Sécurisation de capteurs/actionneurs sur réseau industriel

Thomas Toublanc

THESE DE DOCTORAT DE

L'UNIVERSITÉ BRETAGNE SUD
COMUE UNIVERSITE BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *AST*

Par

« Thomas Toublanc »

« Sécurisation de Capteurs/Actionneurs sur Réseau Industriel »

Thèse présentée et soutenue à « Lorient », le « 18/12/2018 »
Unité de recherche : UMR 6285
Thèse N° : 512

Rapporteurs avant soutenance :

Eric Zamaï	Maitre de conférences HDR, INP de Grenoble
Mireille Bayart	Professeur, Université de Lille

Composition du Jury :

Florent de Lamotte	Maitre de conférences, ENSIBS, Lorient
Armand Toguyeni	Professeur, Ecole Centrale de Lille
Olivier Cardin	Maitre de conférences HDR, IUT de Nantes

Directeur de thèse

Pascal Berruet	Professeur, IUT de Lorient

Invité(s)

Romain Bévan	Ingénieur de recherche, IRMATECH, Lorient

Titre : Sécurisation de Capteurs Actionneurs sur Réseau Industriel

Mots clés : Système Cyber Physique, Détection, Réaction, Aide à la Conception, Simulation Conjointe

Résumé: De nos jours, les systèmes de production sont confrontés à leur 4e révolution. Celle-ci est numérique avec des réseaux toujours plus denses et complexes s'ouvrant sur l'extérieur. Cette ouverture rend ces systèmes plus vulnérables. Les menaces sur ces Systèmes Cyber-Physiques de Production (SCPP) ne sont plus seulement théoriques. L'attaque sur l'aciérie allemande ou le cryptovirus Wannacry en sont de parfaits exemples. Ce travail propose un outil contribuant à la sécurité des SCPP. Nos contributions sont triples :
La conception d'un Système de Détection et Réaction aux Anomalies (SDRA) placé sur le réseau de terrain. Celui-ci intègre des méthodes de détection comportementales et informationnelles. Il comprend également des capacités de réaction à la fois passives, mettant en œuvre de la remontée d'information vers l'humain ou vers des systèmes de niveaux supérieurs, et actives intégrant du filtrage d'ordre ou de la mise en repli.
L'application des méthodes proposées entraîne naturellement un effort de conception supplémentaire qui doit être réduit.
Nous avons donc mis au point une démarche permettant d'assister les concepteurs pour la configuration de notre SDRA.Cette dernière se base sur une approche hybride (composant/opération) et étend un flot de conception existant. Plusieurs transformations raffinent des vues surveillance/supervision des composants alors que d'autres génèrent la configuration du SDRA.
Une troisième contribution propose un démonstrateur réaliste basé sur un environnement virtuel de test. Ce dernier intègre la simulation conjointe de la partie opérative et de la partie commande et permet de montrer les qualités fonctionnelles des solutions face à des scénarios d'attaque ou de défaillance.

Title : Actuator Sensor Securing over Industrial Network

Keywords : Cyber Physical System, Detection, Reaction, Design Assistance, Joint Simulation

Abstract: Today, production systems are facing their 4th revolution. This revolution is digital with increasingly dense and complex networks opening on the outside. This openness makes these systems more vulnerable. The threats on these Cyber-Physical Production Systems (CPPS) are no longer just theoretical. The attacks on the German steel mill or the Wannacry crypto virus are perfect examples. This work proposes a tool contributing to the security of the SCPP. Our contributions are threefold:
The design of an Anomaly Detection and Response System (ADRS) placed on the field network. It integrates behavioral and informational detection methods. It also includes passive response capabilities, implementing feedback to the human or to higher level systems, and active integrating order filtering or fallback.
The application of the proposed methods naturally entails an additional design effort which must be reduced.
We have therefore developed an approach to assist designers in the configuration of our ADRS. It is based on a hybrid approach (component / operation) and extends an existing design flow. Several transformations refine monitoring / supervision views of the components while others generate the configuration of the ADRS.
A third contribution proposes a realistic demonstrator based on a virtual test environment. It integrates the joint simulation of the operative part and the control part and makes it possible to show the functional qualities of the solutions in the face of attack or failure scenarios.

Remerciements

Il me sera très difficile de remercier tout le monde car c'est grâce à l'aide de nombreuses personnes que j'ai pu mener cette thèse à son terme.

Mes premiers remerciements vont à mes encadrants le Professeur Pascal Berruet, HDR, ainsi que le Docteur. Florent De Lamotte, MC, de l'université Bretagne sud, qui m'ont encadré tout au long de cette thèse et qui m'ont fait partager leurs brillantes intuitions. Qu'ils soient aussi remerciés pour leurs gentillesses, leurs disponibilités permanentes et pour les nombreux encouragements qu'ils m'ont prodigués. Je remercie aussi le Dr. Sébastien Guillet et le Dr. Romain Bévan pour leur aide précieuse.

J'adresse tous mes remerciements à Monsieur Eric Zamaï, MC, à l'INP de Grenoble, ainsi qu'à Madame Mireille Bayart, Professeur à l'Université de Lille, de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse. Je tiens à remercier Armand Toguyeni pour avoir accepté de participer à mon jury de thèse et pour sa participation scientifique ainsi que le temps qu'il a consacré à ma recherche. Je remercie également Olivier Cardin, pour l'honneur qu'il me fait d'être dans mon jury de thèse.

Un grand merci aussi à tous les membres de l'ENSIBS et du Lab-STICC et en particulier à Iehann Eveno pour les discutions et l'énergie. Il m'est impossible d'oublier Salwa Alem pour son aide précieuse pour ma recherche bibliographique. Elle a toujours fait tout son possible pour m'aider. Je remercie toutes les personnes avec qui j'ai partagé mes études et notamment ces années de thèse.

Je remercie MON PERE et MA MÈRE également pour tout ce qu'ils ont fait et ce qu'ils feront encore pour moi.

Mes derniers remerciements vont à ma compagne Océane Jézéquel qui a tout fait pour m'aider, qui m'a soutenu et surtout supporté dans tout ce que j'ai entrepris.

Table des matières

Introduction

Nous vivons la quatrième révolution industrielle de l'histoire des systèmes de production. L'invention des premières machines à vapeur au 18e siècle, a permis la mise en place des premières machines de production (machines à tisser). La découverte de l'électricité fournit une nouvelle source d'énergie et aboutit à des équipements moins volumineux. Cela a fait émerger la production en série avec ses nouvelles organisations (Fordisme/Taylorisme/Toyotisme), qui ont favorisé le travail à la chaine. La troisième révolution "électronique et informatique" du 20e siècle est arrivée avec l'automatisme et la robotique. La révolution numérique arrive aujourd'hui avec la digitalisation des systèmes de production et leur mise en réseaux massive.

Les objectifs à atteindre par la numérisation sont variés. Historiquement l'objectif des systèmes de production était le remplacement de l'humain dans les tâches pénibles et/ou répétitives. Ces systèmes devaient également exécuter des opérations impossibles à commander manuellement de par la cadence, la vitesse exigée et/ou le niveau de précision attendu. Ces exigences de qualité de service, *Quality of Services* (QoS) ainsi que de Sûreté de Fonctionnement (SdF) ont ainsi émergé. Les systèmes automatiques devaient permettre d'améliorer la productivité et les conditions de travail ainsi que d'accroitre la sécurité. L'automatisation a permis la discrétisation de l'environnement en variable et a donc assuré une meilleure contrôlabilité des systèmes. De nos jours la digitalisation n'est plus seulement sur l'environnement, mais aussi sur des comportements. En effet les systèmes de production sont de plus en plus flexibles, ils peuvent maintenant réaliser différentes opérations sur un ou plusieurs produits. Cette flexibilité entraine alors divers comportements pour un même système de production que l'on cherchera à contrôler. Cette digitalisation des systèmes de production, avec leur mise en réseaux sous forme de système contrôlé par le réseau, *Network Control System* (NCS), engendre une augmentation de la surface d'attaque avec la généralisation des systèmes de communication pour se rapprocher de l'IT. Les vulnérabilités rajoutées par cet état de fait mènent d'ores et déjà à la violation d'exigences de SdF. Ces exigences devant être satisfaites pendant tout le cycle de vie du système, une surveillance de celles-ci est nécessaire. Cette surveillance doit être établie durant le fonctionnement du système sans le perturber. Elle doit donc être indépendante des éléments à surveiller (contrôle, composant, superviseur). Il est aussi nécessaire d'établir un modèle de sécurité permettant de détecter les violations d'exigence et d'y remédier.

Les experts en sécurité ont en charge dès la conception du système de veiller au respect des exigences de sécurité. Bien que reposant sur des normes communes au point de vue de la SdF, la mise en réseau entraine l'ajout de nouvelles exigences dites de Cybersécurité. Or il n'y a pas vraiment de référentiel commun à ce jour, à l'exception les premiers draft de la norme ISA 62443. Un manque d'outil et de méthode apparait face à ces impératifs de sécurité et de vérification. De plus, de nouvelles failles de sécurité sont utilisées chaque jour et ont déjà ciblé 50% des entreprises dont 77% sont des Petites ou Moyennes Entreprises, *small medium entreprises* (PME). Les répercutions peuvent être dramatiques : -dégradation du produit -endommagement du système -atteinte du personnel -détérioration de la QoS -etc ...

Il est donc nécessaire de disposer de méthodes et d'outils appropriés pour la sécurisation des systèmes, adaptés à ces impératifs. Ceci peut se faire : en ayant une conception sûre avec (pare-feu, réseau local virtuel, *Virtual Local Area Network* (VLAN) ...) pour éviter les attaques, ou au travers l'ajout de dispositif de sécurité pour réduire ou neutraliser les effets.

Contexte

Les travaux présentés dans ce mémoire ont été réalisés dans le cadre d'une Allocation de Recherche Doctorale (ARED). Avec un collaborateur professionnel historique du laboratoire, la société Syleps (PME du groupe Fives de Lorient spécialisée dans la conception et l'implantation de systèmes transitiques et de transtockage). Ils ont été effectués au sein du Lab-STICC de l'UBL. Plusieurs projets ont déjà été réalisés au travers de cette collaboration. Notamment une méthode d'obtention de commandes pour les systèmes complexes a été proposée à partir d'une approche de conception par assemblage successif de *composants* et leurs caractérisations. Ce flot a été proposé par [Lallican, 2007] et complété par un flot de simulation permettant la validation de la commande simple d'un système. Ces flots ont été obtenus en utilisant les principes de l'Ingénierie Dirigée par les Modèles, *model driven engineering* (IDM). Ils ont permis un transfert de compétence au sein du Bureau d'Etude de la Syleps, par l'utilisation de l'environnement SimSED, afin de concevoir et valider les parties commande et opérative d'un système avant l'implantation chez le client. Dans un second temps une méthode pour la commande multi versions des systèmes transitiques reconfigurables, a été mise en œuvre au travers d'un flot de génération implémenté par [Bévan, 2013]. Ce flot permet la génération du programme automate pouvant être chargé instantanément dans celui-ci par les outils propriétaires (Straton, Unity). L'intégration d'outils comme le *Manufacturing Execution System* (MES) pour piloter les systèmes de commande du système de production a aussi été adressée. Ce nouveau flot a été outillé par ComGEM en respectant la même méthode de conception et le même principe que le précédent. Expertisées avec succès ces méthodes ont montré leur intérêt dans l'aide à la conception de programme de commande avec validation avant implantation.

De nos jours, les problématiques de sécurité et SdF nous préoccupent. Nous souhaitons nous inscrire dans la continuité des précédents travaux de façon à rendre plus robuste aux cyberattaques le système conçu. Les attaques peuvent affecter tous les niveaux de la pyramide CIM. Des travaux, notamment sur les réseaux ont adressé les niveaux hauts (Le Système d'Information, *information system* (SI) de la technologie de l'information, *Information Technology* (IT)). Nous souhaitons les compléter en nous positionnant au niveau du réseau de terrain entre les capteurs actionneurs (n :0 du *Computer-Integrated Manufacturing* (CIM)) et l'automate (n :1 du CIM) Nous donnons une représentation du positionnement dans la figure 1.

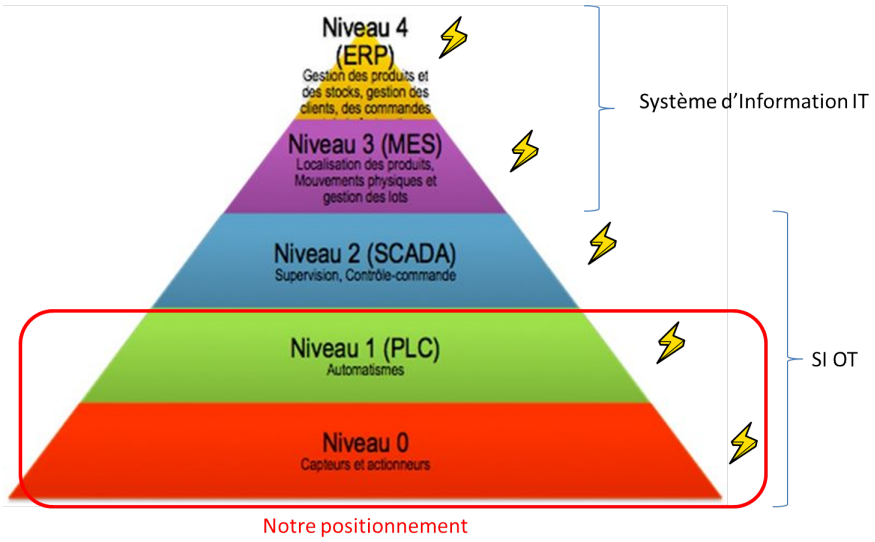


FIGURE 1 – Le positionnement de nos travaux et des attaques

Problématique

Les systèmes de production sont de plus en plus connectés et distribués à travers de denses réseaux. Cette connectivité permet la remontée d'informations à des niveaux d'abstraction toujours plus élevés et distribués (hiérarchiquement et géographiquement). Cet accroissement de la densité, de la complexité et du nombre de points d'accès augmente la surface d'attaque de ces systèmes. Ces attaques sont de diverses natures, allant du déni de service jusqu'à la prise de contrôle du système, en passant par l'exécution de "malware" ou "randsomware" et le vol ou la corruption de donnée. L'une des interrogations que l'on pose est "comment sécuriser un Système Automatisé de Production (SAP) face à ces attaques ?"

Des protections sont déjà présentes à haut niveau, cependant une fois celles-ci franchies le système de contrôle automatisé industriel, *Industrial Automated Control System* (IACS), lui n'en dispose pas ou celles-ci peuvent être contournées. Or l'IACS par nature induit des effets sur le monde physique, allant de la détérioration de produit, jusqu'aux dommages corporels sur les humains, en passant par l'auto dégradation ou la détérioration de l'environnement. L'interface cyber n'étant pas de confiance, une attaque sur celle-ci peut engendrer des retombées catastrophiques sur l'interface physique. De plus l'interface physique peut ne pas être de confiance, car sujette aux attaques physiques. Celles-ci peuvent engendrer des remontées d'informations pouvant corrompre les systèmes de commande et/ou de supervision. Or ce genre d'anomalie n'est détectée que trop tard et/ou reste difficile à diagnostiquer. Dans le monde IT, des stratégies de remédiation existent comme : la neutralisation d'application malveillante ou la sauvegarde et la restauration d'entité. Or dans l'IACS ces stratégies ne peuvent pas ou doivent être adaptées en raison de l'interface physique. Cela nous a menés à poser nos verrous :

1. "Comment sécuriser un IACS ?"
2. "Comment faciliter l'intégration de la cybersécurité ?"
3. "Comment vérifier l'efficacité de la solution et nos apports ?"

Objectifs

Dans ce cadre, une approche globale partant de l'élaboration d'un dispositif de sécurité, puis de la spécification d'une méthode de conception minimalisant le coût de développement de celui-ci, jusqu'à la vérification de son intégration dans un système ainsi qu'une mesure des perturbations que le dispositif occasionne a été proposée.

Nos travaux sont donc triples :

— la conception d'une solution permettant de détecter et réagir à des anomalies,
— l'élaboration d'une méthode pour faciliter la conception du dispositif de sécurité,
— la mise en place d'un environnement permettant de démontrer le fonctionnement et vérifiant les caractéristiques.

Pour ces objectifs :

1. la proposition d'une passerelle intelligente de sécurité sur le réseau de terrain, *Field Bus* (FB), pouvant détecter et réagir à l'occurrence d'une anomalie, pour sécuriser le système face aux effets indésirables de celle-ci.
2. la proposition d'une méthode permettant la génération automatique d'une configuration pour la solution, à partir d'un langage décrivant les systèmes transitiques reconfigurables et en l'étendant aux exigences de sécurité.
3. Enfin l'efficacité de la solution ainsi que de la méthode de configuration devront être confrontées à des cas concrets. Un cas d'étude simulé mettant en avant le besoin de sécurité dans les IACS sera défini. Celui-ci sera mis en œuvre dans un démonstrateur composé d'un environnement de simulation outillé par SimSED, d'un environnement d'émulation reposant sur par le toolkit Straton (IDE et Runtime)et d'un environnement de sécurisation avec la solution. Il permettra d'évaluer la solution selon des métriques propres aux environnements.

Plan

Ce mémoire s'organise en 5 chapitres

— Chapitre 1 : Présentation du contexte général de nos travaux de thèse, des problématiques et besoins liés à la sécurisation de systèmes industriels.

— Chapitre 2 : État de l'art couvrant les moyens de sécurisation pour les SAP, les méthodes de détection d'anomalie et les précédents travaux réalisés au Lab-STICC.

— Chapitre 3 : Présentation de la solution pour détecter et réagir face aux anomalies d'un système (réponse au 1er verrou).

— Chapitre 4 : Proposition d'une méthode de configuration (réponse au 2ème verrou).

— Chapitre 5 : Mise en œuvre du cas d'étude dans un démonstrateur (réponse au 3ème verrou).

Enfin nous concluons ce mémoire et discutons de nos perspectives dans la dernière partie.

1 Les systèmes de production : principes et menaces

« *Logic will get you from A to Z;*
Imagination will get you everywhere. »

Albert Einstein

Sommaire

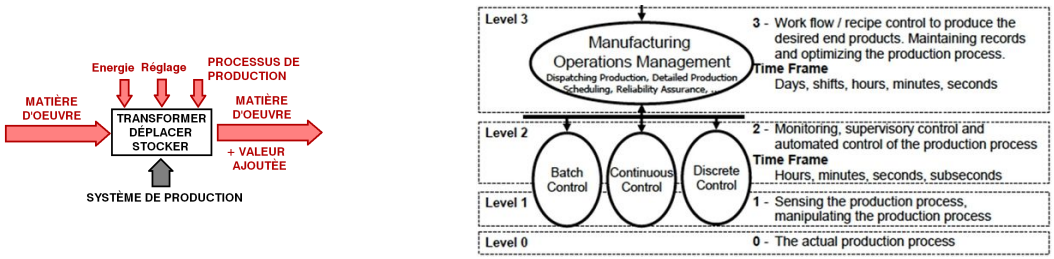
Figures

Tableaux

Ce chapitre propose une présentation du contexte de nos travaux de thèse ainsi que les concepts de base de notre étude. La première section est consacrée aux systèmes de production. Dans un premier temps, nous allons détailler les SAP, en introduisant le paradigme «CIM». Nous décrivons ensuite la cible de nos travaux, le sous-système transitique. Cette description nous amène à caractériser notre environnement d'étude, les Systèmes à Événement Discret (SED).
La section suivante présente les approches de conception du lab-STICC, aux quelle nous donnons suite dans nos travaux.
La menace pesant sur les SAP est présentée dans la troisième section. Elle va nous permettre de définir et caractériser la notion d'attaque dans notre environnement d'étude. Aussi elle fera émerger un constat : "La cybersécurité est difficilement intégrable au système de production".
Enfin, nous aborderons les problématiques et besoins inhérents à la sécurisation de ces systèmes.

