phase de preuve en langage naturel

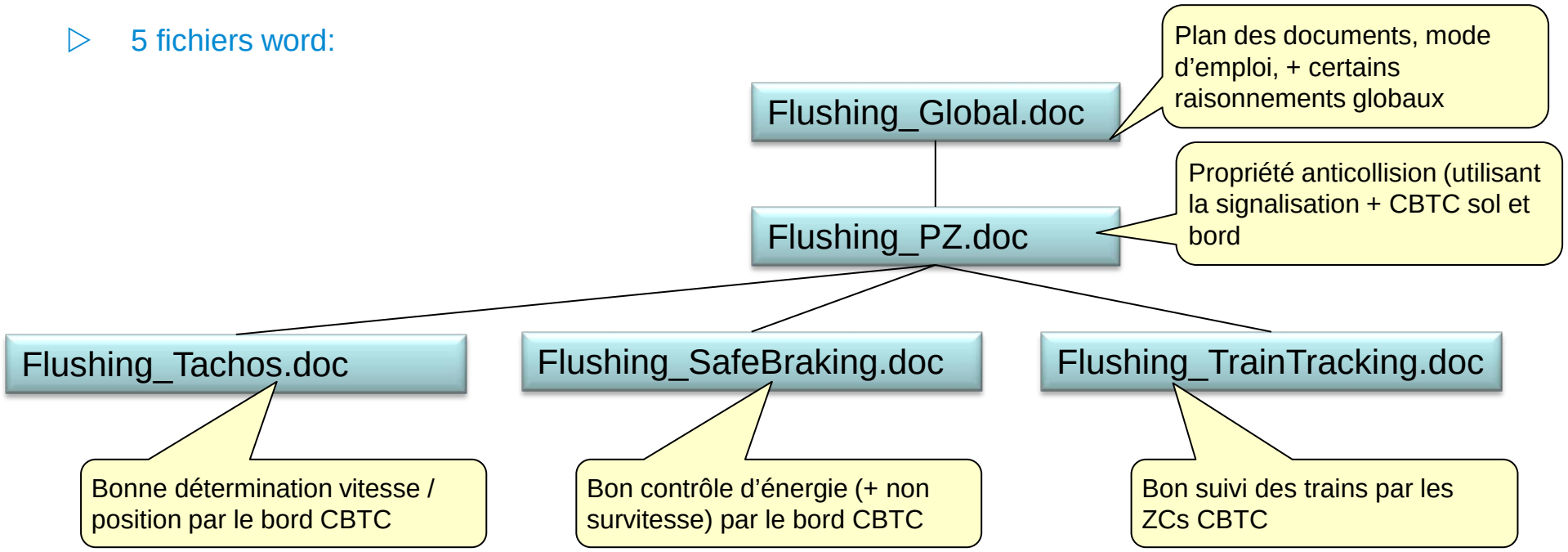
- Construire le raisonnement avec les concepteurs procure un **feedback immédiat**
  - Réunions de revue des raisonnements / hypothèses (téléconférences)
  - Tous les participants se familiarisent avec le raisonnement qui se dégage
    - On rassemble ainsi les experts CBTC, les experts de l'exploitant autour de sujets partagés
  - Les questions sur des hypothèses délicates / cas spéciaux sont connues très tôt
    - Avec suffisamment de temps pour les traiter correctement
- Le bénéfice d'un raisonnement partagé, basés sur des hypothèses explicitées apparaît à cette étape
  - Tôt dans le projet → meilleurs bénéfices
- On évite « l'effet tunnel »
    - L'effet où l'extérieur ne voit pas les travaux de preuve pendant trop longtemps
- Dans le projet Flushing : réunions ClearSy / NYCT / THALES
  - en moyenne 4h de téléconférence toutes les 2 semaines

Produits finaux

- A la fin du processus : on obtient le recueil de vérifications systèmes
  - Contenu principal : toutes les hypothèses
    - **Leur définition précise**
    - Qui peut valider chaque hypothèse (experts OBCU, experts interface roue/rail, etc.)
    - Comment fabriquer des tests et des vérifications pour chaque hypothèse, méthode:
      - Connecter les notions aux objets réels, à partir des explications données dans les documents
      - Voir les cas ou l'hypothèse tiendrait / ne tiendrait pas, employer la méthode de "l'exemple si l'hypothèse est fausse", voir les méthodes proposées et les notes dans les documents
      - En déduire les vérifications à faire sur l'appareil final
  - Utilisation :
    - Re-valider les hypothèses pour garantir les propriétés voulues
      - Après toutes les péripéties du projet
      - En cas d'évolution ou de changements
    - Valider ces hypothèses sur des systèmes similaires
      - Ou un sous-ensemble d'hypothèses correspondant à une sous-propriété
    - Comprendre comment les propriétés cibles sont garanties (rejeu en raisonnement manuel)
- On obtient aussi : Modèles formels B & fichiers de preuve
  - Leur emploi nécessite des connaissances B bien entendu…
  - Utilisation : après une évolution système ou un changement…

Flushing : le recueil des vérifications système

- 5 fichiers word:



- Dans chaque fichier (sauf Flushing_Global):
  - Proof targets § : énoncé des propriétés prouvées
  - Assumptions § : hypothèses nécessaires à la preuve (pour chaque propriété cible)
  - Sub-proofs § : propriétés supposées (pour chaque cible) qui font l'objet d'une sous-preuve ailleurs
  - Shared notions § : les choses que nous avons du définir pour exprimer les propriétés et les hypothèses
  - Proof path § : indications sur comment le prouveur de l'Atelier-B établit la propriété cible

La phase de modélisation formelle

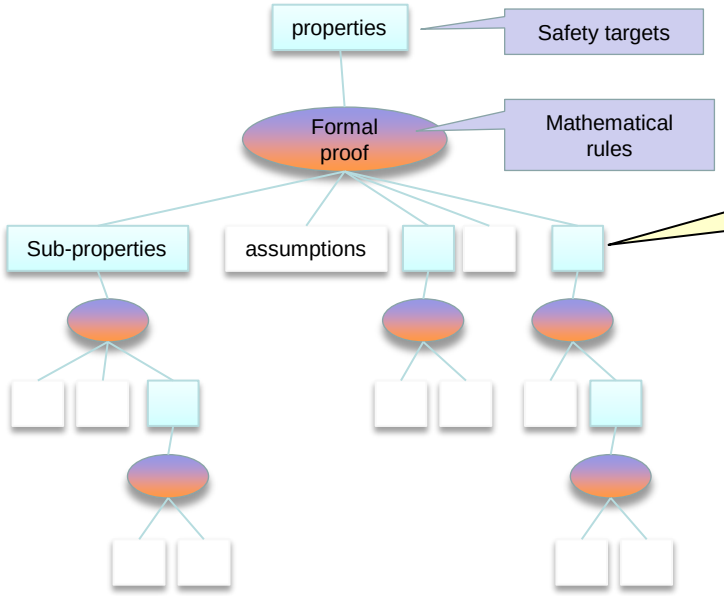
- Convertir le travail précédent en modèles B de sorte que la preuve de ces modèles soit **équivalente** aux raisonnements naturels
- Pourquoi est-ce nécessaire ?
  - ***On sait que c'est assez rigoureux pour être formalisé seulement si on l'a formalisé*** (même si la preuve en langage naturel cherchait à être la plus rigoureuse possible)
- Le recueil de vérifications système est issu des modèles B
  - Notre expérience: les hypothèses changent entre la forme après modélisation B et leur forme antérieure (obtenue pendant la phase de preuve en langage naturel)
- Modélisation B & preuve: en moyenne 1/3 de la charge totale constatée

