MADE IN FRANCE

CLEARSY
SYSTEMS ENGINEERING

ANALYSE FORMELLE DE LOGICIEL

OFFRE CLEARSY
ANALYSE FORMELLE DE LOGICIEL

CONTACT@CLEARSY.COM

ANALYSE FORMELLE DE LOGICIEL

CLEARSY propose une méthode d'analyse inédite permettant de faire la preuve mathématique de la conformité de tout ou partie d'un logiciel vis-à-vis d'un besoin fonctionnel et de sécurité.

L'activité d'analyse que nous proposons établit un lien formel direct entre le code source et les propriétés définies pour le système qui intègre le logiciel. Cette activité permet de détecter tout type de non-conformité ayant pu être introduite dans le cycle de développement : **depuis la pensée des algorithmes identifiés en phase de définition système, jusqu'à leur réalisation concrète** en tenant compte des éventuelles contraintes spécifiques d'implémentation.

En particulier cette méthode est proposée quand les activités traditionnelles de validation et de vérification montrent un certain nombre de lacunes :

- La vérification par scénario est possiblement incomplète sur des systèmes contenant un trop grand nombre d'états.
- La découverte de défauts intervient tardivement dans le cycle de développement. Ceci est d'autant plus vrai lorsqu'il s'agit de défauts de conception du système.

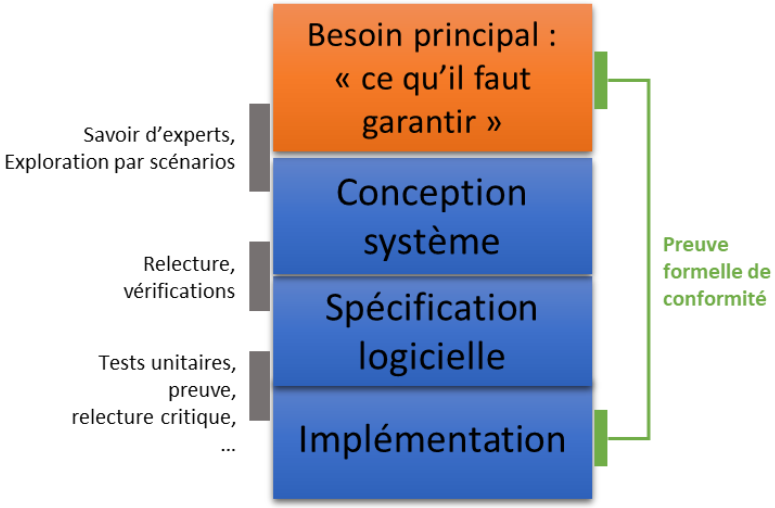


Figure 1 - Position de l'étude

A l'inverse, l'activité d'analyse formelle proposée par CLEARSY est complète : elle garantit de couvrir **tous les cas de fonctionnement possible**, incluant de fait tous les dysfonctionnements et toutes les pannes (du système) qui n'ont pas été explicitement écartées. C'est le bénéfice de l'emploi d'une méthode basée sur les mathématiques et une approche par propriété, et non une approche basée par cas.

Les bénéfices d'une telle analyse formelle sont variés :

- Consolider ou mettre à plat les besoins fonctionnels et de sécurité : expliciter ce que le système doit garantir.
- Identifier des non conformités : des non-respects de propriété pouvant avoir de lourdes conséquences sur la sécurité ou sur l'exploitation du système intégrant le logiciel.
- Retrouver les raisonnements initiaux : ceux imaginés par les concepteurs du système.

- Mettre à plat les choix de conception : en retrouvant leur raison profonde, garantissant la maitrise complète du logiciel étudié.
- Identifier des nœuds de complexité inutiles ou obsolètes : pour simplifier le logiciel et obtenir un gain de performance ainsi que des améliorations fonctionnelles (plus le logiciel est simple, plus il a des chances de bien fonctionner).

La méthode peut s'appliquer de **manière rétroactive sur un logiciel existant.** Elle gagne également à être pérennisée dans un processus industriel traditionnel, pour déplacer **l'effort d'analyse vers les phases amonts** du développement et **détecter les erreurs au plus tôt** pour limiter la boucle de correction (et les tests à refaire).