1.1 Les Systèmes de production

1.1.1 Les systèmes automatisés de production



(a) Diagramme fonctionnel d'un SP (b) Les contrôles de production

FIGURE 1.1 – Les systèmes de production point de vue fonctionnel et contrôle.

De nos jours, les objets et produits de notre quotidien sont passés partiellement ou totalement par un système de production. Celui-ci a la charge de traiter de la matière d'œuvre pour lui apporter de la valeur ajoutée de façon reproductible et rentable. Les traitements peuvent être du déplacement, du stockage, de l'assemblage, de l'usinage ....
Les systèmes de production nécessitent de l'information provenant d'un système d'information supérieur et de l'énergie, comme présentée en figure 1.1a
Plusieurs types de contrôle de production existent, ils sont présentés en figure 1.1b :
1) le batch répond à un type de procédé industriel semblable à celui de l'informatique. Le produit fini est la résultante d'une série de tâches (transformations) faite par des systèmes sur un lot de produits ex : ligne d'assemblage, préparation de commande ....
2) Le continu, ce second type de contrôle satisfait les besoins d'une production sans interruption ex : textile, pétrochimique ....
3) Le discret, le dernier type de contrôle satisfait les besoins d'une production événementielle.
Le produit fini est la résultante d'une ou plusieurs opérations sur un produit ex : système transitique, transtockage, usinage ....

Un système de production est un regroupement de sous-systèmes. Le premier, le sous-système de traitement transforme la matière d'œuvre en l'usinant ou en l'assemblant. Celui-ci nécessite de la manutention pour gérer les flux de matière, c'est le sous-système transitique. Les deux premiers de par leurs mécanisations et automatisations ont besoin en support d'un sous-système, qui gère les flux d'énergie, fluide, et l'information relative à touts les flux. L'information est une donnée numérique, stockée pour être partagée entre les équipements du SAP, soit : par abstraction et centralisation (base de données MES), ou par collaboration et distribution (réseaux de machines autonomes et collaboratives). Les sous-systèmes de traitement et de manutention de par leur connectivité accrue sont passés de la classe des systèmes physiques à la classe des Systèmes

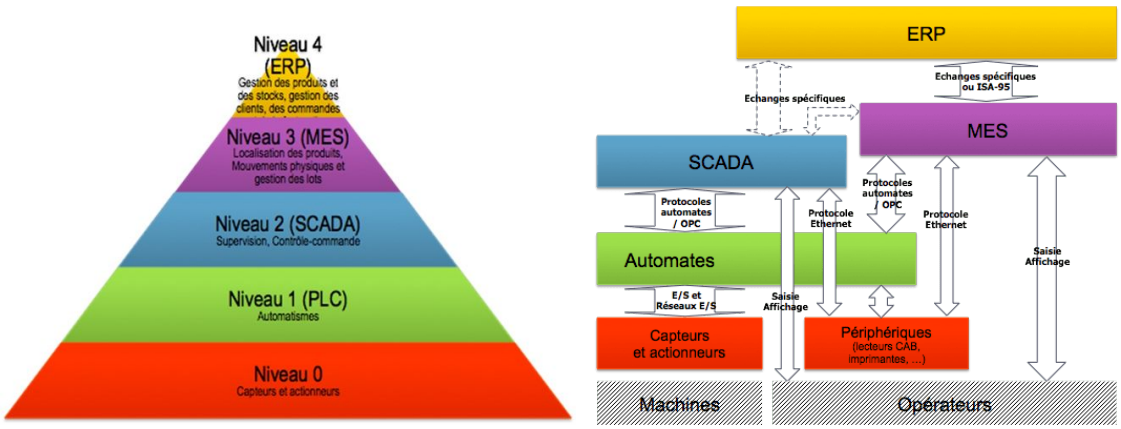
Cyber-Physiques, *Cyber Physical Systems* (SCP/CPS) [Lee, 2006]. La notion cyber vient de la dimension informationnelle qui s'ajoute à la précédente dimension physique, que l'on ce propose d'introduire.

1.1.2 La dimension informationnelle dans les SAP

Les systèmes de production sont de nos jours commandés par l'humain à travers des services cybernétiques implémentés dans des outils de haut niveau (MES, planificateur de ressource de l'entreprise, *Entreprise Resource Planning* (ERP)). L'abstraction engendrée par les outils simplifie les commandes que lui fourni l'humain. Mais elles se transforment à travers les différents niveaux d'abstraction allant de la planification à l'exécution sur le procédé.
Exemple : Un Ordre de Fabrication (OF) est demandé par l'ERP au MES, celui-ci va donc faire des vérifications pour sa faisabilité puis l'ordonnancer. L'ordonnancement va répartir dans le temps les différentes transformations à faire sur la matière d'œuvre et commander les sous-systèmes. Cette commande sera transmise aux contrôleurs des sous-systèmes qui réaliseront les transformations avec les acteurs du processus. Ces mêmes acteurs ont aussi la charge de récolter les informations du processus pour le contrôleur. Celui-ci ensuite transmettra des rapports au MES soit par l'intermédiaire d'un service tiers comme le système d'acquisition et de contrôle de données, *Supervisory Control And Data Acquisition* (SCADA) ou directement.

la dimension informationnelle des SAP actuels est représentée avec le référentiel CIM, qui répond aux problématiques de l'intégration des ordinateurs dans l'industrie. On se propose de l'introduire.

1.1.3 Le référentiel CIM



(a) CIM pyramidale (b) CIM modulaire

FIGURE 1.2 – Une double représentation du référentiel CIM pour l'industrie.

Le concept de CIM a été proposé par [Harrington, 1979] pour mieux structurer le système informatique de l'entreprise, il accroit la productivité dans la fabrication de produit réalisé par une entreprise, mais aussi la gestion et la conception par l'intégration des services de l'entreprise. Bien que quelque peu "académique" vis à vis des réalités industrielles, celui-ci est la référence actuelle. Ce concept peut être modélisé sous forme pyramidale 1.2a pour une représentation hiérarchique ou à plat comme 1.2a afin de voir les modules et leurs interactions (opération). Ce modèle décompose un SAP en 5 niveaux et est prôné notamment par la norme S95 [Brandl, 2000].

— Niveau 4 - Entreprise : Regroupement des services de l'entreprise dans un SI inter sites. À ce niveau, l'outil le plus communément utilisé est l'ERP.

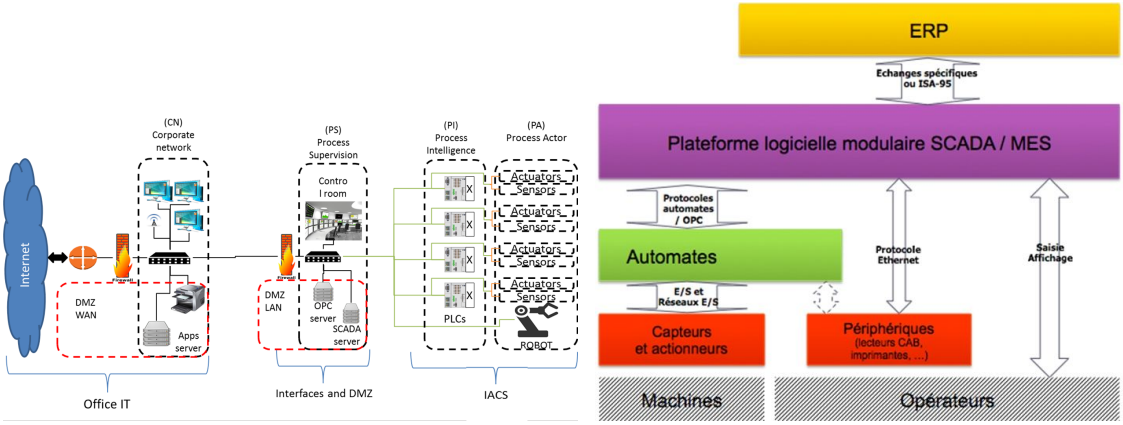
— Niveau 3 - Usine : Regroupement des services d'un site dans un SI intra site. C'est le MES qui est communément utilisé.

— Niveau 2 - Atelier : Supervision de la production pour un site. Divers outils peuvent être implantés comme le SCADA dans un SI.

— Niveau 1 - Cellule : Regroupement d'équipements pilotant la production d'un ou de plusieurs produits (gamme). Plusieurs architectures peuvent alors être utilisées pour ce SI(+/- déterministe) : Hiérarchique, distribuée, centralisée.
Le choix de la "bonne" architecture est imposé par de multiples contraintes : budgétaire, d'exploitation, fonctionnelle ou technologique [Zamaï et al., 2007].

— Niveau 0 - Machine : Regroupement de composants réalisant une opération ou des mesures sur un produit. À ce niveau un SI est aussi présent, mais celui-ci est déterministe.

Cette décomposition fait émerger plusieurs SI que l'on peut regrouper selon leurs caractéristiques. La figure 1.3a nous montre avec un point de vue de réseau, trois grandes catégories :
1) Le SI office IT (Nvx 4),
2) Le SI de gestion & supervision (Nvx 3 & Nvx 2),
3) Le SI de pilotage, contrôle & de terrain (Nvx 1 & Nvx 0).
Ce regroupement est le fruit d'une réflexion sur les caractéristiques des différentes informations.



(a) représentation des SI

(b) nouvelle représentation du CIM d'après [Allot, 2014]

FIGURE 1.3 – Une double représentation des SI d'une entreprise.

Le premier disposera de bases de données volumineuses liant les différents services de l'entreprise ainsi qu'un temps d'accès aux informations de l'ordre de la seconde. Le deuxième se compose de base de données spécifique à la production et le temps d'accès aux informations est beaucoup plus contraint (centaine de millisecondes). Le dernier quant à lui est déterministe avec une garantie d'accès aux informations de manière ordonnée et dans une plage de temps de l'ordre d'une vingtaine de millisecondes.

Les systèmes de production sont de nos jours massivement connectés à travers de denses réseaux, induits par leur étendue géographique et/ou logique. Le paradigme CIM bien que normalisant les flots d'information hiérarchique entre équipements pour la production, ne se préoccupe pas des flots ne s'y rattachant pas ou non hiérarchiques (collaboration). Nos travaux sont principalement dédiés a l'un des sous-systèmes, le transitiques, qui évoluent pour devenir de plus en plus flexible et s'adapte aux productions personnalisables, qui intègre le pilotage par produit et la reconfiguration.

1.1.4 Les systèmes transitiques

La transitique comme définit par [Mouchard, 2002] dans des travaux antérieurs au notre, est le vocable spécifique pour la démarche intégrative des opérations de manutention et de stockage dans les systèmes de production. C'est sur cette classe de système que les précédents travaux (R. Bévan et J.L. Lallican) ont été focalisés en lien avec nos partenaires industriels.

La transitique est présente dans notre vie quotidienne et a tous les stades de la vie économique : extraction, transformation, production, stockage, distribution, consommation, recyclage ... Aussi elle représente 80% du parc d'équipements de toutes les industries, elle est donc omniprésente. Certains SAP ajoutent de la valeur au produit uniquement avec la manutention et le stockage ex : tri de bagage aéroportuaire, tri de courrier, stockage en entrepôt ...
Les tâches qui incombent à ces systèmes sont typiquement : du convoyage, du stockage, de l'identification.

— le convoyage, fait transiter de la matière d'un point A à un point B en respectant des contraintes temporelles et à travers un réseau de manutention,

— le stockage, bloque sur ou en dehors du réseau de manutention de la matière,

— l'identification, gère et pilote les flux de matière et produits.

L'industrie de la manutention offre de nombreuses solutions qui ont pour vocation la gestion des flux physiques et l'information qui les accompagnent. Les systèmes transitiques dont un exemple est donné en figure 1.4 ont comme fonction première le déplacement du produit, mais au-delà de ça ils optimisent les flux physiques. Ils sont constitués d'un nombre limité de composants. Ces

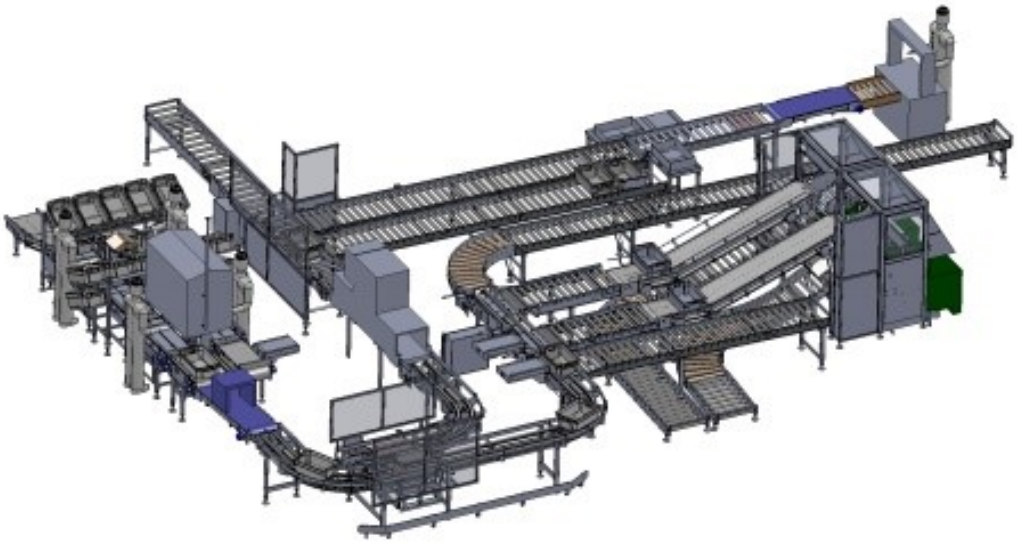


FIGURE 1.4 – Exemple de système transitique

composants sont principalement des éléments de transport et de stockage (convoyeurs, tables de routage, transtockeurs, vérins pneumatiques, véhicules autoguidés, *Automated Guided Vehicules* (AGV)), mais ils peuvent s'agréger avec des composants de détection et d'identification (Cellule optique, mécanique ou électromagnétique et lecteur de code-barre 1D ou 2D (QRcode), ou la radio identification, *Radio Frequency IDentification* (RFID)).

Les systèmes transitiques sont considérés comme des SED de par leur commande, essentiellement discrète, et de par le flux physique sur lequel ils opèrent qui est relativement décomposable [Mouchard, 2002]. Nous allons introduire les SED.

1.1.5 Les systèmes à événement discret

L'automatique représente le comportement des systèmes dynamiques avec des modèles mathématiques. D'abord, continue grâce à des équations différentielles ou des dérivées partielles, puis avec un référentiel discret décrit la commande de ceux-ci. La finalité principale est l'identification de phénomènes physiques. Cependant cette représentation bien que fidèle, est moins adaptée lorsque le comportement réagit à des événements ponctuels.

La notion de SED [Cassandras and Lafortune, 2009] permet de modéliser des systèmes informatiques, embarqués ou de production et des réseaux de communication ou de transport (logistique -> transitique). Un SED doit satisfaire deux propriétés :

— l'espace d'état de celui-ci est discret,

— la transition entre deux états est déclenchée par un événement.

1.1.6 Synthèse et enjeux

Nous avons introduit les systèmes de production ainsi que le référentiel CIM. Puis nous avons détaillé la cible de nos travaux, les systèmes transitiques, que nous considérons comme des SED faisant partie d'un SAP.

Nos travaux font suite à ceux de R. Bévan dans le cadre des approches de conception. Notamment sur la génération et la simulation de contrôle et de configuration MES, pour les systèmes transitiques, que nous présentons dans la section suivante.

1.2 L'approche de conception du lab-STICC

Nos travaux visent à sécuriser la partie opérative d'un système de production, considéré comme un SED.

Au sein de notre laboratoire, l'étude de ces systèmes se fait dans le cadre d'une approche outillée développée par R. Bévan pendant sa thèse [Bévan, 2013] et synthétisant les résultats de travaux précédents [De Lamotte, 2006], [Lallican, 2007]. R. Bévan propose une approche hybride (Bottom/Up, Top/Down), pour décrire un système de production reconfigurable à bas et haut niveau. On peut placer ces travaux dans le niveau 2, 1 et 0 de la pyramide CIM cf. figure 1.3.

La finalité de ce travail est la génération du code de contrôle/commande multi version, en prenant en compte la reconfiguration du système par le MES.

1) R. Bévan à étendu l'approche de [Lallican, 2007] par son DSL pour la description des systèmes transitiques. Ce DSL définit notamment des typologies pour :

— les composants 1.5a.

— les vues 1.5b.

— les opérations 1.5c.

Puis il a enrichi la méthode proposée par J-L. Lallican. Celle-ci proposait 6 étapes pour qu'un concepteur compose hiérarchiquement un système :

— Identification du produit.

— Définition de la partie opérative par agrégation de composant.

— Caractérisation matérielle et topologique des composants.

— Identification des opérations par agrégation.

— Définition des vues contraintes.

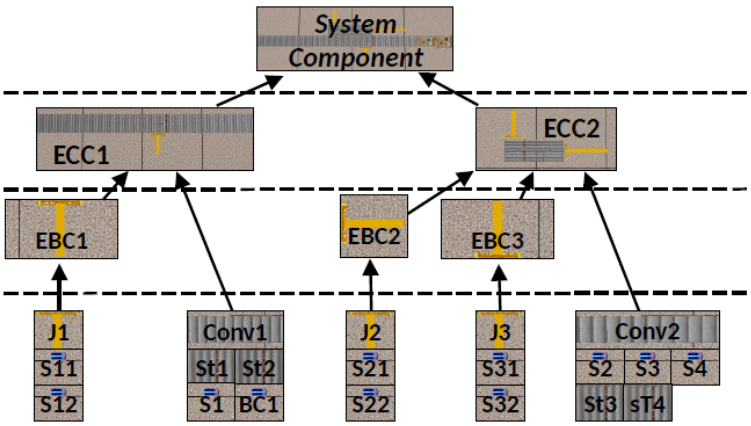
— Définition de l'architecture du système commande (automate, module d'entré/sortie, moyens et protocoles de communication).