Phase finale : rédaction, relectures

- Phase finale : à partir des modèles B prouvés, rédiger **le recueil des vérifications système** (ou recueil des hypothèses)
  - Document prévu pour une utilisation pratique
    - En particulier : pas de variables B et pas d'identifiants B dedans !
- Cette rédaction est à faire à la fin :
  - Une rédaction est moins chère que plusieurs…
    - Grâce aux échanges (phase en langage naturel), pas « d'effet tunnel »
- Relectures internes : essentiel
  - Nous avons une liste précise de points à vérifier :
    - Chaque notion doit être bien définie
      - Bien définie = sur n'importe quel scénario réel, l'interprétation doit être sans ambiguïté
    - Hypothèses : il faut contrôler comment elles peuvent être vérifiées (et par qui)
  - On fait ces relectures internes sur les modèles B d'abord
    - En se concentrant sur les liens entre les notions B et la réalité
- *Ensemble rédaction/ relectures : 1/6 de la charge totale*

Charge globale / niveau de détail

- Comment évaluer la charge globale d'une vérification formelle système sur un système donné ?
  - Sauf reprise, par définition le raisonnement est *inconnu au départ*
  - Ca dépend de la complexité de ce raisonnement
  - Et ça dépend du **niveau de détail**
- Le niveau de détail : une question fondamentale
  - Les propriétés cibles sont prouvées avec des hypothèses et des sous propriétés:



Le choix de ce qui est pris comme hypothèse ou comme sous-propriété (objet d'une sous-preuve) détermine le niveau de détail

Par exemple, sur le mécanisme de destruction d'urgence : si les propriétés de délai doivent être prouvées sur les **schémas de relais**, c'est un travail et un coût additionnel. Mais d'éventuelles erreurs permettant de court-circuiter le délai pourraient être trouvées…

  - Le niveau de détail doit être fixé par un critère clair

Choisir le niveau de détail

- Le choix du niveau de détail détermine ce qui est prouvé
  - Dans l'exemple précédent: la preuve de destruction d'urgence inclut / exclut les schémas de relais
    - Cela détermine si les mécanismes de relais seront couverts par la preuve
    - Cela détermine la forme des hypothèses de sortie obtenues
      - Si on exclut les relais : "la destruction d'itinéraire si l'approche est occupée déclenche le délai T…"
      - Si on inclut les relais : "tous les circuits de destruction doivent correspondre au schéma XXX…"
  - Evidemment : plus on descend bas, plus la validation des hypothèses est facile…
- Niveau de détail : un choix à faire avec le client
  - Se mettre d'accord sur un critère clair, se mettre d'accord sur chaque cas particulier ensuite
- Flushing : niveau système seulement, mais avec les algorithmes détaillés du CBTC
  - Incluant : (exemples)
    - comptage des impulsions des tachymètres (et particularités si changement de direction / glissements)
    - filtrages (Kalman) pour la mesure de vitesse
    - prise en compte des pentes pendant les phases d'établissement de freinage pour le modèle de freinage
    - communication bord / sol : détermination des dates de messages, croisements de messages, systèmes de timeout
    - échanges de suspicion de matériel non communicant entre contrôleurs de zones
    - dépassements des signaux par les trains manuels, enclenchements et provisions en cas de changements de modes ou retournements
    - destructions d'itinéraires et échanges filaires avec l'interlocking sur les approches.
  - Excluant :
    - Le code informatique (en particulier : on n'inclut pas la représentation informatique de la voie)
    - La conception des systèmes externes (par exemple les schémas relais de signalisation, toutefois employés pour en déduire les propriétés)

Un aperçu de la preuve Flushing

- Présentation des propriétés cibles et de leur rôle
- Présentation de la preuve “Protection Zones”
  - Pour comprendre le principe de raisonnement
- Décomposition de la preuve en sous-propriétés
- Pour chaque partie de la preuve :
  - Un aperçu des hypothèses et sous-propriétés employées
    - Pour une compréhension approximative
  - *Désolé, ce ne sera qu’un aperçu, sans réellement rejouer la preuve…*
    - *Ce qui a été fait en 4 jours devant les experts NYCT… Avec des détails confidentiels !*

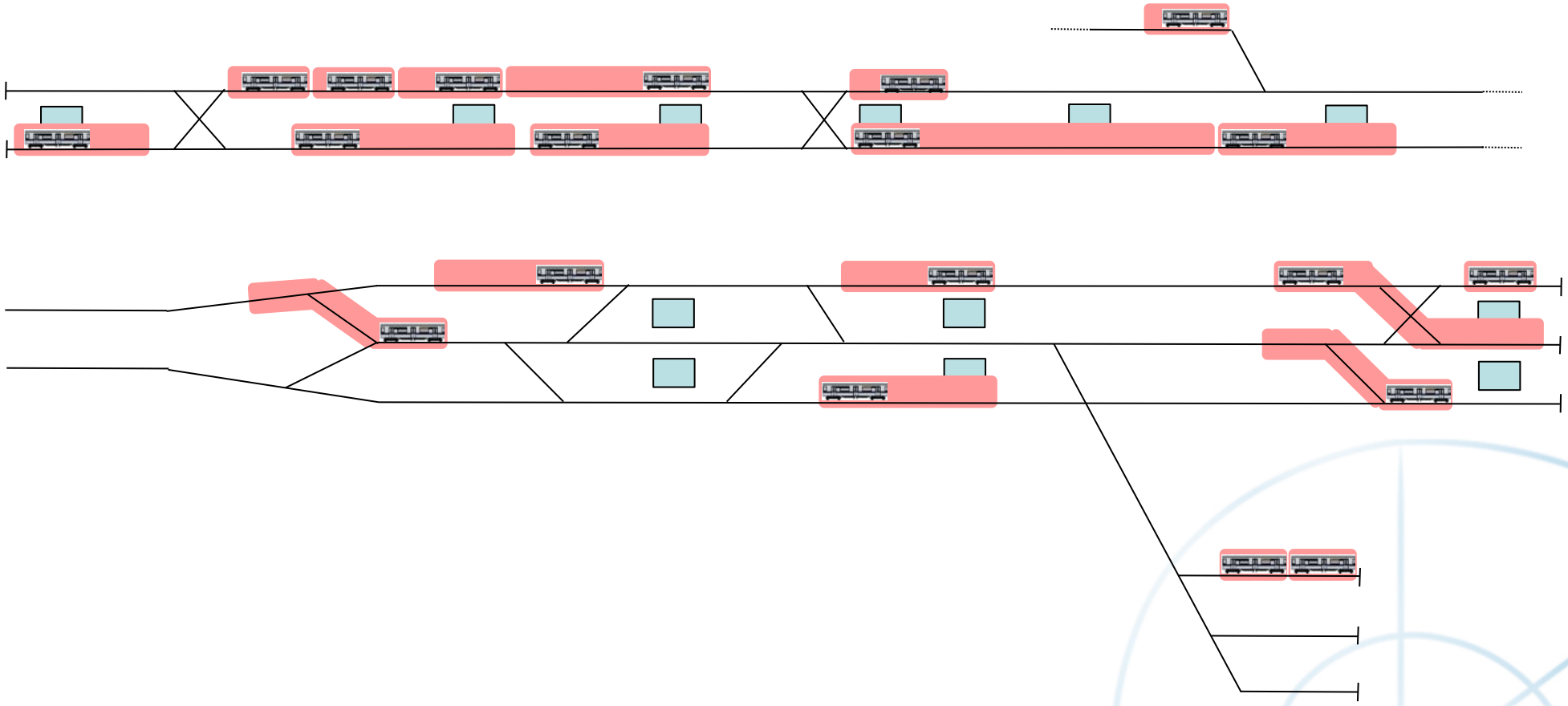
Présentation des propriétés globales

- 1: les collisions entre trains et les déraillements sur une aiguille déverrouillée ou incorrectement positionnée sont impossibles
- 2: les survitesses de trains CBTC sont impossibles