INTÊRET INDUSTRIEL

La méthode se veut pragmatique pour des résultats projets concrets et rapides. Les résultats obtenus ont permis à CLEARSY d'obtenir la garantie que l'activité d'analyse formelle de logiciel représente un réel intérêt industriel tant pour le fournisseur du produit que pour son exploitant. Elle gagne à être utilisée sur des produits qui évoluent régulièrement dans le temps, ou largement déployés ou qui intègrent des options en fonction de leur utilisation.

Pour un constructeur :

L'industriel en charge du développement trouvera son intérêt dans le fait de rendre plus robuste son produit et donc d'éviter de graves problèmes à l'exploitation, d'en assurer la pérennité (des connaissances et des raisons explicites des choix et des besoins), de garantir la transmission des compétences, mais également d'étoffer significativement le dossier de sécurité.

De plus il pourra tirer profit de la méthodologie utilisée pour la généraliser et ainsi créer une **alternative au cycle de développement / vérification traditionnel basé sur l'analyse de scénarios** (analyse des cas sans garantie d'exhaustivité bien sûr).

Pour un exploitant :

L'exploitant qui en cas de non-conformité, et donc de problème à l'exploitation, voit sa responsabilité engagée a également un fort intérêt à appliquer cette méthode.

En effet l'analyse des livrables de l'industriel que peut réaliser CLEARSY pour le compte de l'exploitant permet à la fois d'améliorer la maitrise technique du produit et d'obtenir une méthodologie systématique, sûre et outillée pour **valider mathématiquement la conformité** des logiciels vis-à-vis de leur besoin d'exploitation et de leur sécurité.

Elle permet surtout d'éviter des problèmes pendant l'exploitation, qui peuvent être d'autant plus graves s'ils portent atteinte à la sécurité (avec par la suite de sérieuses implications : interruption d'exploitation, sanctions financières, problème d'image, conflit avec l'industriel).

Les **systèmes interopérables** constituent une cible particulièrement pertinente pour ce type d'approche. En effet, le partage des responsabilités entre les différents sous-systèmes intégrant les logiciels est d'autant plus important. En cas d'incident, chaque constructeur cherchera à défendre le fait que son logiciel n'est pas incriminé. L'approche que nous proposons permet de fournir la preuve mathématique que la combinaison des sous-systèmes satisfait le cahier des charges du système.

APERCU TECHNIQUE

La méthode est basée sur une approche d'analyse en se basant sur des propriétés qui doivent être conservées. Le besoin du système est décliné jusqu'à un niveau de détail permettant de faire intervenir directement les variables du logiciel.

Cette méthode d'analyse apporte une vision modulaire et étagée du logiciel et des concepts qui y sont manipulés. Chaque variable contribuant au respect du besoin est identifiée, son rôle dans la démonstration de conformité est porté de manière claire par des propriétés liant la variable logicielle aux éléments d'environnement réels / physiques / concrets qui interagissent avec le logiciel. Le raisonnement de conformité au besoin est établi en combinant ces propriétés de manière à montrer qu'elles suffisent à le prouver.

La méthode comprend également une analyse de toutes les évolutions possibles des différents éléments (aussi bien matériels que logiciels) et à vérifier que ces évolutions préservent ces propriétés, ce qui permet de démontrer que le besoin du système est garanti dans tous les scénarios possibles d'exécution, *pannes et dysfonctionnements inclus*.

ILLUSTRATION SUR UN EXEMPLE

Prenons le cas d'un logiciel sécuritaire en charge de délivrer une autorisation d'ouverture des portes d'une rame de métro automatique.

Les conditions fonctionnelles d'autorisation d'ouverture peuvent être nombreuses : train correctement positionné face au quai, à l'arrêt, desserte du quai en question prévue pour cette circulation, ouverture d'urgence demandée, etc.

Néanmoins une chose est claire, l'autorisation d'ouvrir les portes ne doit (entre autres) jamais pouvoir être délivrée alors que la rame de métro est en mouvement. Cette affirmation correspond à l'expression d'un besoin, en l'occurrence ici d'un besoin de sécurité. Cette propriété doit toujours être vraie, on parle de propriété invariante.

Afin de garantir cela, le logiciel implémente nécessairement des algorithmes plus ou moins complexes permettant d'obtenir une estimation de la vitesse du train. Cette estimation ne peut être qu'une approximation de la vitesse réelle, garantie à un delta près. On voit immédiatement que, afin de garantir le besoin de sécurité exprimé ci-avant, cet algorithme devra avoir tendance à surestimer la vitesse réelle. En effet dans le cas contraire, il pourrait conclure que la rame de métro est à l'arrêt alors qu'en réalité celle-

ci est en mouvement (vitesse non nulle), ce qui pourrait lui permettre de délivrer une autorisation à tort d'ouverture des portes.