R. Bévan y a contribué en ajoutant :

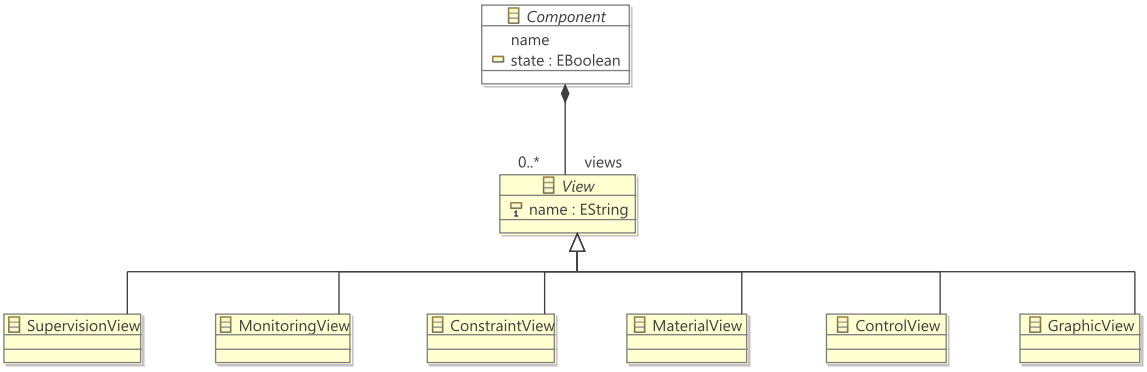
— La définition de nouveaux composants respectant la typologie 1.5a

— La définition de nouvelles opérations respectant la typologie 1.5c

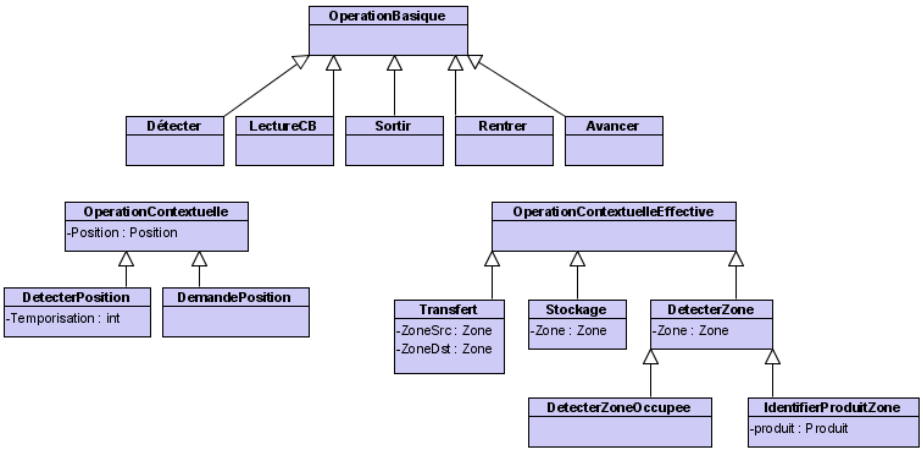
— Le déplacement de la première étape de conception dans une dernière qu'il a ajouté, afin de définir le système de gestion de production.



(a) Typologie des composants



(b) Typologie des vues



(c) Typologie des opérations

FIGURE 1.5 – les typologies pour les composants, les vues (non détaillé), les opérations

Enfin il a intégré les résultats de F. Lamotte, avec la prise en compte d'une gestion de la production (MES) pilotant les opérations. Il a donc créé un nouveau langage de spécification métier, *Domain Specific Language* (DSL), modélisant le MES et ses fonctions comme montrées en figure 1.6.

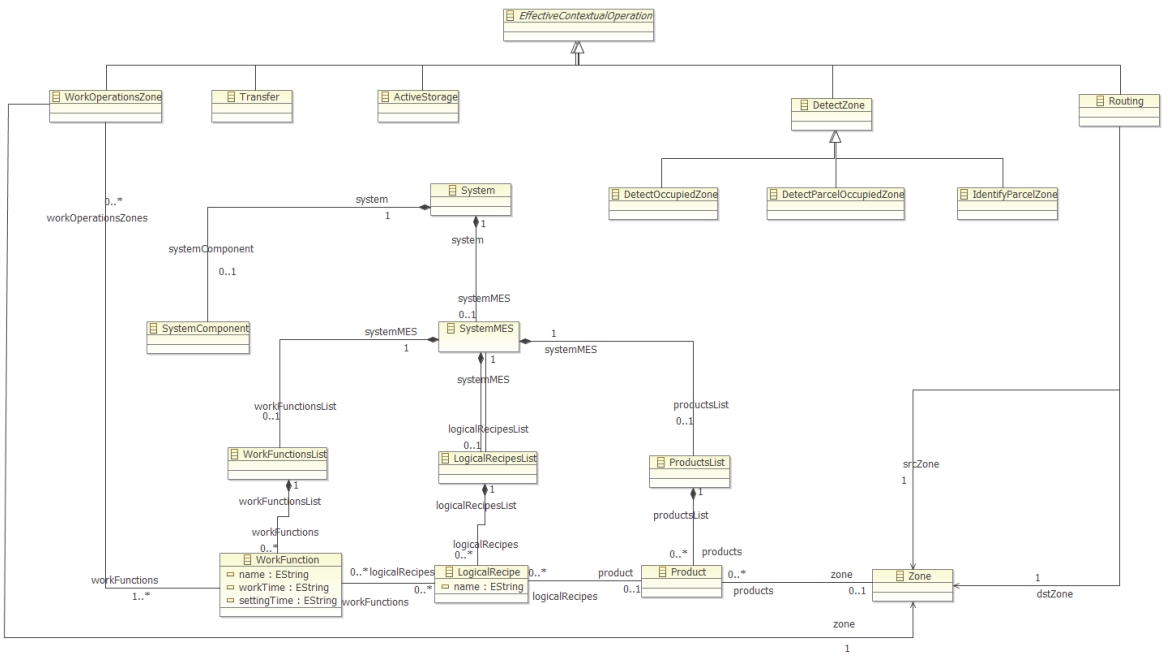


FIGURE 1.6 – Méta modélisation du MES

De plus, deux nouvelles méthodes ont été mises au point :

— l'une génère les informations liées au MES (gammes et routage), qui s'inspire des travaux de routage d'Internet.

— l'autre de niveau coordination génère le code de contrôle/commande pouvant être chargé dans un environnement de développement pour automate.

Trois niveaux de reconfiguration ont été définis et des mécanismes adaptés ont été mis en place. Ces mécanismes tente d'apporter des réponses aux besoins et impératifs croissants des systèmes, en termes de sûreté de fonctionnement, de réactivité et d'évolution face à la demande qui avaient été mis en évidence par [De Lamotte, 2006].

Cette approche a été implémentée dans un flot de conception outillé présenté en figure 1.7. Le flot est décomposé en trois phases avec une boucle d'itération :

— la conception (design), qui par étape successive permet aux utilisateurs de définir les composants d'un système selon des points de vue.

— la configuration (configuration), qui exploite la transformation de modèle pour générer des contenus aux outils modélisés (MES, environnement de développement intégré, *Integrated Development Environment* (IDE)).

— la simulation (online), qui permet la validation du code de contrôle/commande par simulation, ainsi que de reboucler sur la phase de conception.

Dans le cadre de l'industrie du futur (Industrie 4.0), les entités d'un système de production sont de plus en plus connectées et doivent être de plus en plus sécurisées. Or comme montré dans la figure 1.7, dans la phase de simulation d'un système réel il y a peu de sécurité au niveau de l'IACS. Dans nos travaux nous privilégierons l'intégrité du système et la sécurité des personnes et des biens qui interagissent avec lui. Dans ce contexte l'hyperconnectivité engendre une menace vis-à-vis de la sécurité que nous allons présenter dans la prochaine section.

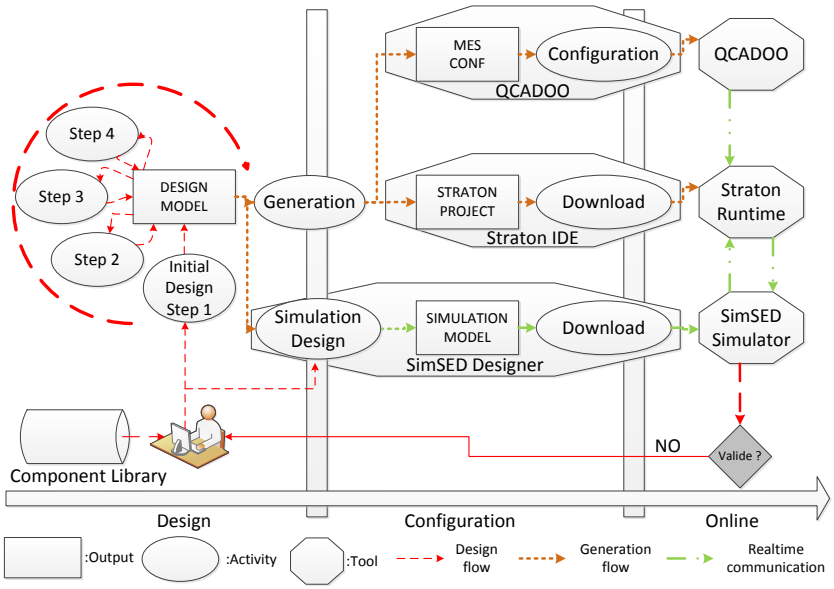


FIGURE 1.7 – Flot de conception outillé avec ComGEM

1.3 La menace

Les systèmes de production sont sujets aux attaques informatiques.

Cette menace est présente pour tous les équipements de l'entreprise. Cependant les techniques classiques s'appliquent dans les hauts niveaux, car les entités sont protégées par des pare-feu et d'autres dispositifs de l'IT.

Au niveau de l'IACS, très peu de solutions garantissant des exigences de sécurité face à cette menace sont présentes, ou celles-ci ne prennent pas en compte l'environnement de la technologie des opérations, *Operation Technology* (OT) des entités.

Or celui-ci permet de spécifier des comportements physiques pour notre système transitique. L'effet d'une attaque conduit a un comportement anormal du système, mais cette attaque peut aussi être physique (venant du terrain) que cyber (venant des réseaux). En effet les entités d'un IACS disposent de ces deux interfaces qui peuvent les engendrer.

Cette disparité de la protection à travers les niveaux hiérarchiques. Entraine des anomalies symptomatiques des attaques à bas niveau. Analysons donc les caractéristiques des attaques.

1.3.1 Les attaques

Une attaque selon l'Agence Nationale pour la Sécurité des Systèmes d'Information (ANSSI) est : «une tentative d'atteinte à des systèmes d'information réalisée dans un but malveillant. Elle peut avoir pour objectif de voler des données (secrets militaires, diplomatiques ou industriels, données personnelles bancaires, etc.), de détruire, endommager ou altérer le fonctionnement normal de systèmes d'information (dont les systèmes industriels)» [ANSSI, 2013]
C'est aussi : «l'emploi de capacités cyber dans le but 1er d'atteindre des objectifs dans ou par le cyberespace [...], utiliser des ordinateurs en réseau dans le but de perturber, interdire, dégrader, manipuler ou détruire des informations dans le système d'information cible»
Sur les systèmes cyber l'impact est plus ou moins important selon l'information contenue. Mais sur les systèmes transitiques, les effets sont bien plus catastrophiques, car ils sont en interaction avec les flux de matière et les systèmes de transformation.
Une attaque sur un Opérateur d'Importance Vitale (OIV) comme une centrale nucléaire, une grande industrie du CAC40, une industrie sévézo, peut engendrer des pertes humaines, matérielles et im-

pacter l'environnement en plus des effets cyber.
Aussi les attaques sont plus ou moins complexes donc difficilement quantifiables ou qualifiables, mais elles sont caractérisées.

1.3.1.1 Caractéristiques d'une attaque

1. Orientée & intentionnelle : un attaquant a un but à atteindre en employant les différentes méthodes sur des cibles.
2. Nécessite la connaissance du système / d'une partie : l'attaquant doit connaitre le système ou les mécanismes qu'il implémente pour choisir les méthodes et les cibles.
3. Non quantifiable (statistiquement) : on ne peut pas faire appel à des lois probabilistes, car certains paramètres sont inconnus cependant d'autres sont déjà définis ainsi que des méthodes :
   — La surface d'attaque : c'est l'ensemble des points d'entrée d'un système qu'un attaquant peut utiliser.
   — La potentialité d'une attaque : c'est un indice allant de 0 → 4 définissant la probabilité qu'une attaque survienne
   — Le rapport gain/risque : est l'évaluation de ce que l'attaquant gagne par rapport à ce qu'il risque.
   Exemple : Le piratage du serveur web d'une petite entreprise qui produit des T-shirts par rapport à l'amende et la peine de prison. « Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données est puni de cinq ans d'emprisonnement et de 75000 euros d'amende » [pei, 2004] bien sûr le pirate aura des motivations (argent, veut nuire, veut faire l'intéressant), mais il les mettra toujours dans la balance avec les risques.

La méthode d'évaluation des risques la plus utilisée en France est [EBIOS, 2010] créée en 1995 par la DCSSI et remise au goût du jour par l'ANSSI en 2010. Cette démarche outillée instrumente les caractéristiques afin d'évaluer le risque et les conséquences d'une attaque sur l'entreprise.

Un travail de classification réalisé par [Papp et al., 2015] permet d'établir des cartographies d'attaques. Celles-ci nous montrent lorsque l'on y superpose des attaques recensées quelles sont les voies les plus couramment utilisées. Comme nous le montre la figure 1.8.

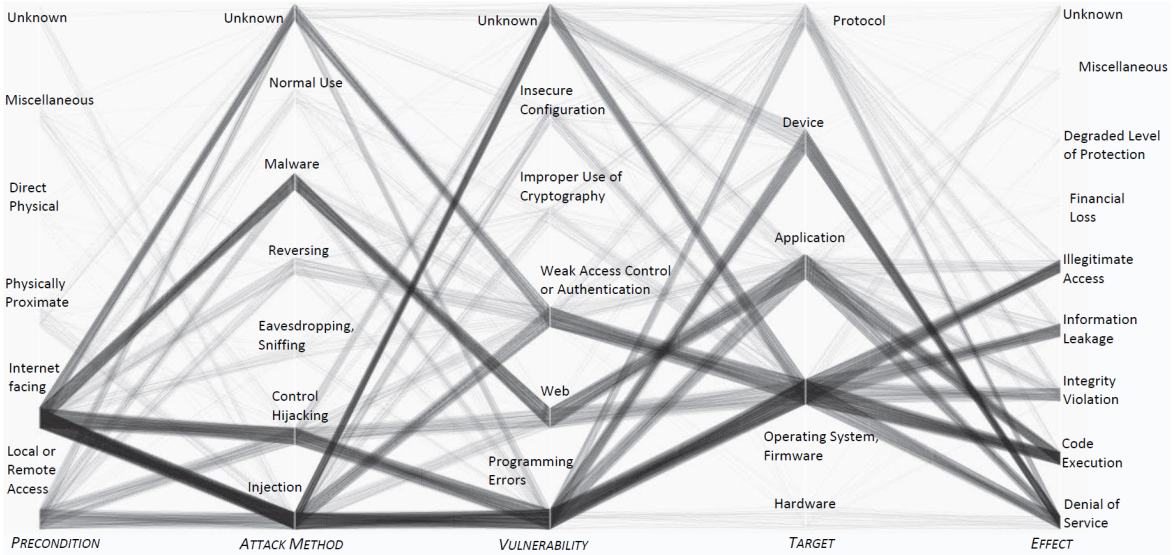


FIGURE 1.8 – Cartographie d'attaque sur les systèmes embarqués [Papp et al., 2015]

Nous avons repris la cartographie de la figure 1.8 en l'adaptant de manière spécifique à l'environnement OT donnée en figure 1.9. Nous nous intéressons plus particulièrement aux effets physiques induits par le vol, la modification ou le blocage de donnée. Une attaque s'établit sur un support cyber et induit un effet sur une cible qui provoquera un effet physique sur une entité en interaction avec la cible.

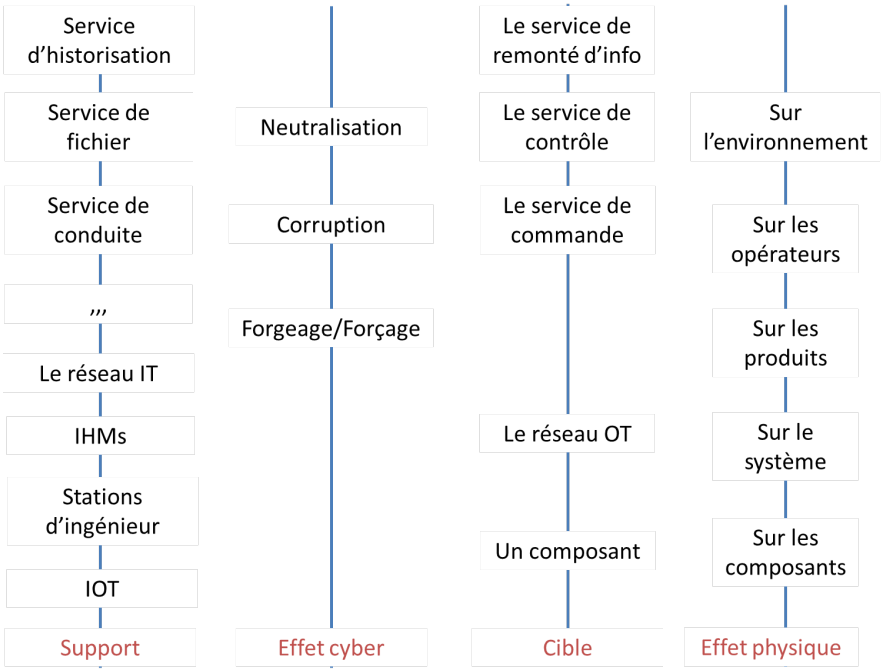


FIGURE 1.9 – Cartographie d'attaque ciblant les systèmes de production

1.3.1.2 Analyse de trois cyberattaques

On se propose d'analyser trois cas d'attaque dans le domaine manufacturier, afin de les mapper sur notre cartographie, mais aussi chiffrer leurs caractéristiques pour les comparer et établir des catégories d'attaques.

L'attaque Stuxnet [Falliere et al., 2011] a été l'une des plus complexes à ce jour. Sans pour autant causer de victimes ou de dommage, elle a fortement diminué la production du système visé. Aussi cette attaque était la première du genre qui ne cible qu'un seul système malgré une contamination première étendue. Les étapes de l'attaque ont été :

1. la propagation d'un ver par des médias USB
2. l'identification par celui-ci du système visé ou la poursuite de la contagion
3. l'ouverture d'un canal de communication dans le système visé
4. la mise en place d'un malware modifiant les informations remontées et reçues par la Partie Opérative, *operational part* (PO) (chemin plein rouge sur l'image 1.10)

Le résultat a été une augmentation de la vitesse de rotation localisée à certaines ressources, mais qui se propageait aléatoirement. Le résultat fut un système de production globalement perturbé, par la mise en maintenance programmée des ressources et un service de maintenance inapproprié (chemin en pointillé rouge sur l'image 1.10). Comme on peut le constater, cette attaque nécessitait une grande connaissance de la cible ainsi que de grandes compétences en développement et stratégie. D'ailleurs l'entité à l'origine de cette attaque n'a toujours pas été clairement identifiée et ne le sera sans doute jamais.