  - Avec ces propriétés, beaucoup d'accidents sont impossibles
  - En fait les propriétés 1 & 2 sont des moyens de garantir l'absence de blessures aux personnes
    - Sous propriétés d'une preuve plus (trop ?) globale…
    - Qui s'appuierait sur d'autres hypothèses, concernant les autres causes de blessures
      - Feu ? Electrocutions ? Fumées ? Agressions ?

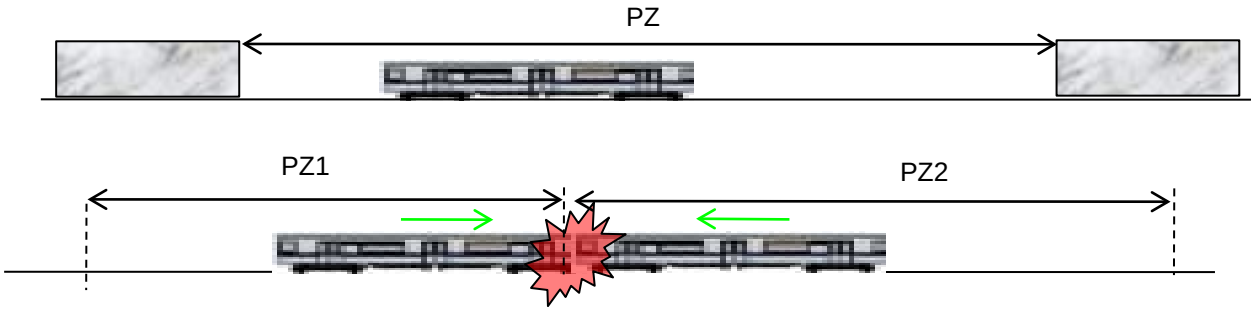
Un aperçu de la preuve de non collision

- A tout moment, il existe un ensemble disjoint de zones de protection PZ telles que chaque train reste dans sa PZ avec ses seules capacités de freinage.

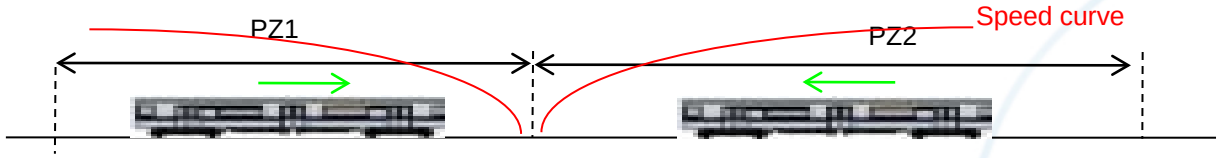


Un aperçu de la preuve de non collision

- Pourquoi « avec ses seules capacités de freinage » ?
  - Exemples de collisions avec des trains restant dans leur PZ sinon :

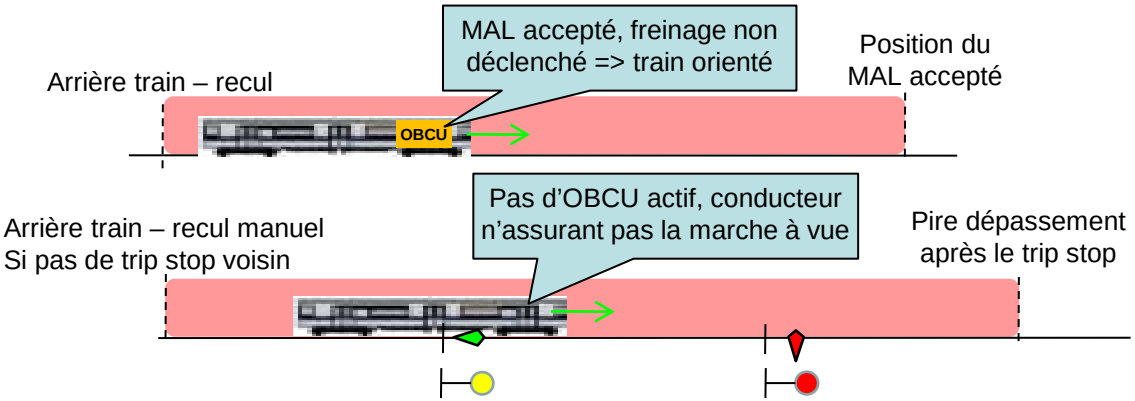


  - Donc à tout instant t, si après t la PZ d'un train restait inchangée, le train devrait rester dans cette PZ et la preuve de ceci ne doit s'appuyer que sur les forces de freinage garanties :

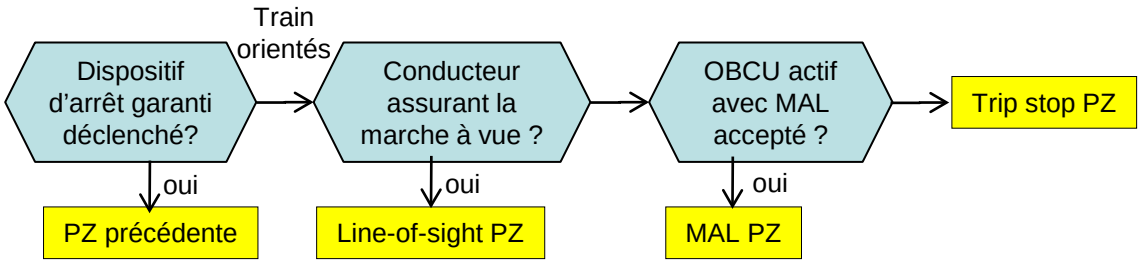


Un aperçu de la preuve de non collision

- Les PZ doivent être définies avec des critères précis, par exemple :