Supposons maintenant que les algorithmes implémentés pour obtenir cette estimation de vitesse se basent sur des capteurs capables d'analyser finement la vitesse de rotation des roues (ou essieux) du métro. Cette information est utile pour déterminer une vitesse globale de mouvement mais elle n'est pas suffisante, il faut également considérer un mouvement de glissement (obtenu sans rotation des roues) sans quoi il est faux d'affirmer qu'un matériel roulant dont les roues ne tournent pas est assurément immobile.

Le logiciel développé intégrera donc en quelque sorte les modules suivants :

- estimation d'une vitesse de rotation, disons vrot_sw,
- estimation d'une vitesse de glissement, disons vgliss_sw,
- détermination d'une vitesse globale v_sw de mouvement à partir des vitesses précédentes.

C'est ce que nous appelions précédemment une vision « étagée » du logiciel : *vrot_sw* et *vgliss_sw* sont des concepts bas niveau, utilisés pour déterminer une information plus haut niveau (à un étage supérieur) *v_sw*.

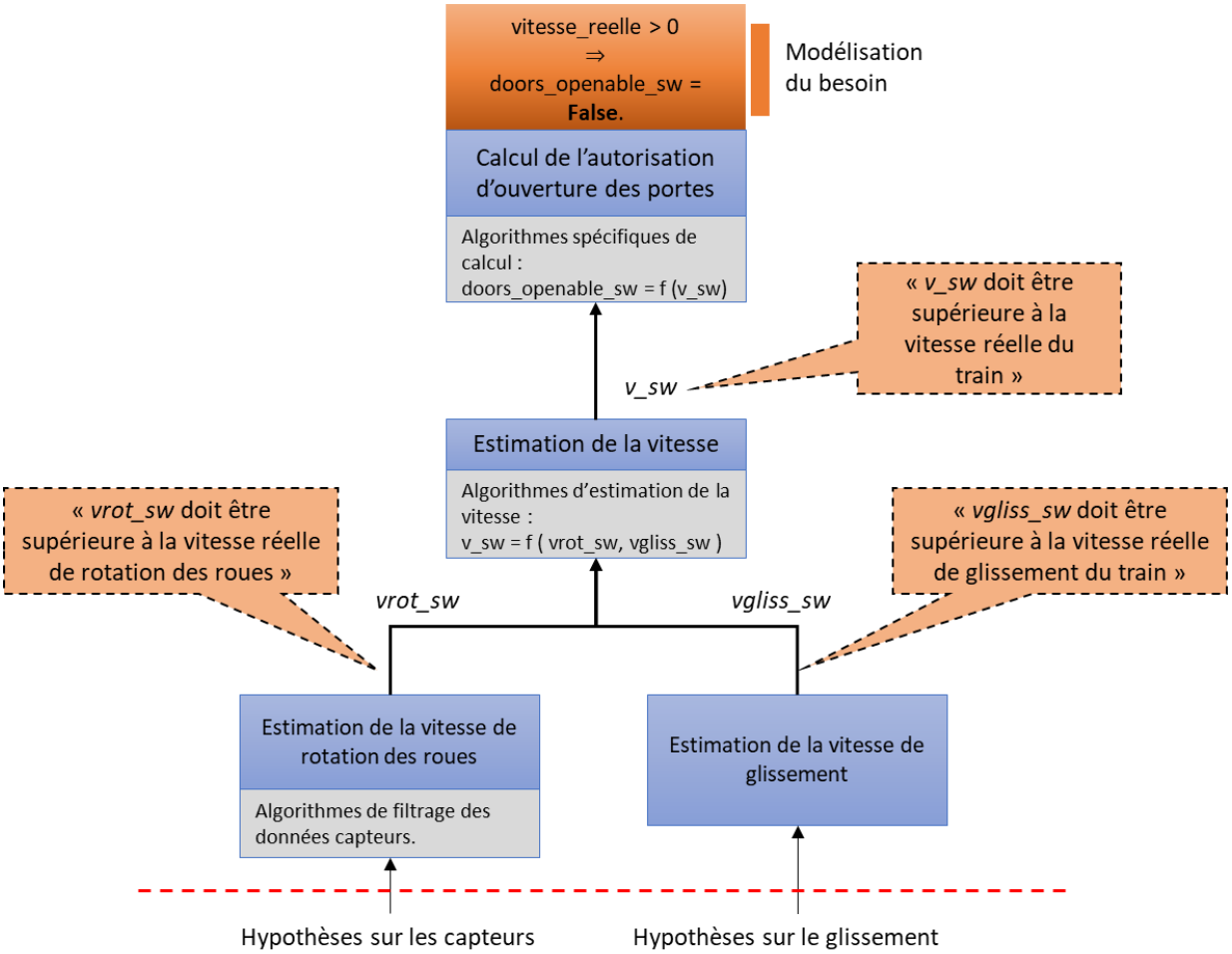


Figure 2 - Illustration de la méthode sur un exemple : Vision étagée et modulaire du logiciel et des propriétés qui doivent être garanties

OUTILS

CLEARSY propose de faire supporter cette démarche par un outil capable de supporter la modélisation système et la preuve formelle de propriétés. La modélisation « B-Système » déployée *via* l'outil industriel Atelier B remplit ces caractéristiques et a été utilisé dans les projets références de CLEARSY dans le domaine.

Néanmoins **la méthode ne dépend pas de l'outillage**, l'instrumentation de la méthode peut être réalisée sur une cible différente de l'Atelier B. CLEARSY dispose d'ores et déjà de premières références en employant d'autres outils, l'idée étant de reprendre les outils déjà employés par l'industriel.