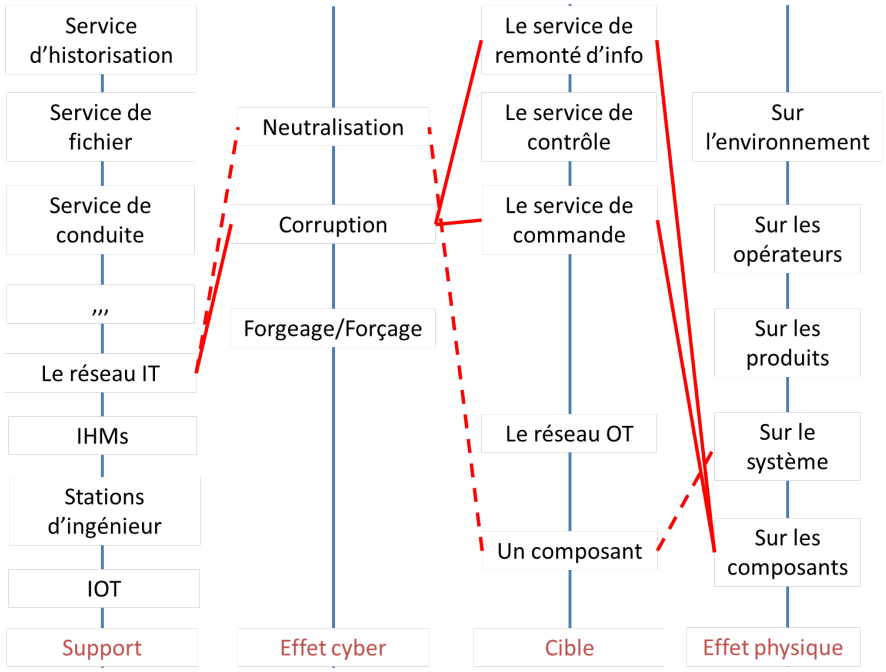


FIGURE 1.10 – Mapping de l'attaque Stuxnet

L'attaque du four d'une aciérie allemande [Robert M. Lee, 2017] fut la plus coûteuse dans le domaine industriel. Elle a détruit un four sidérurgique, d'une valeur estimée à plusieurs millions d'euros, à cela il faut ajouter la perte de production engendrée. Bien qu'ayant causé plus de dommage cette attaque fût moins complexe que la première. L'attaque était ciblée et localisée sur les contrôleurs du four qui étaient connectés comme illustré en figure 1.11.

Wanacry [CERT, 2017] ne ciblait pas spécifiquement les industries cependant, ce sont elles qui ont été les plus lésées. En effet ce crypto virus exploitait une faille de sécurité dans le système d'exploitation, *Operating System* (OS) lui permettant de se propager. Mais surtout de crypter l'information de l'entité infectée. Or les automates, ordinateurs, l'IIoT et même les robots (toutes les entités d'un CPS) y étaient sensibles. Donc beaucoup de ressources d'entreprise ont été cryptées comme illustré en figure 1.12. Les entreprises alors disposaient de plusieurs solutions non exclusives : Payer les pirates pour avoir la clef de décryptage. Attendre le patch permettant de combler la vulnérabilité. Attendre une solution de décryptage. Remplacer tous leurs équipements infectés. Faire fonctionner les équipements sains sans réseau (Internet).

1.3.2 Synthèse

Nous allons comparer les attaques avec un niveau supérieur d'abstraction : Notre évaluation va porter sur la sévérité d'une attaque, qui est difficile à évaluer, car plusieurs critères doivent être pris en compte : l'effet de l'attaque, la stratégie, la discrétion, l'étendue. Nous avons analysé les rapports et distribué des points en prenant comme paramètre : le coût de l'attaque, le nombre de cibles infectées dans différents SI, le plan d'attaque globale, la durée avant détection.
On constate dans le tableau 1.1 que Stuxnet et Wannacry ont un total bien plus élevé que l'attaque

TABLEAU 1.1 – Tableaux comparant des attaques selon : la stratégie, la discrétion, les effets, l'étendue

Dénomination	l'effet	l'étendue	la stratégie	la discrétion	total
Stuxnet	2	1	10	9	22
Aciérie allemande	8	1	2	1	12
Wannacry	5	8	2	5	20

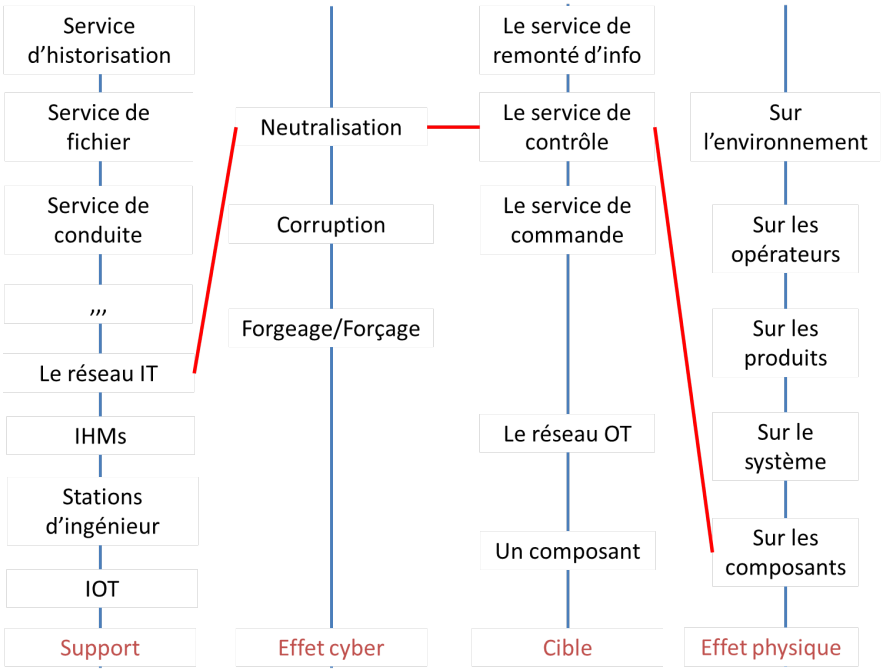


FIGURE 1.11 – Mapping de l'attaque de l'aciérie allemande

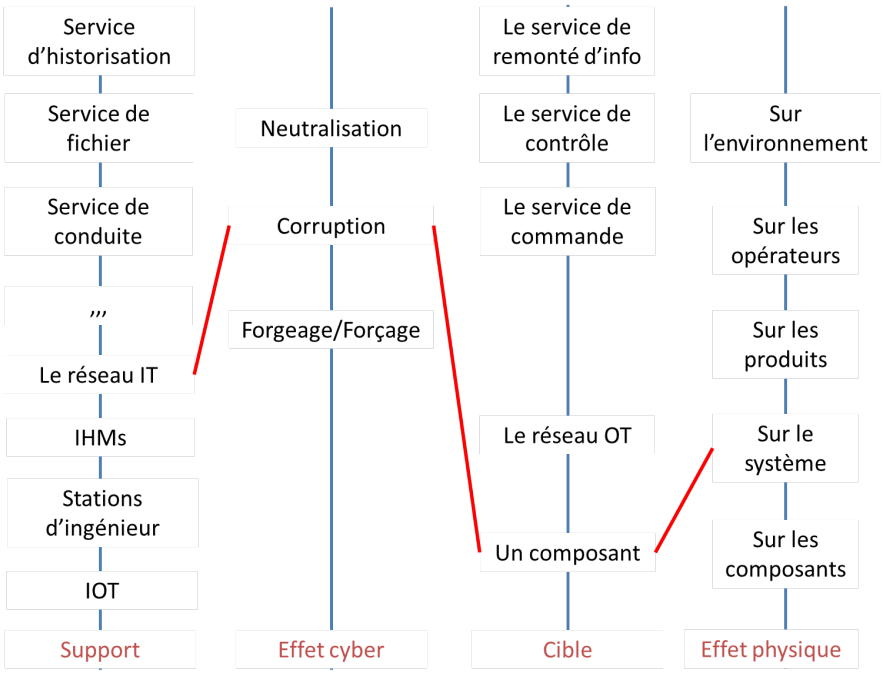


FIGURE 1.12 – Mapping de l'attaque Wanacry

de l'aciérie allemande. Nous faisons une distinction donc entre des attaques majeures qui ont un total supérieur/égale à 20 et les mineures avec un total inférieur. Cette distinction est principalement due au niveau de connaissance en informatique et du système ciblé.
En effet pour Stuxnet et Wannacry, les deux connaissances ont été mobilisées par les attaquants en trouvant une faille 0-day ou par la stratégie mise en œuvre. Pour l'aciérie allemande, l'attaque bien qu'ayant causé des dommages considérables n'a pas nécessité une grande connaissance du système ou de l'informatique, car les équipements de sécurités étaient connectés.
Le CLUb de la Sécurité de l'InFormation (CLUSIF) a analysé plusieurs incidents cybernétiques et les a aussi classés comme montrés en figure 1.13, mais selon deux axes la gravité et la complexité. La gravité étant représentée par l'effet dans notre analyse et la complexité par les trois autres cri-

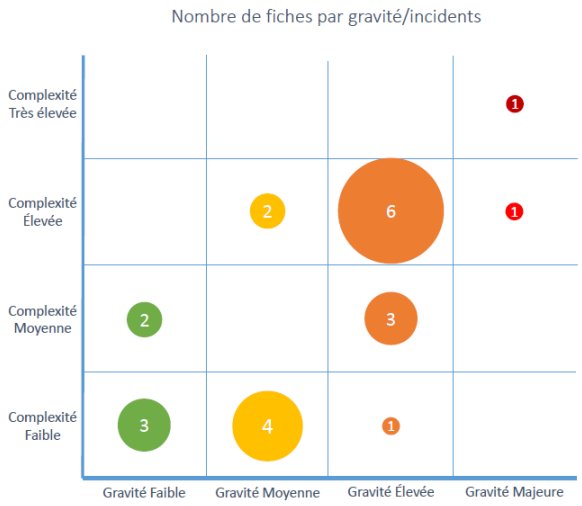


FIGURE 1.13 – Répartition des incidents [CLUSIF, 1992]

tères. Un autre constat est que le nombre d'incidents augmente malgré la mise en place de solution de sécurité comme nous le montre la courbe de la figure 1.14.

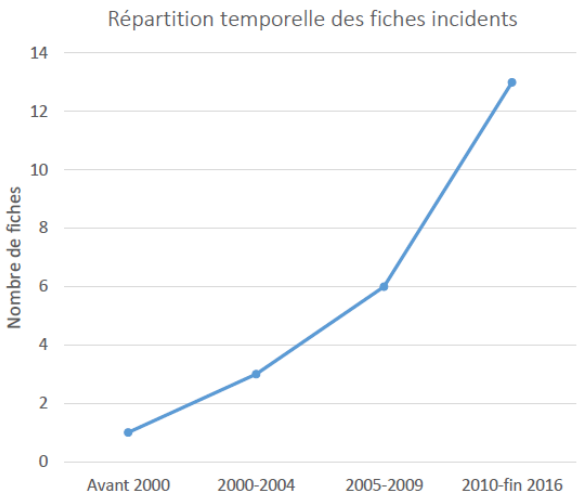


FIGURE 1.14 – Courbe du nombre d'incidents [CLUSIF, 1992]

Dans cette section nous avons établi un constat, les attaques sont en augmentation. Mais aussi qu'elles n'ont pas besoin d'être complexes pour être grave et cause des effets physiques catastrophiques. C'est ce qui a motivé nos travaux sur la problématique de sécurisation de la partie opérative des systèmes de production, que l'on détaillera dans la section suivante.

1.4 Problématique et besoins liés à la sécurité

Plusieurs experts sont en charge de la conception d'un système :
l'ingénieur process fait correspondre les besoins de transformation d'un processus à un cahier des charges ; l'intégrateur trouve des solutions techniques satisfaisant le cahier des charges ; l'installateur réalise l'implantation des machines spatialement et en tenant compte de tous les flux ; l'automaticien se charge de concevoir les commandes et programmes pour les API. Comme on peut le constater dans la figure 1.15, la sécurité est le plus souvent distribuée entre tous les acteurs de la conception. Cependant de nos jours des experts de la sûreté de fonctionnement et du management du risque voient le jour. Ils se chargent de fédérer tous les points de vue tout en proposant des solutions de sécurité.

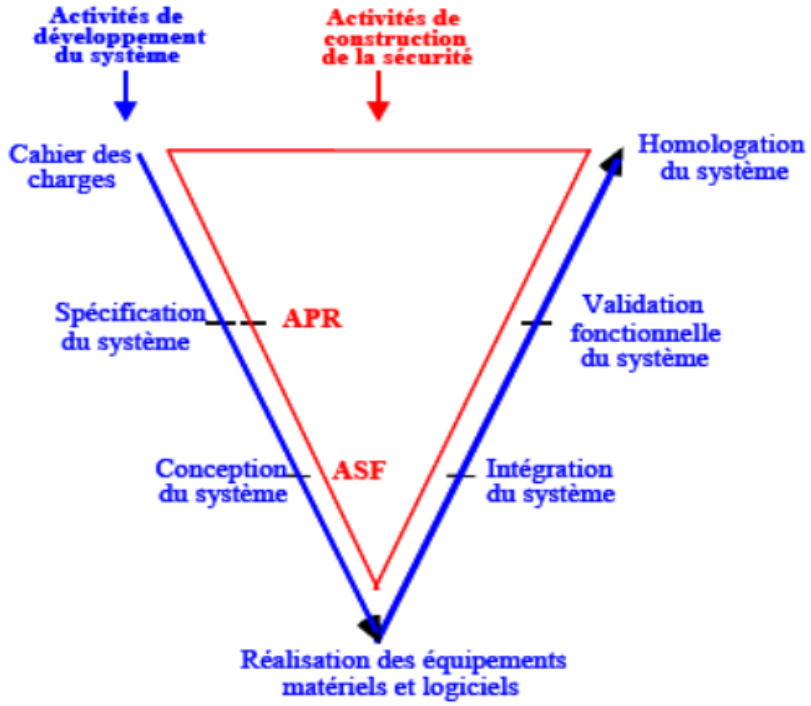


FIGURE 1.15 – Un cycle de développement avec la sécurité intégrée [Mabrouk, 2010]

Ces experts interviennent dès la phase de spécification. Ils doivent relever un défi majeur, le calcul des risques : Trouver l'adéquation entre le risque que le procédé soit : stoppé, dégradé ou corrompu (que le processus s'effectue, mais pas de la bonne manière) et le coût engendré par une solution de sécurité par rapport à celui des pertes estimées. Le risque prend en compte la probabilité d'occurrence et l'impact d'une anomalie sur celui-ci. Cela permet de spécifier temporellement et spatialement où il faut sécuriser le système ainsi que l'investissement qui doit être fourni par rapport au coût si on ne le fait pas. Parallèlement la validation d'une solution de sécurité doit être faite au plus tôt dans le cycle de conception. Il y a donc ici un besoin d'intégrer des méthodes pour les experts en sécurité dès la conception ainsi que fournir des solutions configurables de sécurité. Actuellement les moyens ((prévention, prévision, tolérance aux pannes ...) s'avèrent insuffisants. Dans nos travaux nous cherchons à fournir des moyens supplémentaires.

Dans le paradigme actuel du "toujours plus vite, toujours plus personnalisable et toujours moins cher", les concepteurs de système doivent faire face à plusieurs challenges :

— La complexité : les systèmes de production industrielle sont de plus en plus complexes tant au niveau matériel que logiciel

— L'étendue de la flexibilité : la personnalisation et/ou les gammes étendues de produits nécessitent des processus adaptables voire des usines flexibles. Elles doivent être les plus permissives possible en diminuant les coûts.

— L'hyper connectivité : les systèmes sont de plus en plus connectés entre eux et vers le monde extérieur

— Les impératifs de sûreté de fonctionnement : les systèmes doivent être sûrs, s'auto diagnostiquent et réagissent lors de détection.

Ces constatations ainsi que les différents points de vue d'industriels ont fait émerger une problématique concernant la sécurité des systèmes de production. En effet ceux-ci se rapprochant des SCP/CPS, ils deviennent de plus en plus complexes et connectés. Leur sensibilité aux attaques n'en est que plus augmentée. Les sections suivantes mettent en exergue nos verrous face à la sécurisation des systèmes industriels. La première partie 1.4.1 : traite de la définition de méthodes et d'outils pour sécuriser un système de production.
La deuxième partie 1.4.2 : porte sur la facilitation l'intégration de la cybersécurité.
Pour conclure, la troisième partie 1.4.3 : aborde la vérification de l'apport de nos propositions.
.

1.4.1 La sécurisation de l'IACS

Les IACS sont sujets aux attaques et aux défaillances, ce constat n'en est que plus alarmant avec l'ère du numérique et de l'hyper connectivité.

Une nouvelle architecture pourvue d'entités massivement connectées à travers de dense réseau est intéressante pour y établir un dispositif de sécurité par le réseau. Il faudra cependant veiller à ne pas trop ajouter de latence par rapport au temps de cycle du système commandé.

Aussi des méthodes de détection et réaction devront être adaptées aux ressources informationnelles que le dispositif exploitera. Celles-ci devront être efficaces pour détecter et réagir aux attaques ainsi qu'aux défaillances. Plusieurs approches ou domaines peuvent être exploités, mais nous nous plaçons volontairement au niveau des effets (dans l'environnement OT), comme étant le dernier rempart et le premier observateur.

L'environnement OT permet le rapprochement avec la SdF qui est selon [Laprie et al., 1995] : "une propriété permettant aux utilisateurs d'une entité de placer leurs confiances en elle par rapport à son comportement/service qu'elle délivre."

Ainsi nous sommes complémentaires des méthodes de l'IT, permettant une plus grande couverture des attaques. En effet nous l'avons introduit en section 1.3 les objectifs d'une attaque sont dénombrables, mais pas les moyens d'y arriver.

L'ensemble de méthode étant ciblé, nous devons spécialiser le dispositif de sécurité par rapport à la cible. Les méthodes devront donc s'adapter à la fois aux anciens et nouveaux systèmes.

1.4.2 Faciliter l'intégration de la cybersécurité

La cybersécurité n'est que très peu abordée dans le référentiel CIM. Le modèle de référence pour l'architecture de l'industrie 4.0 quant à lui n'est pas suffisamment mature. La norme ISA [Brandl, 2000] fournit un guide architectural permettant de diminuer l'impact d'une attaque. Cependant cela exige une refonte totale de l'architecture des anciens systèmes en termes de réseau et d'équipement, ainsi qu'un certain niveau de compétences et connaissances pour y parvenir.

Notre constat est que la cybersécurité est difficilement intégrable dans les systèmes industriels. En effet les anciens systèmes ont été conçus sans ces préoccupations et avec une longévité conséquente. Il est difficile maintenant d'y intégrer la cybersécurité, cela peut avoir un coût financier et nécessite des compétences supplémentaires.

Les travaux de ce mémoire vont présenter un dispositif de sécurité adaptatif pour les systèmes industriels. Cependant comme dit précédemment ce dispositif s'appuie sur des méthodes et des ressources afin de protéger un système de production. Or l'une des difficultés est de spécialiser le dispositif pour un système.

Il serait intéressant de proposer une méthode simplifiant l'intégration de la cybersécurité pour les industriels. Ainsi qu'une méthode permettant la génération ou la configuration des solutions de sécurité.

1.4.3 Vérifier l'efficacité de la solution et de nos apports

La vérification d'une solution de sécurité est essentielle et doit se faire dès la conception du système qu'elle protège. Aussi la vérification de toutes les exigences de sécurité doit être faite, en étant le plus proche possible du système à concevoir.