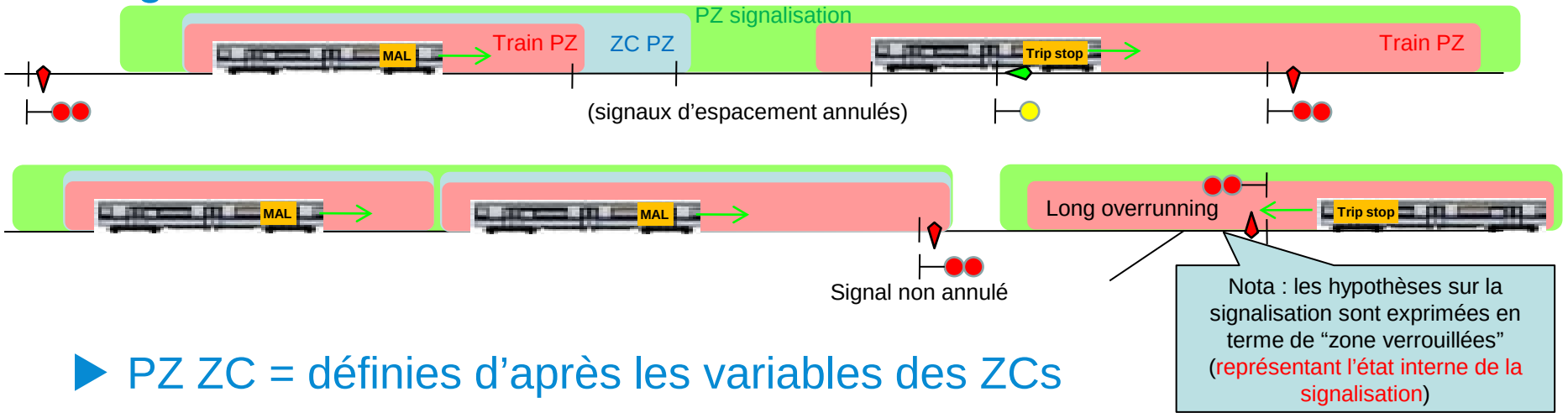
- Comment trouver le type de train (telle qu'il est défini pour la preuve)



- On définit les PZ dans chaque cas pour prouver leur existence. Alors :
  - Chaque train reste dans sa PZ avec ses seules forces de freinage (direct)
  - Toutes les évolutions conservent la non-intersection des PZ, avec aiguilles verrouillées (induction)

Un aperçu de la preuve de non collision

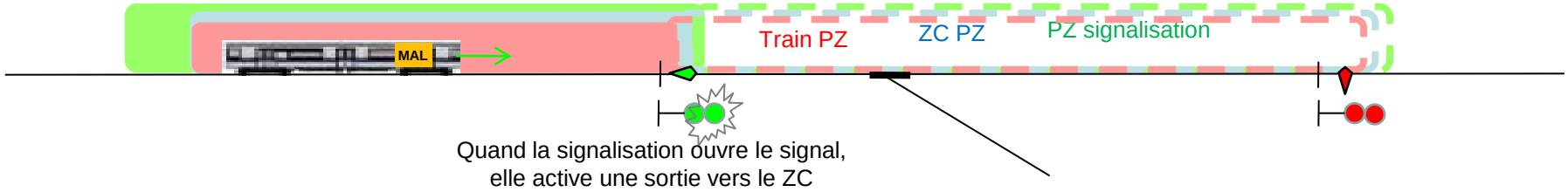
- On définit d'autres types de PZ « précurseurs », dont les « PZ train » héritent à chaque évolution: les PZ des ZC et les PZ de la signalisation



- PZ ZC = définies d'après les variables des ZCs
  - La PZ des trains CBTC hérite de ces PZ ZC lors de la réception des télégrammes
- PZ signalisation = définies d'après l'état de la signalisation
  - La PZ des trains manuels hérite de ces PZ signalisation via les trip stops (et le signaux…)
  - Les PZ ZC héritent des propriétés des PZ signalisation via les liaisons signalisation -> ZC
- Les propriétés des PZ ZC et signalisation sont également prouvées par induction

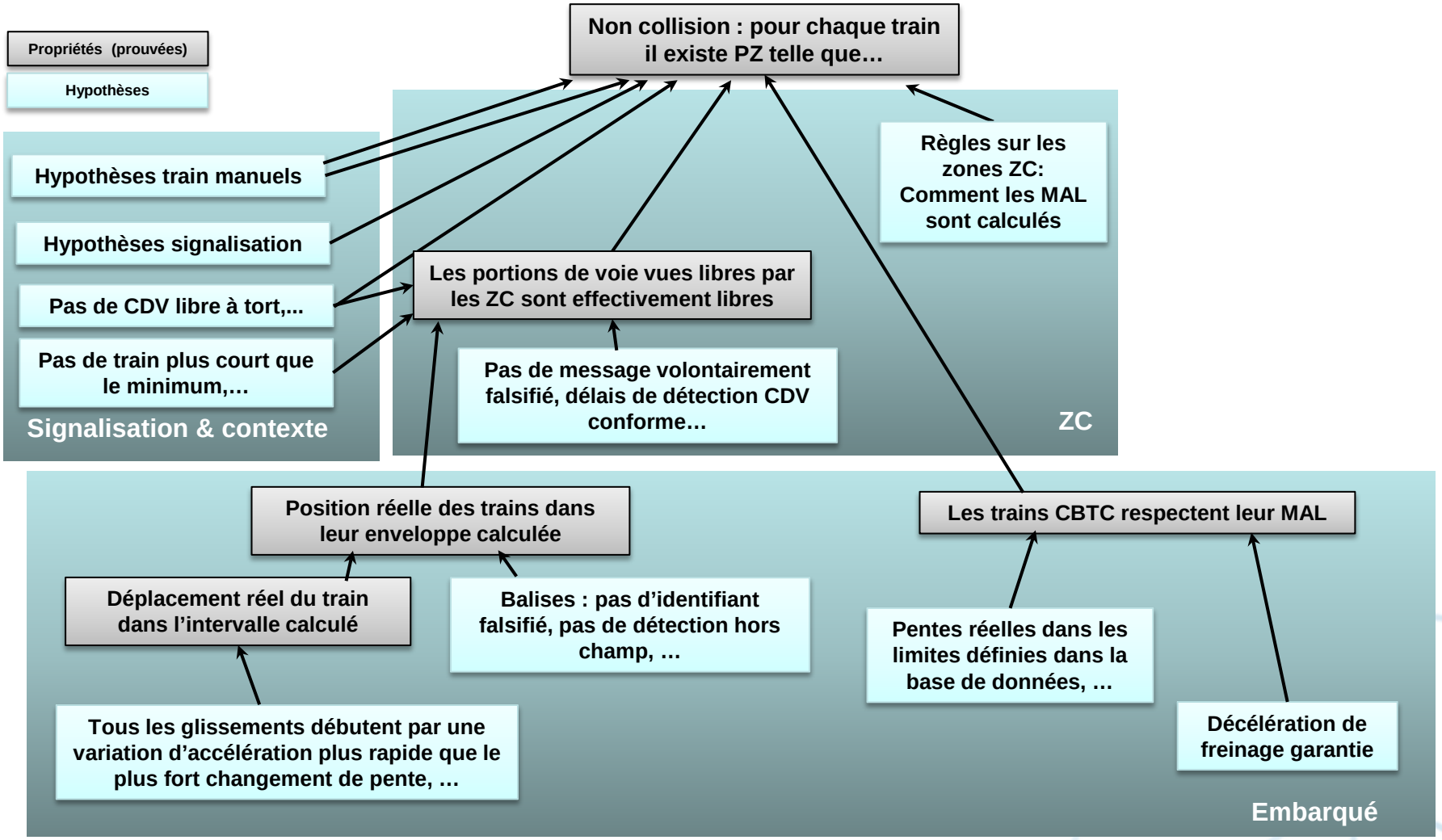
Un aperçu de la preuve de non collision

- L'évolution des PZ : là où réside la preuve…
  - Extension de zone : jusqu'au prochain signal ou obstruction