L'intérêt de faire supporter la démarche par un outil est multiple :

- Assurer la mise au point des propriétés que le logiciel doit préserver avec le niveau de détail et de précision adéquat.
- Garantir l'absence de faute logique dans les raisonnements effectués jusqu'ici « sur papier » afin d'obtenir une garantie de conformité appuyée par un outil qualifié.
- Découvrir les hypothèses implicites dans les raisonnements : l'Atelier B n'a aucune connaissance présupposée du domaine, toute hypothèse devra lui être explicitement transmise.

SYNTHESE

Le schéma ci-dessous présente de manière synthétique les différentes étapes décrites ci-avant pour obtenir la preuve formelle qu'un logiciel est conforme vis-à-vis d'un besoin.

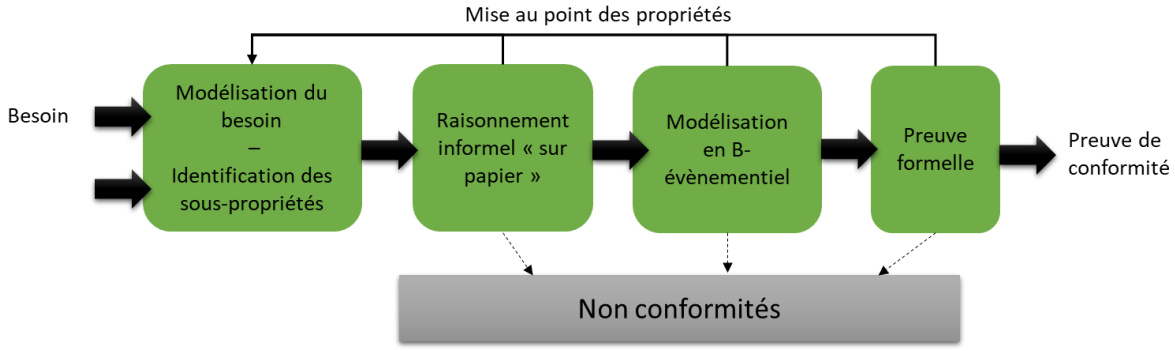


Figure 3 - Synthèse du processus d'analyse formelle du logiciel

ENTRANTS

Ce type d'analyse peut être réalisé dès lors que sont mis à disposition de « l'équipe formelle » en charge de réaliser l'étude les deux éléments clés ci-dessous :

- **Le besoin à démontrer**, celui vis-à-vis duquel on cherche à évaluer la conformité des algorithmes implémentés. Généralement il s'agit de propriétés issues de l'analyse des évènements redoutés mais cela peut également être une propriété fonctionnelle, sans lien avec les objectifs de sûreté.

- **Une représentation fidèle du code** implémentant cette fonctionnalité : la spécification logicielle, un modèle abstrait du code si celui-ci a été développé en utilisant des méthodes formelles (B-logiciel par exemple), ou directement le code lui-même.