Plusieurs méthodes de vérification peuvent être appliquées :

— la vérification formelle permet de vérifier de manière sûre qu'une solution respecte les exigences spécifiées. Par exemple avec le "model checking", [Clarke and Wing, 1995] nous en fait un état de l'art. [Kesraoui, 2017] l'applique, quant à elle sur un cas d'étude industrielle.

— la vérification par simulation permet de vérifier les exigences une à une en simulant le comportement. L'ouvrage [Law et al., 1991] en fournit une vision globale et [Prat, 2017] nous en montre une application. Toutefois, selon Cardin [Cardin, 2007], cette vérification ne sert pas juste à valider le passage du modèle conceptuel à exécutable. Il s'agit aussi de faire des tests de bon fonctionnement du modèle exécutable et de l'outil de simulation.

— le test grandeur nature permet de vérifier les exigences une à une directement sur le système à protéger.

La vérification formelle vérifie un ensemble de propriétés sur un modèle pour tous les scénarios d'attaque de manière exhaustive. Cependant si l'on veut être sûr du résultat avec une nouvelle propriété on doit revérifier tout le set. Le test grandeur nature quant à lui peut s'avérer destructif pour les équipements et exige du temps pour sa mise en œuvre. Aussi ce test peut être unitaire en cas de destruction malgré le temps passer à le concevoir il est donc coûteux. La vérification par simulation à l'avantage d'être plus ou moins permissive on peut vérifier de nouvelles exigences à la volée. Cependant on ne vérifie que par rapport à un ensemble de scénarios. Aussi ce test étant par simulation il nécessite moins de temps de conception et il est réutilisable

Il serait intéressant de pouvoir vérifier l'efficacité des méthodes de détection et de réaction d'une solution proposée, mais aussi de valider la propriété de non-perturbation du dispositif par rapport au fonctionnement du système.

1.5 Conclusion

La sécurisation des systèmes de production est un enjeu mondial de notre époque. Il commence à se faire sentir de par la recrudescence d'attaque ciblée ou non. Comme celle présentée en section 1.3, le futur est incertain, mais l'évolution inévitable. Les politiques actuelles du "toujours plus" (rapidement et de manière sûre) et du "toujours moins" (coûteux, d'actifs immobilisés), s'ajoutent à la complexification des gammes de produits et des systèmes. Cela renforce le constat suivant : la sécurisation d'un système de production est vitale, elle doit détecter des comportements anormaux et y remédier, elle doit être spécifiée en aidant l'homme à concevoir des systèmes sûrs. Ce chapitre a décrit le contexte général de nos travaux de thèse. Nous avons présenté notre cible le système transitique d'un SAP. Ensuite nous avons présenté un état de l'art local sur les approches de conception. Puis nous avons détaillé la menace qui pèse sur ces systèmes et leurs conséquences. Ce qui nous a amenés à formuler la problématique de la sécurisation de système de production, d'où nous en avons tiré des besoins principaux :

— Définir des méthodes et un dispositif permettant de sécuriser les IACS

— Faciliter l'intégration de la cybersécurité

— Permettre la vérification / validation de l'apport des propositions

Mais avant de vous présenter notre solution ainsi que l'approche permettant de la configurer et sa vérification. Nous allons dresser un état de l'art sur la sécurité, les réseaux et les plateformes de test.

2 État de l'art

« Il y a une synthèse quand, en y combinant des jugements qui nous sont connus à partir de relations plus simples, on en déduit des jugements par rapport à des relations plus compliquées. Il y a une analyse quand d'une vérité compliquée on déduit des vérités plus simples. »

André Ampere

Sommaire

Figures

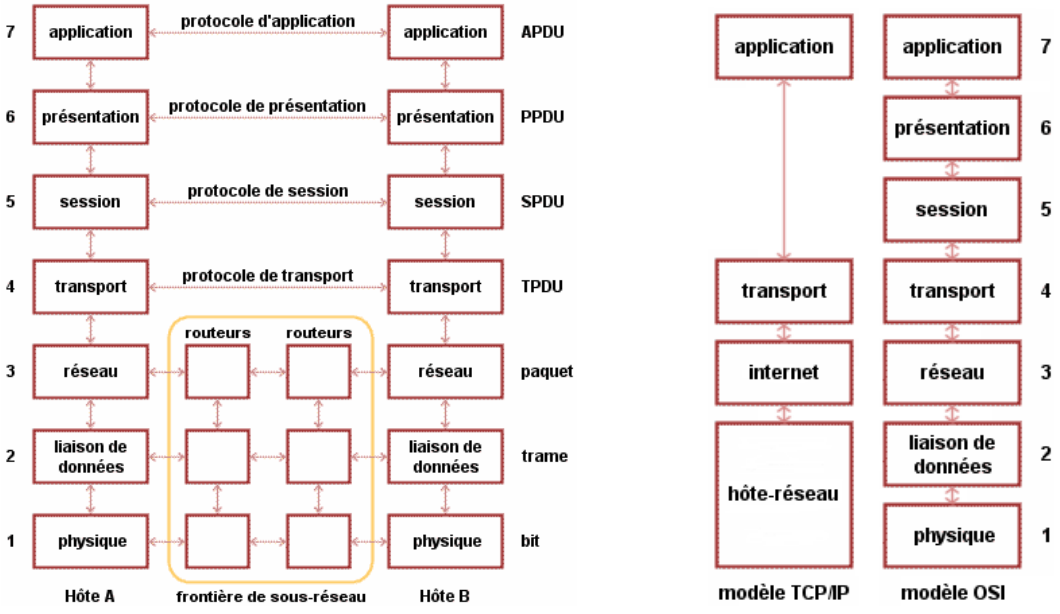
Tableaux

Le chapitre précédent a exposé notre contexte avec l'environnement de recherche et les menaces qui pèsent sur celui-ci. Une attaque est une malveillance intentionnelle et orientée. Dans le monde cyber elle aura comme finalité : le blocage, le vol ou la modification des données. Dans le monde physique pour la partie opérative elle est considérée comme un événement portant atteinte à : la fiabilité ou la disponibilité de l'IACS, l'intégrité physique du système qui la subit ou la sécurité physique des biens et personnes qui interagissent avec le système. Aussi les problématiques et besoins liés à la sécurisation d'un IACS ont été pourvus.

Fort de ce contexte, on se propose ici de dresser un état de l'art couvrant les approches visant à sécuriser un système de production, puis les différents outils ou plateforme de test. Chacun de ces deux axes sera suivi d'une analyse comparative. Afin de déterminer notre placement en terme de protection, de recherche, et de plateforme de test.

2.1 Les protections industrielles pour l'IACS

L'approche basée sur le flot d'information (informationnelle) est massivement utilisée dans les solutions de sécurité pour les SI industriels. Elle pourvoit des méthodes et outils pour détecter des anomalies par le réseau via l'information qui y transite. L'avantage de cette approche est que l'on ne s'intéresse pas à l'information en elle-même, mais plutôt à ses caractéristiques pour transiter sur le réseau. En effet toute communication va nécessiter des mécanismes, on parle de protocoles. Ils vont décrire comment l'information va passer du détenteur de celle-ci aux différents acquéreurs et sous qu'elle forme (comment la comprendre et l'interpréter). Ces différents acquéreurs sont des entités du flot qui peut être modélisé : Le plus connu, générique et historique est



(a) Le modèle OSI à 7 couches (b) Le modèle TCP-IP à 4 couches

FIGURE 2.1 – Les modèles OSI à 7 couches et TCP-IP à 4 couches pour l'intercommunication.

le modèle *Open Systems Interconnection* (OSI) proposé en 1978 [OSI, 2017]. Celui-ci découpe en 7 couches une intercommunication entre des entités ouvertes (dont l'adjonction ou la suppression ne modifie pas le comportement global du réseau). Ce modèle devait logiquement mener à une normalisation internationale des protocoles pour tous les constructeurs d'équipement réseau par la définition des fonctions de chaque couche. Puis l'*International Standards Organisation* (ISO) propose le sien en 1984. La complexité du modèle OSI dans son implémentation ainsi que d'autres paramètres ont fait de lui un modèle de description plutôt que d'exécution. Par ailleurs, grâce à sa simplicité, le modèle *Open Transmission Control Protocol/Internet Protocol* (TCP-IP) [TCP, 2017],

à donné naissance aux protocoles TCP et IP, aujourd'hui utilisés de manière massive grâce à l'éclosion d'Internet.

On peut catégoriser deux types de protocoles :

— ceux de couche basse TCPIP {hôte réseau, Internet} ou OSI {physique, liaison de donnée, réseau}, qui décrivent les mécanismes servant à faire transiter l'information.

— ceux de couche haute TCPIP {transport, application} ou OSI{transport, session, présentation, application}, qui décrivent les mécanismes servant à sécurisé et traiter l'information.

Les méthodes de sécurité employées seront donc différentes selon les types de protocoles et d'interfaces ciblés.
La norme [ISA/IEC 62443-3-1, 2009] ainsi que le guide [Stouffer and Falco, 2006] du *National Institute of Standards and Technology* (NIST) présentent l'ensemble de ces méthodes. Cependant nous allons lier les solutions industrielles de sécurité, aux préconisations du guide de l'ANSSI [ANSSI-GT-SI, 2014] qui en est un sous ensemble. Ces préconisations permettent de conseiller et guider des responsables dans l'élaboration et le maintien de leurs SI.

2.1.1 La gestion des accès

La première préconisation concerne les interfaces du SI. En effet les attaquants ont besoin d'un point d'entré sur le SI, celui-ci peut être physique ou cyber. C'est la précondition d'une attaque comme montrée en figure 1.8, d'où l'importance à donner à cette préconisation. Deux solutions s'adressant chacune à une des interfaces sont passées en revue :

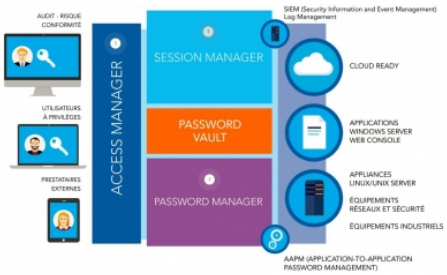
2.1.1.1 Les bouchons Ethernet/USB

Ces bouchons [Rotronic, 2014] dont un exemple, est donné en figure 2.2a, bien que rudimentaire ils sont la première ligne de défense. Ils neutralisent l'ajout de solution servant de support à une attaque, sur des ports d'accès physique du réseau ou des équipements. Les bouchons permettent principalement la diminution de la surface d'attaques d'un SI par prévention.

2.1.1.2 Les bastions numériques

Les bastions [WALLIX, 2018] sont présentés en figure 2.2b, ils sécurisent l'accès à des services ou de l'information par de l'authentification multicritère et d'autres méthodes décrites dans les prochaines sections. Leur rôle est de protéger les parties privées d'un SI, d'attaques venant de l'extérieur en fournissant une ou plusieurs barrières. L'authentification multicritère et le pot de miel neutralisent ou détectent des méthodes exploratoires. Tandis que des barrières comme les pare-feu avec la redirection de port sont aussi mis en place pour réagir. Les bastions diminuent aussi la surface d'attaque, mais face à la menace extérieure en combinant de la prévention ainsi que de la détection réaction.

(a) Bouchon Ethernet sécurisé



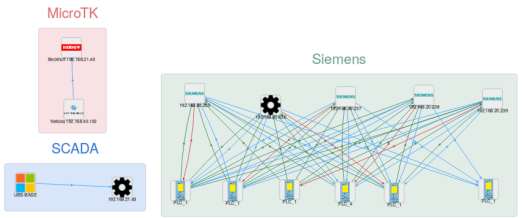
(b) Bastion numérique

Cette préconisation sur la gestion des accès est liée à la prochaine, car c'est en connaissant et gérant sont SI, que l'on peut mieux sécuriser les accès physiques et/ou cyber.

2.1.2 La connaissance et gestion des systèmes d'information

La deuxième préconisation de l'ANSSI concerne toute la cartographie du SI. En effet nous avons pu le constater dans la section précédente qu'une attaque a comme précondition un moyen d'accès. Cependant toute entité du SI peut être considéré comme un moyen d'accès. Il faut donc connaitre et gérer toutes ces entités.

La cartographie [Sentryo, 2000] illustrée en figure 2.3a permet d'acquérir de la connaissance sur les entités et les flots. Aussi elle permet de les représenter graphiquement. Les switchs manageables 2.3b et les pare-feu 2.3c sont des solutions permettant la gestion des flots d'informations.



(a) Cartographie partielle du hall GI de l'ENSIBS

(b) Switch Hirshman

(c) Parefeu Stormshield

FIGURE 2.3 – les solutions pour la connaissance ou la gestion des systèmes d'informations

Une fois toutes les entités d'un système connues, un regroupement et une hiérarchisation de ceux-ci sont nécessaires.

2.1.3 La mise en place de zones et conduits

La troisième préconisation de l'ANSSI concerne particulièrement l'architecture réseau. Elle vise à regrouper des entités dans des sous-réseaux physiques ou virtuels sous forme de zones. Ces zones seront interconnectées par des conduits supportant d'autres solutions de sécurité. Nous présentons deux solutions permettant la mise en place de zones et conduit :

2.1.3.1 Le réseau local virtuel

Un VLAN [Stormshield, 2019] est présenté en figure 2.4, c'est une solution faisant transiter des flots d'information indépendants sur un même média physique avec comme intérêts :

— une meilleure gestion du réseau,

— la segmentation, qui augmente la sécurité avec des ensembles logiques isolés (zones).

2.1.3.2 Le réseau privé virtuel

Un réseau privé virtuel, *Virtual Private Network* (VPN) [VPN, 2017] est décrit en figure 2.5. Cette solution est dédiée principalement aux communications entre entités distantes (conduits).

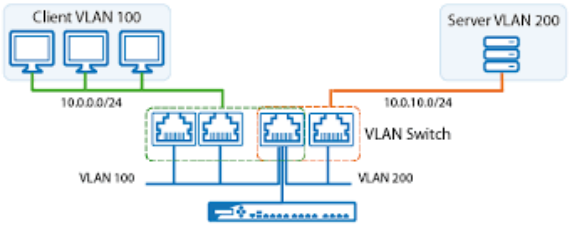


FIGURE 2.4 – Réseau local virtuel (VLAN)

Mais aussi elle permet de superposer les réseaux afin de s'abstraire de la topologie sous-jacente. L'un des avantages est la mise en place du chiffrement sur ce support.

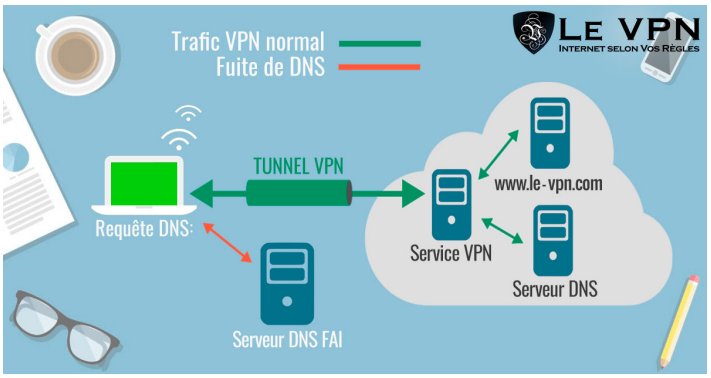


FIGURE 2.5 – Réseau privé virtuel (VPN)

Le chiffrement est une méthode permettant l'authentification de l'information et/ou la vérification de son intégrité. Elle permet via des interfaces de programmation applicatives, *Application Programming Interfaces* (API) spécifiques de vérifier la provenance de l'information ou l'information en elle même. Cette méthode est la plupart du temps implémentée dans les protocoles, exemple : Le protocole *Secure SHell* (SSH) [Ylonen and Lonvick, 2005] qui permet une communication sécurisée par chiffrement entre une machine hôte et une/plusieurs clientes. Ou les protocoles *Secure Sockets Layer* (SSL) et *Transport Layer Security* (TLS) [ssl, 2017] négociant le chiffrement des communications entre client et serveur dès la couche applicative. Ils permettent le chiffrement des communications selon différents concepts cryptographique asymétrique ou symétrique, même sur le réseau local ou la boucle locale de l'équipement. Cependant l'utilisation de proxy SSL comme dans les parfeux Stormshield neutralise le chiffrement à des fins d'analyse de sécurité notamment. Aussi il y a les certificats des protocoles de communication [Heer and Varjonen, 2016], qui vont valider que l'information que l'on reçoit est bien traitée avec un API autorisé sur la machine destinataire via la signature de l'information, ou même d'autres qui vont hacher l'information avec une clé que le récepteur pourra consulter à la réception pour vérifier l'intégrité de l'information [Rivest et al., 1992]. Cependant ces solutions sont utilisées pour se prémunir d'une attaque, elles consomment de la ressource et ajoutent du temps à chacune des communications, ce qui n'est pas forcément soutenable pour tous les équipements d'un SI.

Dans les SI industriels on segmentera d'autant plus les réseaux en isolant les clients des services. La mise en place d'une *DeMilitarized Zone* (DMZ) comme présenté en figure 2.6 permet de sécuriser l'accès au services critique par des clients du haut ou bas niveau du SI. Les VLAN et VPN facilite cette mise en place en simplifiant l'installation physique et le déploiement de parfeux.
La mise en place d'une DMZ facilite l'acquisition de connaissance sur les communications, que nous proposons de détailler.

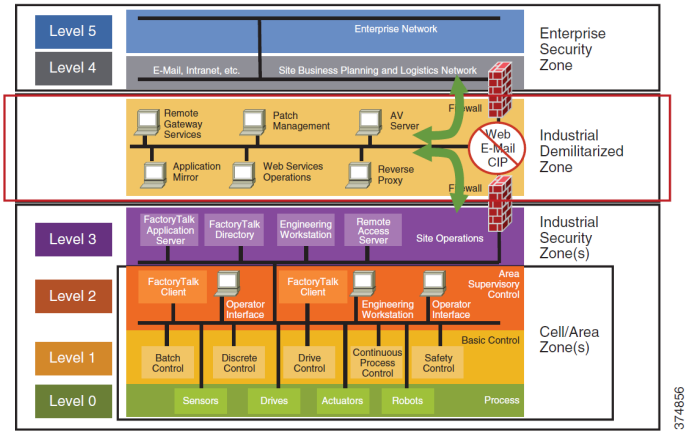


FIGURE 2.6 – DeMilitarized Zone DMZ

2.1.4 La connaissance des communications

La connaissance des communications peut être :

— point de vue de la forme (les différents protocoles encapsulant),
— point de vue du fond (l'information communiquée),
— point de vue du temps (les temporalités).

Les deux premières solutions font intervenir l'inspection profonde de paquet, *Deep Packet Inspection* (DPI), mais aussi les proxys ou méthodes de déchiffrement-chiffrement.

L'inspection profonde de paquet est illustrée en figure 2.7. C'est une méthode utilisée pour classifier des communications. Elle traite et classe par rapport aux protocoles utilisés en dépilant les entêtes d'un paquet selon les couches du modèle OSI. Puis elle analyse les informations contenues dans les paquets. Cette méthode est une référence pour de nombreux "system de détection d'intrusion, *Intrusion Detection System* (IDS)" ou "system de prévention d'intrusion,*Intrusion Prevention System* (IPS)", car elle est la "pierre de Rosette des réseaux". De nombreux outils l'ont optimisé par des implémentations matérielles [Cho and Mangione-Smith, 2004] ou algorithmiques [Dharmapurikar et al., 2003] [Yu et al., 2006] Elle peut maintenant cibler des réseaux denses faisant transiter plusieurs gigabits/secondes d'information. la DPI est même utilisée par certains fournisseurs d'accès pour fournir une sécurité amont [Smallwood and Vance, 2011], mais aussi pour détecter de mauvaises utilisations d'Internet par l'utilisateur comme le piratage de contenu.