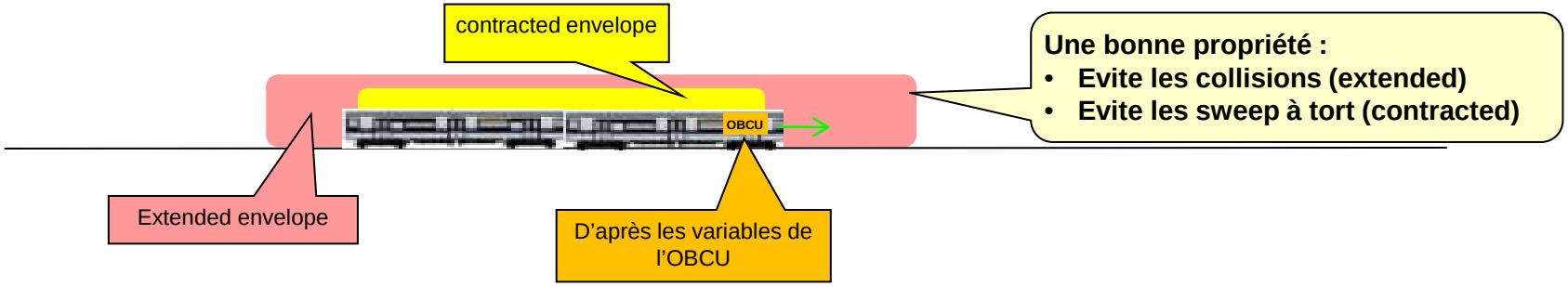
  - Réduction de zone derrière : libération d'espace devenu inaccessible
  - Réduction de zone devant : plus délicat
    - Exemple : réduction d'une zone par la signalisation seulement si :
      - Le ZC indique que le train respectera la nouvelle limite
      - Ou si c'est au-delà d'un maximum en temps et en distance, arrêt toujours possible
        - Signal annulé à t0, x0 => arrêt possible avant t0+T ou x0+D
- Les hypothèses de sortie apparaissent :
  - Exemple : la signalisation doit ouvrir les signaux de sorte que les "zones verrouillées" (PZ signalisation) qui s'en déduisent restent sans intersection

Propriétés & sous propriétés



Propriété de bonne détermination position / vitesse

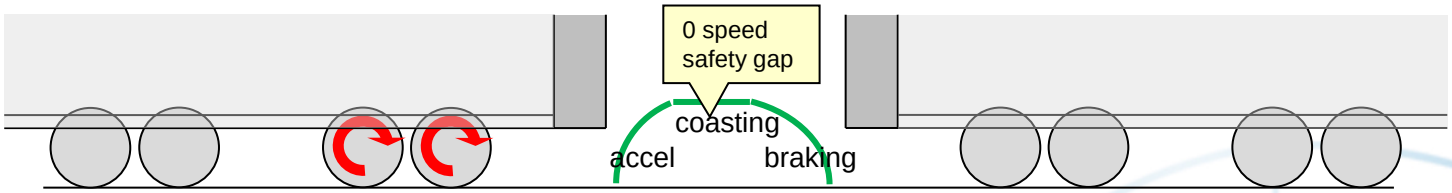
- Propriétés cibles :
  - A tout moment : contracted envelope incluse dans train réel inclus dans extended envelope



  - La vitesse réelle du train est toujours dans l'intervalle vitesse calculée + / - incertitude calculée
- Hypothèses :
  - *Calibration correcte / orientation correcte par le processus de localisation initiale*
  - *Limitations glissements / pires pannes des capteurs, limitations sur pentes (probabilistes)*
  - *constantes OBCU décrivant le train correctes*
  - *Base de donnée des balises correcte dans l'OBCU*
  - *Caractéristiques de détection des balises (et disposition de balises avec ID uniques et possibilités de diaphonies limitées)*
  - *Limitations de la voie (pires virages…) / des erreurs roues phoniques, des erreurs de comptage*
  - *Hypothèses sur le calcul de l'OBCU, temps de cycle garanti*
  - *Hypothèses représentant des connaissances d'automatique (Kalman filters)*
  - *Vitesse / accél. maximale (par exemple pour le comptage des dents de roues phoniques)*
- Démonstration possible avec tout ceci : OK !

Contrôle d'énergie correct (et non survitesse pour les trains CBTC)

- Propriétés cibles
  - Un train en mode MAL ne dépasse jamais le MAL qu'il a accepté
  - Un train en mode MAL ne fait jamais de survitesse
- Hypothèses
  - *La détermination position / vitesse est correcte (sous-preuve)*
  - *Les constantes de l'OBCU correspondent au caractéristiques réelles du freinage d'urgence et des limites de masses du train (dont masses rotatives)*
  - *Le freinage d'urgence est plus fort que la pire pente*
  - *Les autres forces (vent…) sont négligeables*
  - *Hypothèses probabilistes pour certains cas spéciaux (par exemple le cas du train avec patinage maximal au départ)*



  - *Les calculs de l'OBCU correspondent aux formules théoriques (+ T cycle garanti)*
  - *Hypothèses pour représenter les lois physiques (théorème énergie cinétique…)*
  - *Pentes correctes dans la base de données*

Suivi des trains

- Propriétés cibles
  - Les zones déterminées libres par les ZC sont effectivement libres
- Hypothèses
  - *Les constantes des ZC sont correctes :*
    - *Accélération maximale des trains*
    - *Longueur du plus petit matériel, overhang maximum (longueur entre 1er essieu et avant du train)*
    - *Les fins de voie sont réellement des fins de voie*
    - *Circuits de voie plus longs que l'écart mini entre essieux*
    - *Carte des circuits de voie correcte*
    - *Trous de shunt plus petits qu'une limite*
    - *Délai maxi d'acquisition des circuits de voie*
  - *Pas d'apparition de train au milieu*
  - *Les ZC calculent conformément aux algorithmes choisis, temps de cycle garanti*
    - *En incluant les communications de ZC à ZC et les communications train / ZC*
  - *Les trains se déplacent sur des portions linéaires de voie (n'atteignent jamais d'aiguilles incorrectes, sous-preuve re-bouclante, prouvée)*
  - *Les trains CBTC communiquent des enveloppes correctes (sous-preuve)*
  - *Hypothèses minimales sur la couche de communication: pas de messages volontairement falsifiés*
  - *Hypothèses sur les communications de l'OBCU (exemple : envoi des enveloppes calculées)*

Preuve des PZ (propriété globale)

- Propriété cible
  - A tout instant, il existe un ensemble de zones de protection PZ disjointes, telles que chaque train reste dans sa PZ avec son seul freinage. Ces PZ ne contiennent aucune aiguille déverrouillée ni obstacle.
- Hypothèses
  - Hypothèses sur la signalisation (utilisant la notion de zones verrouillées)
    - Pas de mouvement d'aiguille dans les zones verrouillées
    - Déverrouillage d'espace véritablement inaccessible
    - Ouverture de signaux uniquement sur de l'espace verrouillé
    - Extension / réduction des zones de protection compatible avec les comportements des trains
  - Hypothèses sur les procédures des trains
    - Par exemple : si un train CBTC en panne repart en manuel, le conducteur doit aller au prochain signal en marche à vue
  - *Les ZC calculent conformément aux algorithmes choisis, temps de cycle garanti*
    - En incluant les règles d'extension / réduction des PZ ZC d'après les indications des trains et de la signalisation
    - *En incluant les communications de ZC à ZC et les communications train / ZC*
  - *Les trains CBTC communiquent des enveloppes correctes (sous-preuve)*
  - *Suivi des trains correct (sous-preuve)*
  - *Contrôle d'énergie correct (sous-preuve)*
  - *Hypothèses minimales sur la couche de communication: pas de messages volontairement falsifiés*
  - *Hypothèses sur la communication de l'OBCU*

Fin de l’aperçu de la preuve…

- C’était juste un aperçu, bien sûr !