En pratique pour assurer une meilleure efficacité des travaux il faut également prévoir des **échanges ponctuels avec une personne familière avec le logiciel** et plus précisément avec la fonctionnalité étudié(e).

LIVRABLES

Les livrables de ce type de prestation se présentent comme une **collection de documents** dans lesquels sont indiqués :

- Le périmètre précis de l'étude ainsi que les hypothèses faites sur les entrants de la fonctionnalité étudiée (messages en provenance d'un autre sous-système, capteurs, résultat de calcul d'une autre fonctionnalité…).
- Les propriétés déclinées du besoin fonctionnel et de sécurité, faisant intervenir les variables logicielles.
- La démonstration de conformité du code vis-à-vis du besoin.
- Les éventuelles situations de non-conformité ainsi que les éléments nécessaires à leur analyse.

Ils s'accompagnent également des **modèles développés** avec l'Atelier B (ou tout autre outil retenu pour la prestation) qui pourront être maintenus à l'avenir afin de questionner les évolutions envisagées en les intégrant au modèle avant même de lancer le cycle de développement logiciel.

Ces modèles et la preuve qui les accompagne garantissent mathématiquement l'exactitude des raisonnements.

REFERENCES COMMERCIALES

Les clients sont les grand industriels ou les donneurs d'ordre du marché ferroviaire.

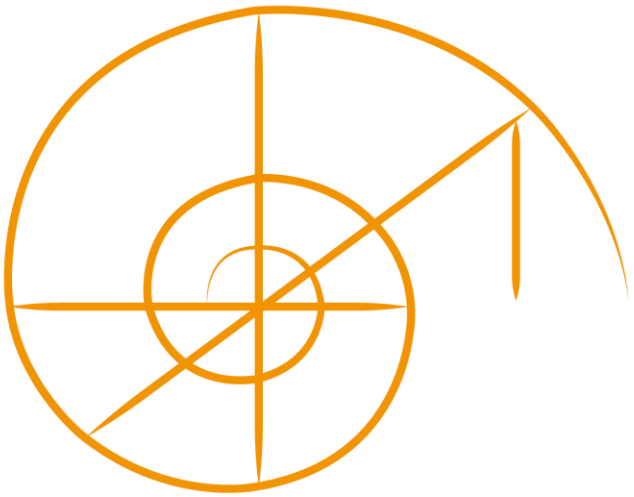
Ce type d'approche a d'ores et déjà été employé sur un logiciel sécuritaire SIL4 (selon la norme Cenelec EN 50128) dans le domaine du ferroviaire, sur une fonctionnalité représentant plus de 12 000 lignes de code et 100 pages de spécification logicielle. En termes de volume cette étude a représenté au global l'équivalent de 1,5 homme / an pour des profils experts, avec des premiers résultats obtenus rapidement, puis tout au long du projet.

Une approche similaire a également été employée dans le domaine automobile pour valider du code embarqué permettant de gérer le neiman (antivol installé sur la colonne de direction de véhicules motorisés), ainsi que dans le domaine des cartes à puce pour démontrer la conformité des spécifications à la norme critères communs au niveau 5+ (utilisée pour la sécurité des systèmes et des logiciels informatiques).

Dans ce document nous avons présenté une approche formelle pour valider une implémentation vis-à-vis d'un besoin fonctionnel et de sécurité. CLEARSY dispose également de références importantes en termes d'analyse formelle de système complexes SIL4, cette fois-ci au niveau inter-système. L'objectif de ce type d'approche est de valider le fait que, si chaque sous-système garantit les propriétés souhaitées (ce que nous tachons de démontrer via la méthode décrite dans ce papier), alors la sécurité globale du système est assurée.

Une telle approche a été utilisée pour démontrer formellement l'anticollision et le non-déraillement de trains au niveau des spécifications inter-système (sous réserve que ces spécifications soient correctement implémentés par le constructeur) pour les CBTC de New-York (Ligne 7) ainsi que pour la RATP (Octys) et enfin le système ERTMS pour SNCF.

ANALYSE FORMELLE DE LOGICIEL



CLEARSY

SYSTEMS ENGINEERING

OFFRE 2019

320 AVENUE ARCHIMEDE
LES PLEIADES III BAT A
13100 AIX-EN-PROVENCE - FRANCE

TEL. +33 (0)4 42 37 12 70 FAX. +33 (0)4 42 37 12 71

WEB. contact@CLEARSY.com
WWW.CLEARSY.COM