Cependant elle doit être couplée avec d'autres pour sécuriser réellement un système, car la DPI est une méthode de lecture et de classification de trafic. D'autres méthodes utiliseront cette classification pour prendre des décisions de sécurité. cependant un troisième point de vue relatif au temps est aussi disponible notamment avec la solution des "baselines"

Les baselines sont utilisées dans l'outil industriel Sentryo par exemple. Cet outil permet de surveiller des réseaux avec des sondes physiques qui observent l'ensemble du trafic. Ces sondes envoient des rapports à une application (le center) qui a une connexion avec une base de données constructeur, cette base de données permet de reconnaitre et interpréter tous les protocoles de transport ou de réseau et même certains applicatifs (DPI). Ainsi les sondes fournissent une cartographie du réseau qui est observé. En combinant la cartographie et les rapports, le "Center" peut alors lier toutes les entités avec leurs communications. C'est là que les baselines jouent un rôle, car la cartographie des communications est historisée. L'utilisateur à partir du "Center" va configurer des références temporelles de communication exemple : "l'hôte 1 peut recevoir des requêtes : *File Transfert Protocole* (FTP) des machines 5 à 10 entre 6 h et 7 h". Cette règle assez semblable à celle

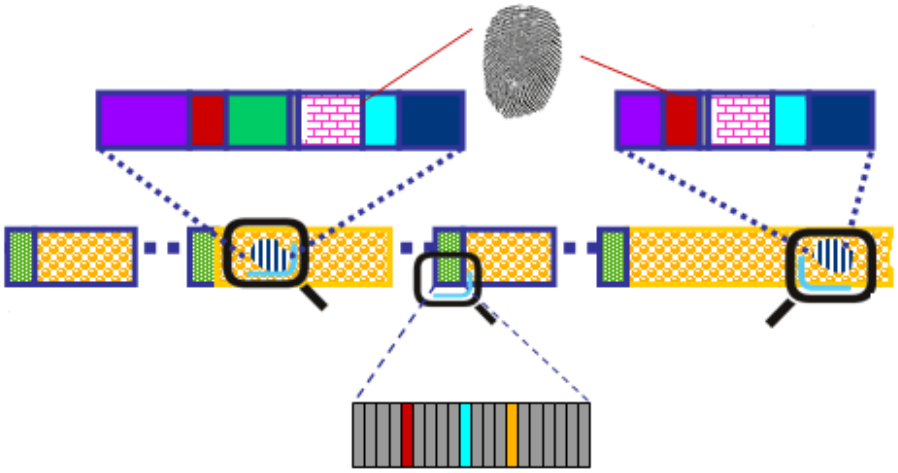


FIGURE 2.7 – L'inspection profonde de paquet

d'un pare-feu a pour but de contraindre l'observation des communications. Ainsi si une communication déroge à cette règle (une autre machine tente d'y accéder ou il y a un accès sur une plage horaire différente. Alors cela sera détecté comme une communication anormale et signalé. Cette méthode utilise le résultat de la DPI en le calquant sur une cartographie et en la comparant avec des contraintes fixées par l'utilisateur. Elle réclame donc une configuration par l'utilisateur qu'il devra la faire évoluer au besoin.

La communication engendre des événements sur les différentes entités qui interagissent avec. Or ces événements peuvent eux aussi fournir des connaissances.

2.1.5 La connaissance des événements

La connaissance des événements est la quatrième préconisation de l'ANSSI. Elle permet de monitorer le passage des communications par les interfaces des différentes entités. In fine grâce au "log" des différentes entités on peut reconstituer tous les tenants et aboutissants d'une communication ainsi que les effets qu'elle a engendrés. C'est le rôle du gestionnaire d'événement et d'information de sécurité, *Security Information and Event Manager* (SIEM) [Bhatt et al., 2014], qui permet à partir de gros volume de données (les logs des entités) une fois ceux-ci interprétés, de générer des rapports et lever des alertes comme représentées en figure 2.8.

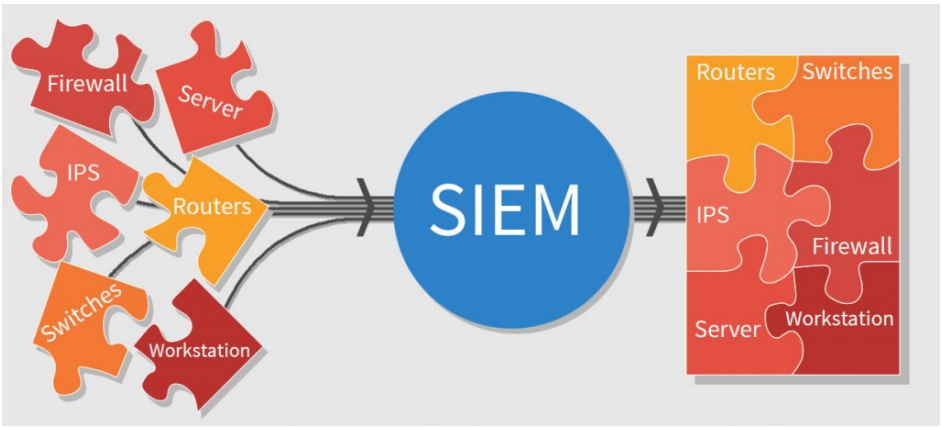


FIGURE 2.8 – Le gestionnaire d'événement ou d'information de sécurité

2.1.6 La gestion des sauvegardes

La gestion des sauvegardes du système est la dernière préconisation de l'ANSSI. Elle est dédiée à la remédiation d'une attaque visant à manipuler l'information du SI. En effet une fois l'attaque neutralisée, le système d'information doit reprendre son fonctionnement avec des informations de confiance. Le réseau de stockage, *Storage Area Network* (SAN) est une solution dédiée à cette méthode et présentée en figure 2.9. La mutualisation des ressources de stockage permet la remédiation en retrouvant les informations de confiance stockées.

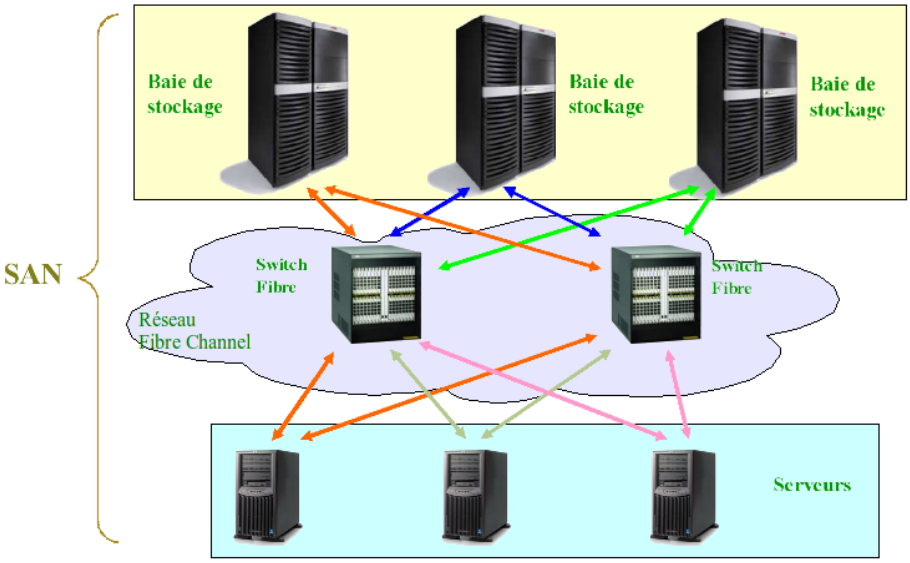


FIGURE 2.9 – Exemple d'un système de sauvegarde redondante

Les protections industrielles tirent partie des avancées et standardisation sur la technologie de l'information. Cependant bien que permettant d'éviter l'attaque ou sa propagation, la détection d'attaques avec une méthode inédite est ardue et nécessite encore aujourd'hui des travaux.

2.2 Les approches académiques

Beaucoup de personnes pensent que l'on peut sécuriser complètement un IACS comme nos systèmes informatiques actuels. L'approche serait d'adapter des technologies telles que les pare-feu, les anti-malware, le monitorage ou la remontée d'information. Mais cela est une erreur, car les particularités des IACS ne permettent pas l'adoption des méthodes classiques. Cet état de fait est détaillé dans l'article [Cheminod et al., 2013]. Donc pour sécuriser un SAP il faut s'intéresser à sa partie cyber/IT, mais aussi à sa partie physique/OT.

Nous détaillerons dans cette section les IDS décris dans [Debar et al., 1999], principalement sur les thématiques de détection et réaction. Il existe une troisième thématique la prévention, qui ne fait pas partie de nos travaux et ne sera pas détaillée. Cette dernière vise à employer des méthodes afin d'éviter qu'un système soit sujet aux anomalies : Par exemple la prévention/prévision de fautes ou avec de l'isolation logique et/ou physique par le réseau.

Dans cette partie, nous allons faire des regroupements :

— les composants de la partie opérative (capteurs, actionneurs, IIOT, robots) sont les acteurs du processus industriel.

— les composants de la partie contrôle/commande sont les intelligences du processus industriel.

Nous les regroupons ainsi par rapport à leurs interfaces, les acteurs disposent d'une interface cyber connecté au réseau OT et physique, alors que les intelligences ne disposent que d'interfaces cyber, mais celles-ci sont connectés aux réseaux OT &/ou IT.

2.2.1 Panorama de la détection

La détection permet d'observer des anomalies. trois types d'observations peuvent être utilisés :

— boite blanche, avec une connaissance de tout le système, son comportement et ses variables.

— boite noire, avec une connaissance mineure du système, seules les entrées et sorties globales de celui-ci.

— boite grise, avec une connaissance de toutes les entrées et sorties des entités du système, mais pas leurs comportements intrinsèques.

Les anomalies peuvent quant à elles être de deux types selon la partie en cause :

— comportementales, issues ou induisant un effet physique

— informationnelle, issues d'une corruption ou d'un blocage dans le domaine cyber

Chacun de ces types d'anomalies fera appel à des méthodes pour la détecter. L'un des points importants est le niveau de confiance. En effet, de nombreux travaux qui seront cités par la suite prennent une ou plusieurs entités de confiance.

2.2.1.1 L'approche comportementale

Ce type d'approche vise à détecter les mauvais comportements d'un système par observation du comportement des utilisateurs (humains ou machines). Elle a été introduite par [ANDERSON, 1980] et reprise par [Denning, 1987] et [Lunt et al., 1989]. Dans l'environnement OT, les ordres/-informations nous permettent de déterminer l'état courant (t) d'un système et l'état suivant (t+1). Les variables de contrôle comme les ordres sont relatifs à l'état courant; les informations sont relatives aux changements d'état. Elles sont données par des entités de même niveau hiérarchique ou inférieur : Automate, capteur ou actionneur. À cela s'ajoutent les niveaux hiérarchiques avec des variables de commande, données par des entités de niveaux hiérarchiques supérieurs : Fournies par l'homme via des Interfaces Homme-Machine (IHM) ou par des services du MES. Toutes ces variables sont des informations transitant sur le SI de l'IACS et représentent l'état du système selon plusieurs points de vue :

— Supervision : ensemble d'informations transitant sur les interfaces cyber du MES ou du SCADA par exemple.

— Commande : ensemble d'informations transitant sur l'interface cyber des PLC par exemple et provenant du niveau supérieur exemple les variables globales de commande.

— Contrôle : ensemble d'informations transitant sur l'interface cyber/physique des PLC par exemple et provenant du même niveau ou inférieur exemple les IO/les variables globales inter automates.

— Physique : ensemble d'informations transitant sur l'interface des acteurs du processus et provenant du niveau supérieur exemple les IO.

La comparaison prédictive détecte en prédisant le comportement du système. Un modèle du système est simulé afin d'observer l'impact d'une mise à jour de variable sur le celui-ci. La simulation donne les variables par pas de temps, donc toutes les évolutions des variables temporellement. La comparaison du set de variables et leur temporalité, à des caractéristiques préétablies, permettent la détection d'un comportement anormale et son échéance. [Lerner, 2015] en est le parfait exemple. Dans ce projet, le but est de sécuriser le comportement d'un IOT (robot) contre 3 grands types d'attaques connues

— corruption d'information

— corruption d'entité

— déni de service

La méthode fait appel à la modélisation mathématique d'un système ainsi que sa simulation. La modélisation décrit le comportement en liant par équation les sorties aux entrées. La simulation permet d'avoir les sorties résultantes aux entrées par la résolution des équations à différents pas de temps temporels. Ainsi en caractérisant avec des paramètres pour borner le fonctionnement normal du système, on détectera une divergence ainsi que son échéance avant même l'envoi de l'information. Une réaction qui sera détaillée dans une autre section 2.2.2.1 peut être déclenchée. Bien que très intéressante en termes de détection cette méthode à deux gros inconvénients : 1)le décalage de tous les envois d'informations dans le temps. En effet, il faut le temps de simuler avant d'envoyer l'information quand elle est valide. Cela pose un problème pour les réseaux de terrain, car ils deviennent non déterministes. Cependant pour sécuriser un réseau hiérarchiquement supérieur (au-dessus du premier niveau d'intelligence), cette solution est tout à fait appropriée. 2)la non-prise en charge d'anomalie provenant du premier niveau d'intelligence (le robot) ou de la partie physique. En effet, les informations à valider sont majoritairement pour commander à hauts niveaux le système. Cela pose un problème, car les informations de bas niveau provenant des acteurs du procédé ne sont elles pas validées. Donc des anomalies issues d'un mauvais comportement physique ne sont pas détectées.

D'autres méthodes plus spécifiques au bas niveau d'un SAP, avec cette même approche seront détaillées dans la prochaine section 2.3. Cependant une autre approche vous est introduit qui complète la première.

2.2.1.2 L'approche par scénario

Ce type d'approche tente de répondre à la question "l'utilisateur a-t-il un comportement normal/anormale?". Elle nécessite donc une connaissance des comportements et de les exprimer sous forme de scénario ou "patterns" comme présenté dans [Spaord, 1994]. Cette approche est aussi appelée approche basée connaissance, car les scénarios servent de références lors de l'observation pour détecter une attaque, cependant il faut les définir avant l'analyse.

La comparaison avec signature détecte en comparant l'état du système avec une modélisation comportementale et temporelle (les Signature Temporelle Causale, *causale temporal signature* (STC)). Une STC est une contrainte fixant un comportement du système dans le temps. La comparaison entre l'évolution du système dans le temps et le déroulement d'une STC précédemment activée, permet la détection et le diagnostic de mauvais comportements dans un SED. Cette méthode décrite dans [Saddem and Philippot, 2014] synthétise les deux premières. Elle valide l'exécution d'une fonction en vérifiant tous les comportements que la fonction occasionne. Aussi en cas d'anomalie elle permet son diagnostic. 3 grands types d'anomalies peuvent donc être détectés. Cependant le diagnostic quant à lui se fera sur les défaillances de la partie opérative.

— corruption d'information

— corruption d'entité

— défaillance de composant

Dans cette section nous avons présenté deux approches de détection pour les IACS. Cependant pour une sécurité optimale la détection doit s'accompagner de réaction (contre mesure). Le lien entre les deux se fera par des moteurs décisionnels, qui en fonction des différents rapports de détection décideront de la réaction appropriée.

2.2.2 Panorama des contres mesures

Les contres mesures dans un IACS ont pour rôle de solutionner une anomalie elles sont présenté dans [Dzung et al., 2005]. Elles sont déclenchées par une fonction de décision et s'établissent sur une des interfaces. Elles peuvent être de deux natures : active et passive. Elles doivent assurer la protection du système contre une attaque. Cette protection peut s'établir selon trois temporalités :

— avant, c'est une réaction qui empêche l'attaque d'avoir des conséquences en l'évitant

— pendant, cette réaction limitera les conséquences d'une attaque
— après, la réaction traitera les conséquences de l'attaque une fois celle-ci neutralisée

2.2.2.1 Réaction active

Les réactions actives vont avoir une influence sur le système en interagissant avec ses interfaces. Elles peuvent garantir un fonctionnement minimal, mais leur exécution a l'obligation de garantir la sécurité du système par rapport :

— à son environnement
— aux personnes qui interagissent avec lui
— aux autres systèmes qui dépendent de lui
— à lui même

Le cloisonnement d'un système consiste à neutraliser le média ou son interface, afin de le priver d'échange d'information vers l'extérieur. Elle sera définie selon la temporalité par rapport à l'attaque.

— avant, c'est l'isolation logiciel et/ou physique des réseaux par exemple avec VLAN ou pare-feu. Elle permet de ne pas propager les attaques d'un réseau à un autre [Chowdhury and Maier, 2010].
— pendant, c'est l'isolation commandée soit par un dispositif tiers comme TAIGA [Lyn et al., 2015].Elle permet d'éviter la propagation lorsque la cible est infectée, mais aussi de maintenir un fonctionnement autonome si elle ne l'est pas.
— après, l'isolation peut être une solution à long terme, pour une entité qui n'a pas la nécessité de communiquer avec d'autres.

Le sectionnement où l'isolation des réseaux peut être faite physiquement avec l'emploi de matériels comme les pare-feu, ou d'architectures DMZ, mais aussi virtuellement avec des VLAN ce qui n'exclut pas la DMZ, celle-ci sera virtuelle. Cette méthode de réaction est efficace, cependant l'isolation peut être impossible du point de vue des impératifs de production Aussi le système isoler devient autonome sans contrôle des niveaux hiérarchiques supérieurs, ce qui entraine une rupture dans la chaine de l'information.

Le forgeage d'information consiste à changer l'information contenue dans une communication ou d'en générer une, afin de modifier le comportement d'un système ou d'un service. C'est normalement une méthode utilisée pour attaquer [ElGamal, 1985], mais elle peut aussi être défensive quand contrôlée. Elle fait appel à l'émulation d'information dans le cas de la remontée ou à la mise en repli pour la transmission d'ordre. Elle n'intervient que lorsqu'une anomalie est identifiée. C'est une réaction qui générera de l'information soit, car il y a un manque, soit pour imposer un comportement.

Le filtrage d'information consiste à empêcher l'envoi d'informations erronées, afin d'éviter un comportement anormal du système [Lhoste, 1994]. C'est une réaction commandée ou une réaction réflexe [Philippot et al., 2014]. Elle est tout le temps active et supervise le système en cas d'anomalie. C'est une réaction qui filtrera toutes les informations pouvant entrainer un mauvais comportement du système, Elle assure donc un comportement normal, peu importe la situation.

Les réactions actives ne permettent cependant pas de diagnostiquer les causes d'une anomalie. D'autres réactions rendant compte de l'état du système doivent être définies.

2.2.2.2 Réaction passive

Les réactions passives n'ont pas d'influence sur le système. Elles ne font que générer de l'information, cependant ces informations peuvent ensuite être analysées afin de déclencher d'autre protection ou trouver les causes de l'anomalie.