La sécurité et les propriétés prouvées

- La vérification formelle sur Flushing :
  - Propriétés prouvées :
    - Pas de collision et pas de déraillement (preuve “PZ”)
    - Pas de sur-vitesse
  - Niveau de détail : système
    - Inclus : algorithmes, exclu : conception bas niveau (dont le code informatique)
- Positionnement de ce travail dans le dossier de sécurité
  - A coté des audits, de la détermination des défaillances, etc.
- *Le juste équilibre entre les efforts formels parmi toutes les autres tâches de sécurité doit être bien déterminé*

Preuve et détermination des défaillances

- Nos hypothèses doivent être vérifiées malgré toute défaillance ou accumulation de défaillances possible
  - Possible = probabilité non en dessous du seuil de ce niveau de sécurité
- Exemple :
  - Nous avons des hypothèses sur comment un OBCU localisé met à jour les enveloppes
    - Hypothèse : *Si état = localisé, alors l'OBCU doit mettre à jour ses enveloppes conformément à …*
    - Donc : l'hypothèse n'exige plus rien si l'état n'est plus « localisé »
  - Si l'OBCU a une panne d'alimentation → pas de problème, ces hypothèses tiennent toujours grâce à "l'état localisé" (car on n'est plus dans l'état localisé)
    - Si les variables OBCU sont corrompues (toujours détecté): même raisonnement.
  - Les hypothèses choisies sont celles qui doivent être vérifiées malgré les pires défaillances
    - Le crash est possible si elles ne sont plus vérifiées…
- *Donc la détermination de défaillance et l'étude de leur probabilité et accumulations est toujours nécessaire !*

Les défaillances

- Aucune propriété ne peut supporter *toutes* les défaillances imaginables…
  - Exemple : un relais de sécurité ne s'active pas sans commande
    - Mais ceci peut-il tenir en cas de défaillances « sabotages » ?
  - Donc les considérations probabilistes pour écarter les cas extrêmement improbables sont *toujours requises*
    - On suppose nos hypothèses vérifiées, sauf dans ces cas extrêmement improbables
    - Certaines hypothèses sont explicitement probabilistes (exemple : pas de glissement indétectable)
- Si les cas où la propriété cible ne tient plus doivent être $<10^{-9}$, alors les cas où 1 hypothèse ne tient plus doivent être (au moins) $<10^{-9}$…
  - Avec des solutions pour éviter de trop accumuler les pires cas :
    - Exemple : la preuve de position (train dans son enveloppe calculée) s'appuie sur des hypothèses de la forme "entrées comprises entre certaines bornes de pire cas"
      - On peut alors valuer numériquement ces hypothèses avec un ensemble de bornes n'accumulant pas les pires cas au-delà du possible
        - *Atteindre les pires bornes en même temps sur toutes les entrées : invraisemblable !*

Les analyses des dangers

- Les normes exigent que la démonstration de sécurité parte d'une analyse des dangers
  - En considérant les accidents possibles
  - En décidant ce qui est acceptable ou non
- Les collisions frontales de trains sont à rendre impossible avec un niveau SIL4
    - Occurrences moins que 10^{-9} par heure, temps moyen entre occurrences 114 000 ans
  - Comment ceci a-t-il été décidé ?
    - On considère le nombre de tués potentiels…
    - Mais il y a une **décision de ce qui est acceptable / non acceptable** (jugement humain)
- Ce genre de décision n'est pas une affaire de preuve
  - L'analyse des dangers est toujours requise, c'est le point de départ du choix des propriétés cibles
- Ainsi que le disent les normes, il est très mauvais de mal jauger les dangers…

Niveau d'intégrité de sécurité & méthodes

- Les normes exigent des méthodes appropriées pour parer **les erreurs de conception**
  - Nommées défaillances systématiques
  - Les calculs de probabilités ne sont pas considérés comme appropriés ici car ces erreurs s'expriment de manière systématique dans certaines conditions
  - Les normes définissent des méthodes appropriées pour parer les défaillances systématiques, en fonction du niveau d'intégrité de sécurité
- Les hypothèses issues d'une vérification formelle système ont un niveau SIL « hérité »
  - Donc les méthodes appropriées sont requises pour la conception sous jacente
- Pas d'erreur de conception dans le niveau de conception système (couvert par la preuve) : considéré SIL4
  - Bien que les normes considèrent les méthodes formelles plutôt un à niveau software ("Highly Recommended" pour SIL4 dans 50128…)

Le formel au niveau système ou au niveau détaillé ?

- Le plus souvent les méthodes formelles ne sont utilisées qu'au niveau software
  - La preuve couvre le passage "spécification software → code software"
    - En parant les erreurs dans le code sauf si elles étaient déjà dans la spécification
- Ici : preuve des propriétés de sécurité de plus haut niveau couvrant la conception système
- Comparaison ?
  - La preuve au niveau système est généralement dédiée aux seules propriétés de sécurité
    - Alors que la preuve software inclut du fonctionnel (car celui-ci figure dans les spécifications software)
    - Au niveau système : aspects fonctionnels = performances (exemple : réduire l'intervalle entre trains au maximum tout en restant sécuritaire)
  - La preuve au niveau système couvre « tous les aspects »
    - Depuis les algorithmes du logiciel sous-jacent jusqu'aux procédures d'exploitation
  - En preuve software, on a un lien direct entre les derniers niveaux de modèles formels et le software (génération de code)
    - Avec une action directe contre les bugs de codage bas niveau
    - La preuve système produit des hypothèses sur les propriétés que le logiciel doit assurer : action indirecte
- Une affaire de choix et d'équilibre
  - *Placer le paratonnerre là où la foudre risque de tomber…*

A propos des processus / organisation des projets

- Le succès d'un projet et la sécurité sont liés ; il y a de nombreux moyens d'améliorer le succès d'un projet :
  - Plus d'encadrement processus ?
  - Plus de relectures et de qualité ?
  - Plus de formation ?
  - Plus de science ?
  - Plus de gens et de moyens ?
  - Plus de tests ?
  - Plus de communication entre les équipes et le client ?
- *Il ne doit rien manquer !*
- Les preuves formelles sont à placer en considérant l'équilibre de tout ceci
  - Et en considérant la charge appropriée pour la sécurité