La remontée d'information consiste à transmettre en détail les résultats des méthodes de détection ou du monitoring comme présenter dans [Terai et al., 2017], en rendant compte de l'état du système selon différents points de vue, afin d'alimenter un moteur de prise de décision par exemple. C'est une réaction qui passe par des interfaces pour atteindre les entités de niveaux hiérarchiques supérieurs, elle a pour rôle de :

- générer de l'information relative à l'état de l'entité
- mettre en forme cette information en fusionnant des sous-ensembles et en l'adaptant à l'entité qui la lira exemple :
  - un SIEM, qui s'occupera de corréler les différentes informations de "log" pour trouver la cause de l'anomalie (défaillance ou attaque cyber)
  - un outil de diagnostic, qui s'occupera de trouver la cause d'une anomalie (défaillance ou attaque physique)
- transmettre de l'information aux interfaces de niveaux hiérarchiques supérieures.

Cette remontée d'information est événementielle ou cyclique et elle est active durant toute la phase d'utilisation d'un système.

L'alerte consiste à transmettre une synthèse des résultats des méthodes de détection, afin de renseigner l'humain sur l'état anormal du système par des alarmes cette méthode est analyser dans [Wang et al., 2016]. C'est une réaction qui passe par des interfaces pour atteindre l'humain. elle a pour rôle de :

- générer de l'information relative à l'état de l'entité
- mettre en forme cette information en fusionnant des sous-ensembles et en l'adaptant à l'humain. L'adaptation est ici importante, car deux paramètres sont limitants :
  - le nombre d'informations qu'un homme peut interpréter.
  - sa charge mentale lors de la résolution d'une anomalie
- transmettre de l'information aux IHM.

L'alerte permet d'inclure l'humain dans la boucle, afin qu'il puisse mettre à profit ses compétences pour trouver une solution ou la cause de l'anomalie.

Nous avons présenté un ensemble de méthode et d'outil relatif à deux approches de la sécurité la détection et la réaction. L'environnement des SAP industriels nous permet l'implémentation de deux méthodes :

- la comportementale relative à la partie physique, le processus de production.
- L'informationnelle relative à la partie cyber, les informations ou mécanismes de communication.

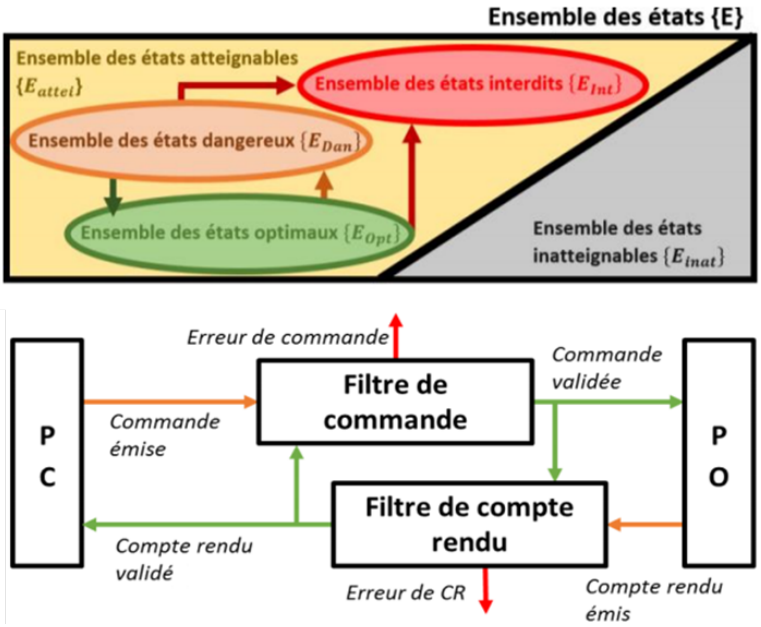
Nous allons maintenant comparer ces méthodes et outils.

2.3 Les approches bas niveau

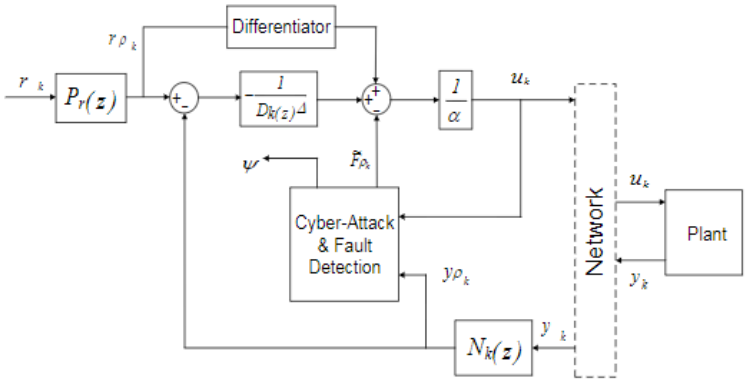
Dans cette section nous allons dresser un état de l'art des approches ayant un contexte similaire aux nôtres. En effet nous nous plaçons à bas niveau entre l'intelligence de processus et les acteurs.

La comparaison avec des ensembles d'états détecte en comparant l'état du système et la distance avec des ensembles d'états définis comme dangereux ou interdits. [Sicard et al., 2017] en est le parfait exemple en figure 2.10a. Ici ils cherchent à sécuriser le comportement d'un SED contre 2 grands types d'anomalies.

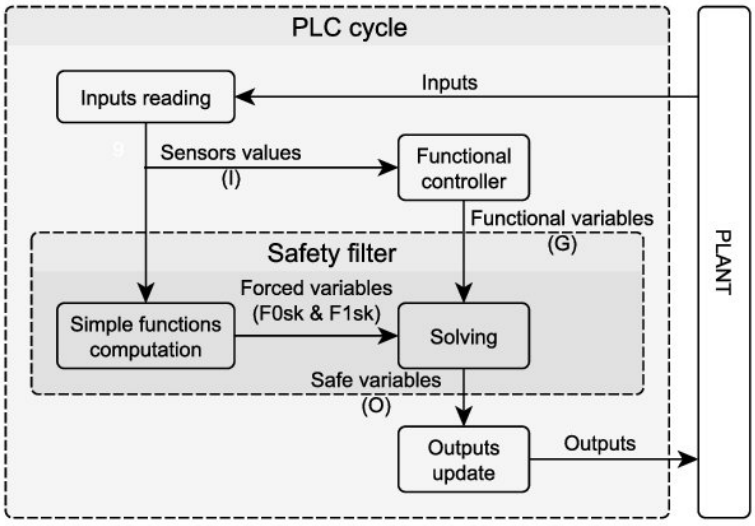
- corruption d'information
- corruption d'entité



(a) Un filtrage par mesure de distance entre états



(b) Un contrôleur prédictif généralisé intelligent



(c) Un filtrage sûr par résolution contrainte

La méthode utilisée passe par la modélisation des comportements du système. La modélisation est l'ensemble des états d'un système, qui regroupe des sous-ensembles définis comme dangereux, interdits et optimaux. Une mesure de la distance entre l'état actuel et les sous-ensembles définis à chaque mise à jour d'une variable (↑1 ou ↓0) est réalisée. Cette mesure permet la prise de décision sur la mise en place d'un filtrage.

La comparaison prédictive Une autre approche est celle de [Yaseen and Bayart, 2017] illustrer en figure 2.10b, qui utilise un contrôleur prédictif généralisé intelligent pour le NCS.

Il est conçu pour résoudre des problématiques de tolérance aux fautes et de détection de cyber attaques. Ce dispositif fait la différence entre des défaillances de la partie opérative et des attaques. L'avantage de cette méthode est la prise de conscience qu'un système industriel peut subir des défaillances et des attaques.

La comparaison à partir de contrainte détecte en comparant l'état du système et la mise à jour de certaines variables face à une contrainte préétablie. [Pichard et al., 2018] en est le parfait exemple en figure 2.10c. Ici ils cherchent à sécuriser le comportement d'un SED contre 2 grands types d'anomalies.

— corruption d'information

— défaillance de composant

La méthode utilisée passe par la modélisation des comportements du système. La modélisation est un ensemble de contraintes, liant la mise à jour d'une variable (↑1 ou ↓0), à l'état d'autres variables du système (état du système). Les contraintes sont spécifiées pour chaque variable et seront à valider à chaque mise à jour de la variable. La validation se fait par comparaison entre l'état du système et la contrainte.

2.4 Comparatif des principales approches de sécurisation

Nous avons synthétisé les outils et méthodes dans la figure 2.11. On peut constater que pour celles basées informationnel, elles sont sur le réseau. Leur placement va permettre la mise en place de remontée d'information. On constate aussi que les réactions sont plus étendues avec une approche comportementale. En effet, l'emploi de ces méthodes implique la connaissance des comportements du système à protéger, donc elles peuvent implémenter des solutions plus "intelligentes". Par exemple : en garantissant une continuité de la production, ou la sécurité en forçant le changement d'état du système pour un état sûr.

De cette synthèse nous avons fait un comparatif avantage/inconvénient pour chaque méthode ou outil industriel dans le tableau 2.1. On peut constater que le firewalling et l'outil Sentryo sont complémentaires. En effet Sentryo augmente la compréhension du réseau et de ce qu'il s'y passe, alors que le firewalling le complète par une protection active du système. Aussi on remarque l'approche de détection de TAIGA est la plus aboutie, cependant la réaction qu'elle engendre n'est pas appropriée pour des systèmes de production. Alors que les approches par signature / contrainte comportementale, ainsi que les réactions de filtrage de commande et de remontée d'information / alerte le sont. Ces approches sont aussi adaptées à la robotique ou la cobotique.

Nous avons présenté un état de l'art sur les approches de sécurisation d'un IACS avec une étude comparative. La validation de notre méthode s'appuyant sur une plateforme de démonstration, une étude comparative de plusieurs plateformes est donnée dans la prochaine section.

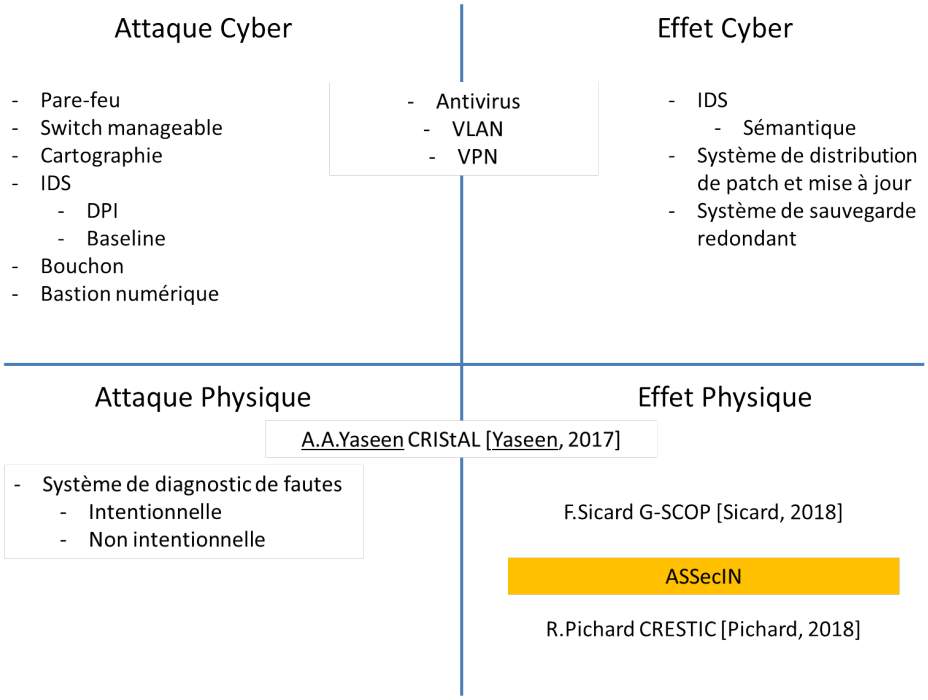
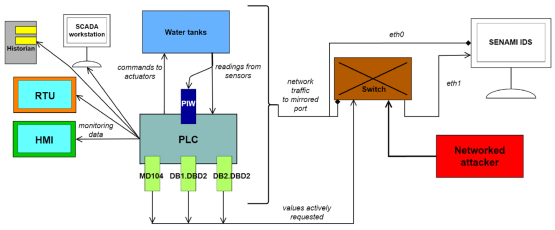


FIGURE 2.11 – Une synthèse des méthodes, outils et recherches

TABLEAU 2.1 – Tableaux comparant les différentes approches de sécurisation

Méthodes/outils/recherches	Avantage	Inconvénient
Sentryo	Compréhension du réseau	Réaction passive
Firewalling	Configurable, réaction active	Compréhension du réseau
TAIGA	Détection d'attaque certaine	Confiance en IO
signature C	Détection relative au temps	Réaction passive
Pichard	Cohérence des contraintes	Placement
Yaseen	Différence attaque défaillance	Complexité algorithmique
Sicard	Filtrage dual des informations	Concept de distance

2.5 Comparatif des principales plateformes de test



(a) Plateforme SENAMI

(b) Factory IO

(c) Plateforme TAIGA

2.5.1 Plateforme SENAMI

La plateforme SENAMI [Jardine et al., 2016] est conçue pour tester des mécanismes de détection d'un IDS pour des systèmes industriels. Elle implémente un IACS avec un réseau connectant ses entités :

— une IHM de supervision

— un automate programmable industriel, *Programmable Logic Controler* (PLC) contrôlant un réservoir d'eau

— des actionneurs et capteurs (vannes, capteurs de niveau d'eau)

— un *Remote Terminal Unit* (RTU) permettant le transfert d'information entre l'PLC et le SCADA

— un serveur d'historisation

— deux machines l'une hébergeant un SCADA l'autre l'IDS

— une switch avec un port miroir permettant la connexion de toutes les entités, dont les attaquants

2.5.2 Factory IO

Factory IO [Riera et al., 2017] est un outil de "virtual commissionning", il permet la simulation de la partie opérative d'un système industriel piloté par une partie commande. Cet outil implémente une solution de conception et de simulation :

— un designer de système par agrégation,

— un simulateur de partie opérative,

— une interface de communication avec la partie commande,

— une interface graphique pour l'utilisateur,

— un gestionnaire d'aléas,

Le "virtual commissionning" est l'activité de répliquer le comportement physique d'un composant ou d'un système dans un environnement virtuel. Cela permet le test fonctionnel du système lors de la phase de conception, mais aussi la formation des opérateurs sans risque de détérioration de l'outil de production. De nos jours les travaux s'orientent vers le jumeau numérique pour émuler le système physique et prédire son comportement.

2.5.3 Plateforme TAIGA

La plateforme TAIGA est conçue pour éprouver le système de protection décrit en section 2.2.1.1. Elle implémente un robot avec le dispositif de protection et un réseau pour le commander [Lyn et al., 2015]. Un robot est assimilé à un CPS, car il est composé d'entités communicantes et a des interfaces cyber et physique.

2.5.4 Comparatif

TABLEAU 2.2 – Synthèse sur trois plateformes de test pour des mécanismes de sécurité

plateformes	tests	orientation	architecture	scénarios d'attaque
SENAMI	détections	OT	physique	corruption, prise de contrôle
Factory IO	fonctionnement/défaillance	OT	virtuelle	corruption, prise de contrôle
TAIGA	détection/réaction	OT (robotique)	physique	tous

Les points communs des trois plateformes sont : leur volonté de tester des mécanismes de détection et leur orientation les systèmes OT ou de production industrielle.

TABLEAU 2.3 – Comparatif des trois plateformes de test pour des mécanismes de sécurité

plateformes	avantages	inconvénients
SENAMI	l'ICS est complet	attaque simple, implémentation physique, niveaux 0 et 1 du CIM
Factory IO	implémentation virtuelle, gestion des aléas	scénarios d'attaque, niveaux 0 et 1 du CIM
TAIGA	scénario d'attaque	spécifique robotique, implémentation physique, pas d'aléa

La plateforme TAIGA est selon nous la plus aboutie bien qu'elle ne prend pas en compte les aléas. Les scénarios d'attaques que l'on peut y jouer n'ont de limite que l'imagination des attaquants et la plateforme.
La plateforme SENAMI est celle qui correspond le mieux au milieu industriel. Cependant les scénarios d'attaques pouvant y être jouées sont beaucoup plus simples. En effet il n'y a pas d'autre point d'accès que celui proposé et le niveau 2 du CIM (la supervision) est manquant.
L'outil Factory IO offre un environnement virtuel pour l'observation des attaques sur un système. Celui-ci une fois interfacée avec une partie commande virtuelle ou physique devient une plateforme de test cependant comme pour SENAMI les scénarios d'attaques pouvant y être jouées sont simples.

2.5.5 Synthèse des plateformes

Les trois plateformes présentées sont outillées pour observer les effets ou détecter des attaques. Cependant SENAMI et TAIGA sont des plateformes physiques, les scénarios attaques qu'elles subissent sont conçus afin d'éviter la destruction de la plateforme. Aussi ces plateformes sont difficilement adaptables à des cas industriels variés, car elles ne sont pas modulables. Factory IO quant à lui est adaptable aux cas industriels et l'effet des attaques étant virtuel il n'y a pas de limite. Cependant des développements conséquents sont nécessaires pour orienter la plateforme sur le domaine cyber. Mais le concept de simulation des parties opérative et commande est intéressant.

Nous mettons donc ici en avant un besoin de réalisation d'une plateforme hybride orientée cyber.

2.6 Conclusion

Dans ce chapitre nous avons fait un état l'art de sur trois domaines différents :

- Les méthodes de sécurisation pour les IACS avec deux approches :
  - La comportementale ("boite blanche" connaissance du système)
  - L'informationnel ("boite noire" connaissance de mécanismes générique exemple : la communication)
- Les travaux antérieurs du laboratoire en conception de système et génération à partir de modèle
  - l'approche top/Down (définition du système par ses rôles)
  - l'approche bottom/up (définition du système par ses composants)
  - l'approchera hybride (définition du système par ses rôles et composants)
- les plateformes de test orientées sécurité pour les SAP

Nous avons pu constater que de nombreuses solutions existent, mais à bas niveau très peu peuvent être déployées. Or ce positionnement est intéressant, car il protège le monde physique face aux attaques.
Les chapitres suivants vont s'attacher à présenter notre démarche et nos contributions pour la sécurité des SAP. Notre démarche tire profit des analyses effectuées dans ce chapitre. Aussi elle s'appuiera sur les travaux antérieurs du laboratoire.

3 La passerelle ASSECIN

« Maintenant, nous sommes tous connectés par internet, comme des neurones dans un cerveau géant. »

Stephen Hawking

Sommaire

Figures

Le chapitre précédent a présenté un état de l'art sur les approches de sécurisation pour les SAP. Aussi les problématiques et besoins liés à leurs sécurisation ont été justifiés dans le contexte.
Dans nos travaux nous nous intéressons tout particulièrement aux méthodes de détection comportementale. En surveillant un IACS durant son exploitation par ses variables et en le supervisant, afin qu'il ne puise pas impacter son environnement ou détériorer la production.
Fort de ce contexte, on se propose ici de présenter notre outil de sécurisation pour l'IACS :

1. en donnant sa spécification.
2. puis en présentant les différentes fonctions du dispositif.
3. enfin nous donnerons la spécification de nos fonctions.

3.1 La spécification du dispositif de sécurité

Dans cette section, différents concepts sont abordés afin de spécifier notre outil. Nous allons présenter son rôle ainsi que ces caractéristiques. Notamment, le placement du dispositif sera présenté, ainsi que les différents niveaux temporels qu'il adresse.