Le budget sécurité dans un projet

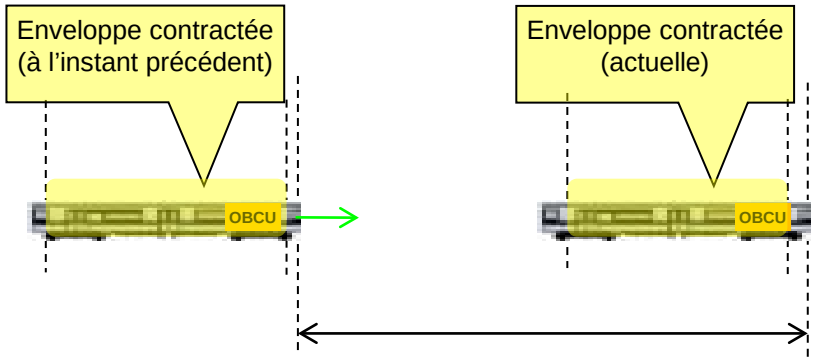
- La sécurité : une “performance” qui ne se voit pas dans les tests
  - La réduction d’intervalle entre trains grâce à un nouveau CBTC est directement visible
  - L’amélioration de la sécurité grâce (par exemple) à une preuve formelle n’est pas visible de la même manière…
    - Et “il n’y a pas eu d’accident pendant X années” n’est pas suffisant pour des systèmes où l’intervalle moyen entre événements redoutés est 114 000 ans…
- Retour sur investissement plus difficile à évaluer
  - Les nouvelles fonctionnalités sont directement visibles, pas les améliorations de la sécurité
    - Sauf si le système était vraiment dangereux avant !
- Un bon dimensionnement de l’effort de sécurité dans le projet et un emploi optimal de cet effort doit être une préoccupation constante
  - Budget trop réduit sur la sécurité :
    - Le système peut être dangereux (s’il n’est pas bloqué à la démonstration de sécurité…)
  - Budget trop large sur la sécurité :
    - Système trop cher, risque d’échec projet
- L’emploi des méthodes formelles doit être considéré dans cet esprit
  - Après un accident, les choses qui auraient du être faites paraissent tellement naturelles !

Dans la conception ou parallèlement à la conception?

- Preuve formelle au niveau système : à coupler avec les tâches de conception ou avec les tâches ISA (Independent Safety Assessors) ?
- Avec la conception :
  - Cela favorise la communication avec les concepteurs, la compréhension de la conception et le réalisme des hypothèses choisies
  - Les questions et les points particuliers sont partagés très tôt
- Avec l'ISA:
  - Indépendance
- Notre opinion : quelle que soit l'organisation,
  1. L'équipe de preuve doit être une équipe spécifique
     - Impossible de créer la conception sécuritaire et formaliser le raisonnement à la fois
  2. L'équipe de preuve ne doit pas démarrer trop tard (pas quand la conception est figée)

Communications équipe de preuve / conception : ces maudites « notions »…

- Exemple : « mise à jour des enveloppes »



- On doit donc avoir par exemple cemax = cemax_r + Scmin
- Peut-on trouver *cemax = cemax_r + Scmin* dans le code de l'OBCU ?
  - NON, le software désigne sûrement les positions de voie avec un système segment / abscisse
    - Positions représentées par <segment name, abscissas>, et non par une abscisse simple… Conception bas niveau.
- cemax, cemax_r, Scmin = **notions** / notations venues de la preuve
  - Il faut un Effort pour les faire correspondre au logiciel réel
  - On ne doit donc définir ces notions que quand c'est **inévitable**
  - Et les employer près de leur définition (**ne pas demander aux lecteurs de les mémoriser** !)

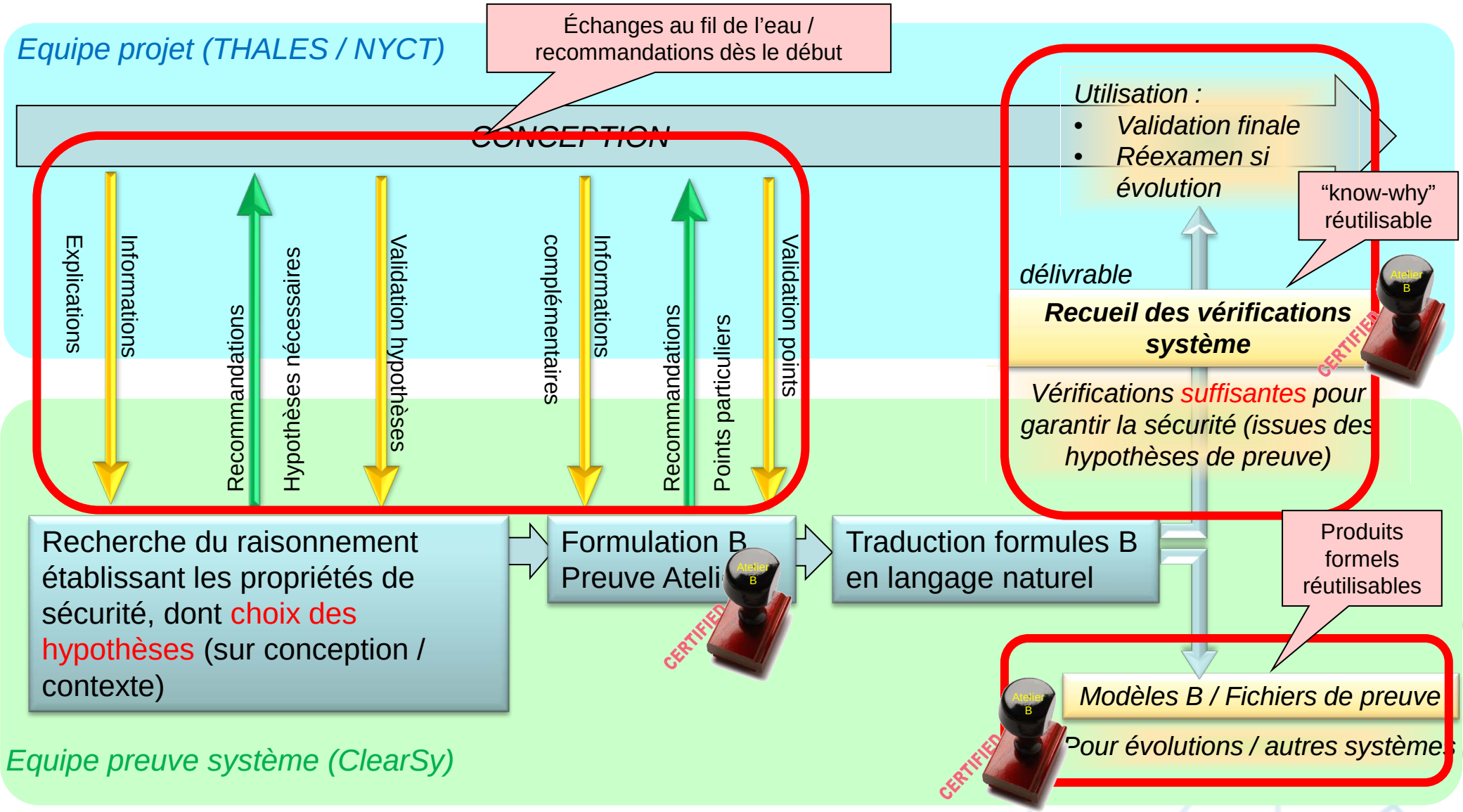
“notions” nécessaires… Au moins bien les définir

- Donc les hypothèses ne s’explicitent que par des mots…
- Imaginons une hypothèse échantillon : “la maison est rouge”
  - Mais qu’est-ce qu’est “*la maison*” ?
    - Murs ? Toit ? Intérieur ? Extérieur ?
  - Et qu’est-ce que “*rouge*” ?
    - Orange sombre ? Brillant ? Rayé ?
- Définir ceci signifie lier les mots à la réalité
  - Formuler un critère clair pour définir ce qui fait partie de « la maison » et ce qui n’en fait pas partie
  - Et un critère pour définir ce qui est « rouge » ou pas
- Seulement alors l’hypothèse est correctement vraie ou fausse
- Méthode : préciser les notions « maison » et « rouge » pour nous
  - Et formuler les hypothèses avec ces notions

Les communications et l'optimisation de la construction de la preuve

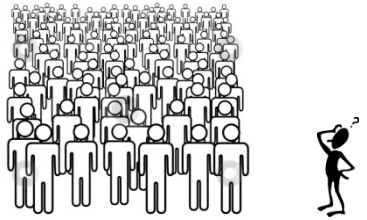
- A nouveau, la phase en langage naturel est essentielle pour « trouver le chemin » qui mène aux propriétés voulues à partir d'hypothèses réalistes
- Les documents projets décrivent le plus souvent le « comment »
  - Avec le nom des fonctions, les champs des messages, etc.
- Processus Bottom-up :
    - Formaliser tous les détails bas niveau
    - En déduire les propriétés de plus haut niveau
    - Jusqu'aux propriétés voulues
  - Notre expérience : ce processus est une **mauvaise idée**
    - Car il implique de formaliser tous les détails superflus
- Notre expérience : l'équipe de preuve doit avoir la volonté dirigée vers comprendre comment les propriétés voulues sont assurées
  - *Aussi vite que possible, aussi directement que possible*
  - En utilisant les contacts avec les concepteurs dans cet esprit
  - En lisant les documents dans cet esprit
    - Bien que par la suite il faudra bien faire des vérifications intégrales dans les documents
      - Pour contrôler qu'une fonction existe il suffit de lire les documents jusqu'à cette fonction. Pour contrôler *qu'elle n'existe pas* il faut tout lire…

Résumé des produits



Preuve formelle système : conditions du succès

- *D'après nous…*
  - L'équipe de preuve doit réellement vouloir :
    - Comprendre le système (« plonger » dans le domaine)
      - Mais optimiser leur raisonnement (en employant le minimum nécessaire de détails)
    - Echanger avec les concepteurs (dans l'esprit d'apporter un service)
      - En se rappelant que les notions et les noms ajoutés sont une pollution
    - Formaliser le raisonnement optimisé (et seulement le raisonnement utilisé)
  - Organisation : il faut un accès aux personnes qui connaissent réellement le système
    - Avec assez de temps
    - Equipes preuve // conception : ni 1 à 100 ni 100 à 1 !



  - Une organisation favorisant une communication facile et légère
    - Test : les équipes arrivent-elles à échanger des schémas manuels (dans les 2 directions) ?
  - L'équipe de preuve doit maîtriser la méthode formelle pour que ce soit un outil (savoir ce qu'elle peut et ne peut pas faire)

Qualifications pour l'équipe de preuve

- Un leader d'équipe est nécessaire pour rappeler constamment les conditions de succès (diapo précédente)
- Bien sûr, il faut des compétences avec la méthode et l'outil formel (méthode B / Atelier B)
  - Comme un outil, mais ce n'est pas le point principal
- Une « ouverture technique » est nécessaire
  - Les membres de l'équipe doivent désirer franchir les limites des domaines techniques

Bénéfices d'une démonstration formelle au niveau système

- Actuellement : système sûr car
  - Les experts ont donné un avis favorable
  - Le constructeur a mis en service d'autres systèmes sécuritaires comparables
  - Un dossier de sécurité conséquent a été approuvé
- Avec une preuve système **rejouable** : système sûr car *en plus* :
  - L'impossibilité des accidents est démontrée
    - Chaque pas de démonstration est vérifiable par logique seule
      - Ces pas peuvent être présentés à tous ceux qui veulent les voir
    - On arrive aux propriétés à partir d'hypothèses bien définies
      - On peut présenter ces hypothèses à tous ceux qui veulent les voir (et ils comprendront ce qu'elles représentent sur le terrain)
    - La correction des pas de preuve est garantie par un outil (Atelier B)
  - Tout le monde peut lire la preuve
    - Éventuellement remettre en doute la validité sur le terrain de telle ou telle hypothèse;
    - **Mais JAMAIS remettre en doute le fait que les propriétés se déduisent logiquement de ces hypothèses**

Choisir de faire une preuve formelle système

- *Critères (de nouveau : d'après nous…):*
  - Besoin d'une garantie de sécurité globale
    - Avec un focus équilibré sur toutes les parties du système choisi
  - Besoin d'avoir toutes les conditions nécessaires à la sécurité exprimées
  - Besoin d'avoir le "pourquoi c'est sécuritaire" exprimé
    - Exprimé et re-jouable
  - S'il n'y a pas de problème évident à corriger avant
    - Technique ou organisationnel
  - *La résistance d'une chaîne est celle du maillon le plus faible…*
    - *Utiliser les méthode formelles pour renforcer un maillon ou pour démontrer qu'il n'y a pas de maillon faible*

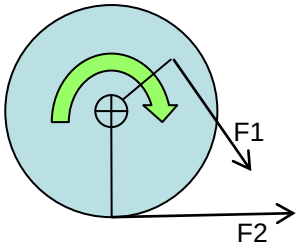
Remerciements

- Merci pour votre attention…
- Et grand merci à NYCT / Thales

- Pour toute question complémentaire, contactez:
  - Denis.sabatier@clearsy.com
  - Lilian.burdy@clearsy.com

Des hypothèses explicites et vérifiées

- Exemple: un essieu libre ne peut pas patiner
  - Seulement freiné = pas de moteur de traction, patiner = glisser plus vite que la vitesse train
  - Propriété utile pour un essieu avec roue phonique…
  - Mais est-ce réellement vrai ?
    - Si oui on devrait pouvoir le démontrer (*accélération en rotation x moment d'inertie = couples*)



    - Pas toujours vrai… Pour un essieu avec élan, sur rail glissant, si le train décélère fortement
      - Mais ces cas de patinages sont considérés comme irréalistes
- Voici un exemple d'une hypothèse qui semble évidente, et qui en fait demande le jugement d'un expert domaine
  - La preuve oblige à expliciter TOUTES les hypothèses
  - Une fois explicité, ce type de problème peut être examiné correctement

Un aperçu de comment on prouve les propriétés plus algébriques

- Exemple : lien entre la rotation réelle de l'essieu et les sorties OBCU
  - Utilisé pour la propriété « train dans son enveloppe », par exemple :

  - Hypothèses :

R(t): rotation réelle → Roue phonique : N

Emploi de variables physiques, avec précision infinie

t, N(t) –

    - A t, Rc peut provenir d'un N datant de t-2Tc
    - Variation maxi / mini de R pendant 2Tc : emploi des hypothèses telles que l'accélération / décélération maxi du train
  - En combinant les équations: on prouve t, Rc(t) – • ' • • • • • • • • • •
    - OK si l'erreur calculée par l'OBCU est plus grande que e''
    - Pas de preuves : règle très simple (ex: a<b et b<c implique a<c; a<b et x>0 implique ax<bx…)