3.1.1 Les objectifs de sécurisation

L'outil de sécurisation doit assurer que, pendant le fonctionnement d'un système, les effets d'une anomalie n'impacteront pas son environnement ou lui-même. Cette assurance sera prodiguée par la détection de violation de contrainte de sécurité, relative à la sécurité et l'intégrité physique du système, ainsi que par une stratégie de réponse vis-à-vis de cette violation. Il doit donc détecter et réagir de manière autonome aux effets qu'engendrent les anomalies, lorsque ceux-ci impactent la sécurité de son environnement ou l'intégrité du système.
L'outil doit aussi assurer l'identification d'une anomalie qui impacte les opérations de production. Il doit donc détecter et informer de manière autonome en observant les effets qu'engendrent les anomalies sur le comportement du système.
Les effets sont observés en surveillant les variables de la communication entre l'intelligence et le/les acteurs de processus. Cette observation ne devant pas perturber le fonctionnement du système, le dispositif ASSecIN est placé en filtre sur le réseau de terrain. C'est ce que nous nous proposons de vous présenter dans la section suivante.

3.1.2 Le concept de passerelle

Le dispositif a pour vocation de sécuriser un IACS face à des attaques cyber et/ou physique. Comme vue précédemment deux approches sont possibles. Cependant, toutes deux nécessitent de se placer dans le système. Plusieurs solutions ont été envisagées :

— L'insertion des méthodes de détection et réaction dans un/des équipements existants. C'est le cas notamment pour le projet DICOLO [Pichard et al., 2018].

— L'implémentation des méthodes dans un nouvel équipement. Cet équipement s'insérera ensuite dans le système sur le réseau de terrain. Il peut aussi nécessiter la mise en place d'un nouveau réseau.

Notre proposition consiste à s'insérer de manière non intrusive dans le système. Les problématiques de sécurisation peuvent nécessiter de prendre pour hypothèse qu'une partie du système est de confiance. Nous prenons comme hypothèse que : ni la partie commande d'un système comme le PLC ni la partie opérative ne sont de confiance. Cela implique que notre outil ne peut pas être mis en place dans l'une ou l'autre de ces entités. Cependant celui-ci détecte des anomalies en observant les variables du système et y réagi. L'outil s'insère donc dans le flot de communication en filtre entre ces deux entités, la partie commande et les I/Os (partie opérative).
Il est mis en œuvre sous la forme d'un équipement permettant d'interconnecter des réseaux, d'où le concept de passerelle, entre le réseau de terrain, l'interface physique et le réseau de la partie

commande, l'interface cyber. La passerelle se charge de faire circuler l'information d'une interface à l'autre tout en implémentant des méthodes de détection et réaction. Les informations seront donc vérifiées dans un sens comme dans l'autre de la communication. Le placement de la passerelle est illustré dans la figure 3.1.

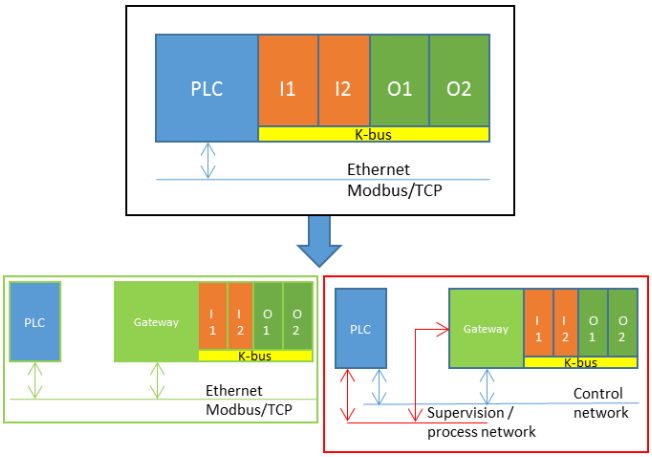


FIGURE 3.1 – Placement physique de la passerelle

En raison de son placement stratégique, la passerelle est :

— Le premier observateur d'anomalies typées physique, car elle reçoit en premier les informations remontant de la PO.

— Le dernier rempart contre des anomalies typées cyber, car elle transmet en dernier les ordres vers la PO.

Ce placement n'est pas dépendant de la topologie réseau (bus, ligne, étoile, mesh) physique ou logique déployée.

Les Réseaux Locaux Industriels, *industrial local area networks* (RLI) sont denses et étendus géographiquement et logiquement et ont plusieurs topologies [Penney and Baghdadi, 1979]. De grands volumes d'information transitent par ces réseaux et de multiples protocoles interviennent dans leur mécanique de communication. Les RLI partent du point d'accès de l'entreprise vis-à-vis d'internet et s'étendent à travers les différents sites par des intranets comme montrés en figure 3.2. Puis ils permettent la communication entre les équipements et dans l'équipement. On distinguera ici deux niveaux :

— le niveau IT : qui est l'ensemble des réseaux partant du point d'accès d'une entreprise jusqu'aux entités du processus. Ils sont implémentés physiquement avec des bus hiérarchisés par des switches et routeurs, ce qui privilégie les communications d'entité à entité.

— Le niveau OT : qui est l'ensemble des réseaux entre les entités du processus. Ils sont implémentés physiquement avec des bus / ligne / anneau, ce qui privilégie les communications en diffusion d'entité à ensemble d'entités.

Nous présentons les topologies qui nous intéressent : le bus, la ligne, l'étoile et le "mesh", elles sont les plus représentatives des réseaux OT. Nous illustrons le placement de la passerelle selon deux positions :
La première au plus proche de l'intelligence de processus en figure 3.3. Ce placement favorise la sécurité inter équipement, mais nécessite un modèle de sécurité interne conséquent (toutes les E/S de tous les équipements)
La deuxième au plus proche des blocs d'E/S ou des équipements en figure 3.4. Ce placement favorise la sécurité des équipements et diminue/focalise la latence sur un équipement, mais l'influence du pilotage des équipements les uns par rapport aux autres n'est pas prise en compte.

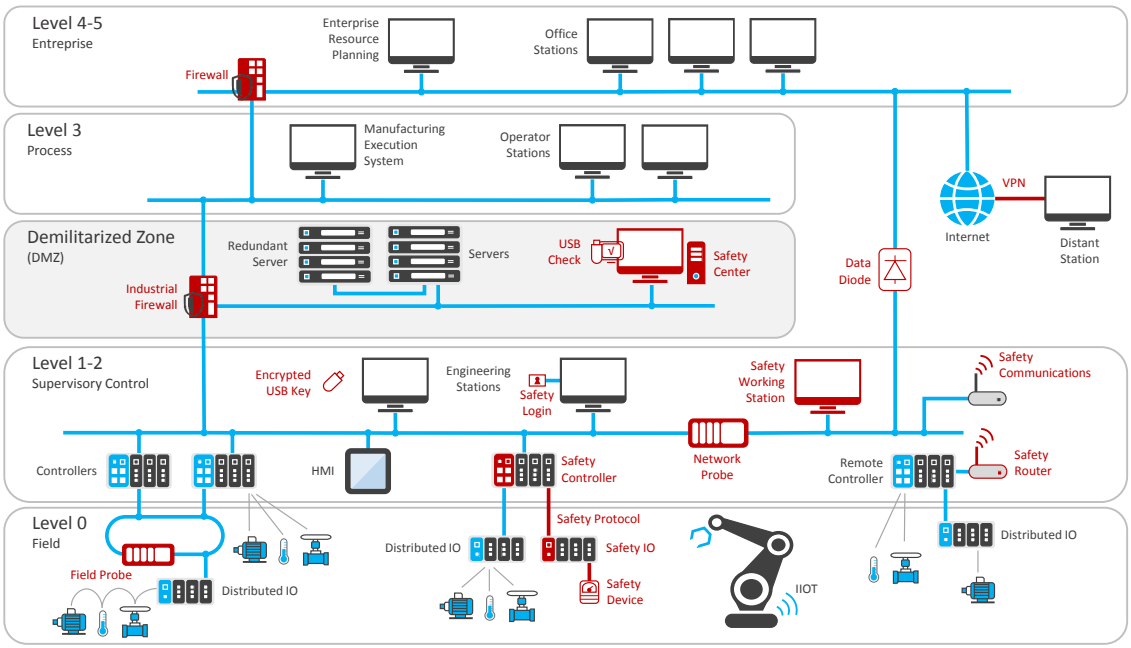


FIGURE 3.2 – Point de vue sur le réseau d'un SAP

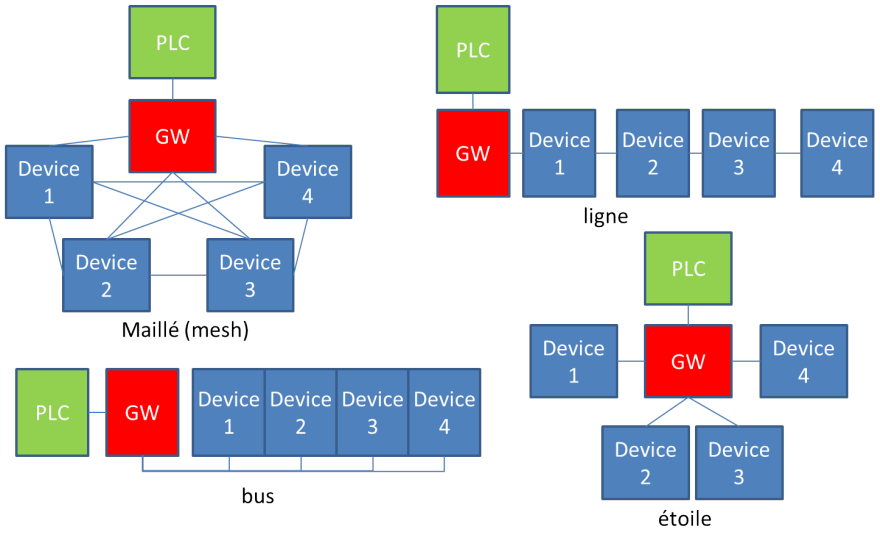


FIGURE 3.3 – Positionnement de la passerelle proche du PLC pour différentes topologies

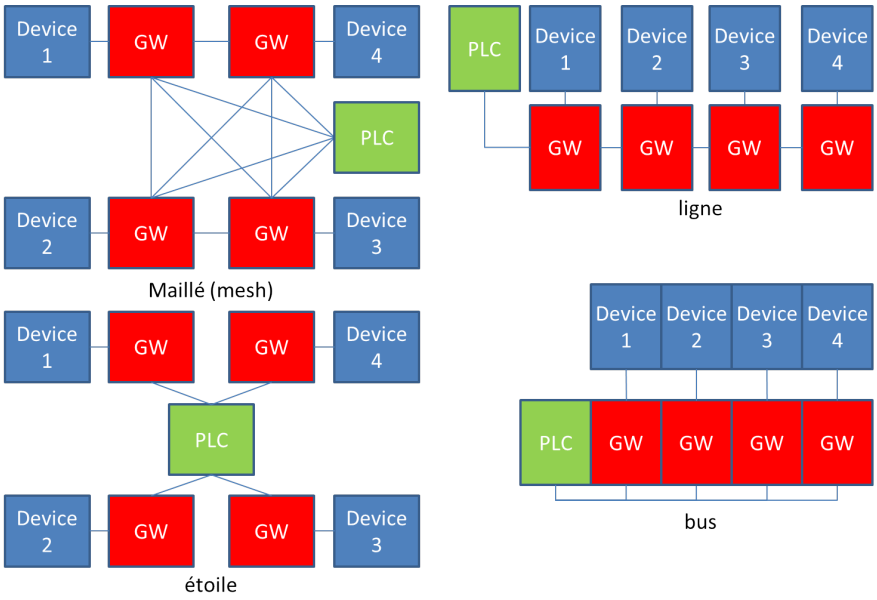


FIGURE 3.4 – Positionnement de la passerelle proche des équipements pour différentes topologies

Le placement de la passerelle est donc flexible, cela permet d'optimiser le déploiement de la solution par rapport à l'architecture du réseau et du système, mais aussi à la charge, au protocole de communication employé ou à l'influence du pilotage des équipements. En effet s'il n'y a pas ou peu d'influence dans le pilotage des équipements par rapport aux autres, la meilleure position est proche des équipements, car elle allège le modèle de sécurité dans chaque passerelle.
Sinon c'est la proximité avec le PLC qui est préférable, car elle tient compte du pilotage de tous les équipements les uns par rapport aux autres.
Ce choix de positionnement est laissé au concepteur, car il est indépendant du fonctionnement de la passerelle seul le modèle de sécurité à configurer change. Lorsque plusieurs passerelles sont présentes pour sécuriser un système, une communication entre celles-ci est envisagée en perspective 6.2

Les informations du réseau de terrain sont typées et des mécanismes permettent de les traduire, comme montrés dans la figure 3.5. Il y a 5 types d'informations qui circulent dans les équipements sur le réseau de terrain :

- Les informations binaires, ou Tout Ou Rien (TOR)
- Les informations analogiques,
- Les informations numériques,
- Les trames,
- Les paquets,

Ces 5 types d'informations peuvent ensuite être transformés à travers des mécanismes orientés :

- Le seuillage, transformation d'un signal d'entrée analogique en signal de sortie TOR
- Les transistors, transformation d'un signal d'entrée Tout Ou Rien en signal de sortie analogique. Exemple : les ponts en H qui transforme une modulation de largeur d'impulsions, *Pulse Width Modulation* (PWM) en grandeur de tension dans un circuit.
- Les convertisseurs,
- le multiplexage et le démultiplexage,
- l'encapsulation et la désencapsulation,
- La paquetisation et la dépaquetisation,

Un composant consomme, fournit ou transmet de l'information. Avec la révolution du numérique et l'avènement de l'internet des objets industriels, *Industrial Internet of Thing* (IIoT), les composants présents sur le terrain sont de plus en plus intelligents. En effet, elle transforme directement l'information en la fournissant ou en la consommant.
Pour conclure sur les informations du réseau, on trouve de nos jours les informations basiques (binaire, analogique ou numérique) sur les bus internes aux composants. Les réseaux inter(composant/équipement) mettent en œuvre des protocoles de communication, qui encapsulent/empaquettent l'information sous forme de trames/paquets. Ce sont ces informations qui maintenant circulent sur les réseaux de terrain de I4.0.

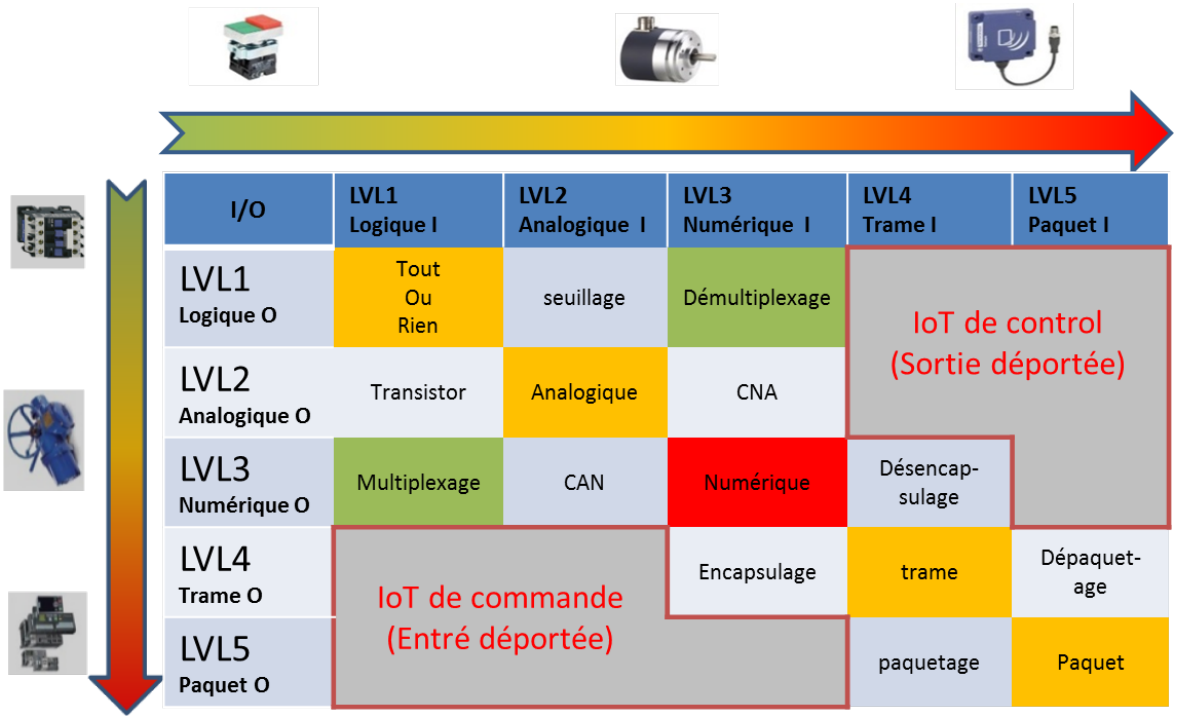


FIGURE 3.5 – Typologie des informations et mécanisme

Le déterminisme du réseau de terrain induit des contraintes temps réel. Or notre passerelle s'y place pour capter les différentes variables représentant l'état du système.

3.1.3 Les différents niveaux temporels

Nous avons formulé une nouvelle hypothèse :"Le dispositif ne doit pas perturber le système". La nécessité de concevoir une passerelle avec des niveaux temporels a donc émergé. Or le système a des impératifs au niveau temporel. En effet la communication sur les RLI en particulier à bas niveaux est déterministe.

Le déterminisme des réseaux industriels Le déterminisme implique l'assurance que les informations seront envoyées aux destinataires en un temps déterminé. De cette façon, on s'assure que l'occurrence d'un événement sera prise en compte, mais aussi que la prise en compte se fera dans le bon ordre. Même si plusieurs composants mettent à jour leurs informations durant un même cycle automate, on veut que celles-ci soient prises en compte dans l'ordre de leur occurrence.
De plus la communication à ce niveau est dans la majorité des cas cyclique. Elle a un cycle de quelque ms, qui correspond à la durée d'un cycle automate. Or certaines méthodes de détection réclament des analyses pouvant prendre du temps (plusieurs s) ou en dépendre. Nous avons donc fait le choix d'implémenter des méthodes de détection et de réaction dans une passerelle avec différents niveaux temporels, décrits dans la figure 3.6 :

— Le niveau temporel 1 (N1) correspond à la chaine de communication décrite en section 3.3.1. Il traite l'information du réseau en respectant les contraintes temps réel. C'est aussi à ce niveau que la chaine de détection et réaction "réflexe" 3.3.2 est mise en place. Elle implémente des méthodes décrites précédemment en section 3.2.1.1,3.2.2.2.

— Le niveau temporel 2 (N2) correspond à l'analyse des informations ainsi que du trafic. À ce niveau, les contraintes ne sont plus "temps réel", mais sont plutôt opérationnelles. Les méthodes de détection sont donc relatives au temps ou à l'historique des valeurs. Elles sont décrites dans les paragraphes 3.2.1.1.

— Le niveau temporel 3 (N3) correspond à la décision avec un point de vue local (l'équipement protégé). Il gère la remontée d'information au haut niveau et l'alerte ainsi que des réactions actives plus complexes.

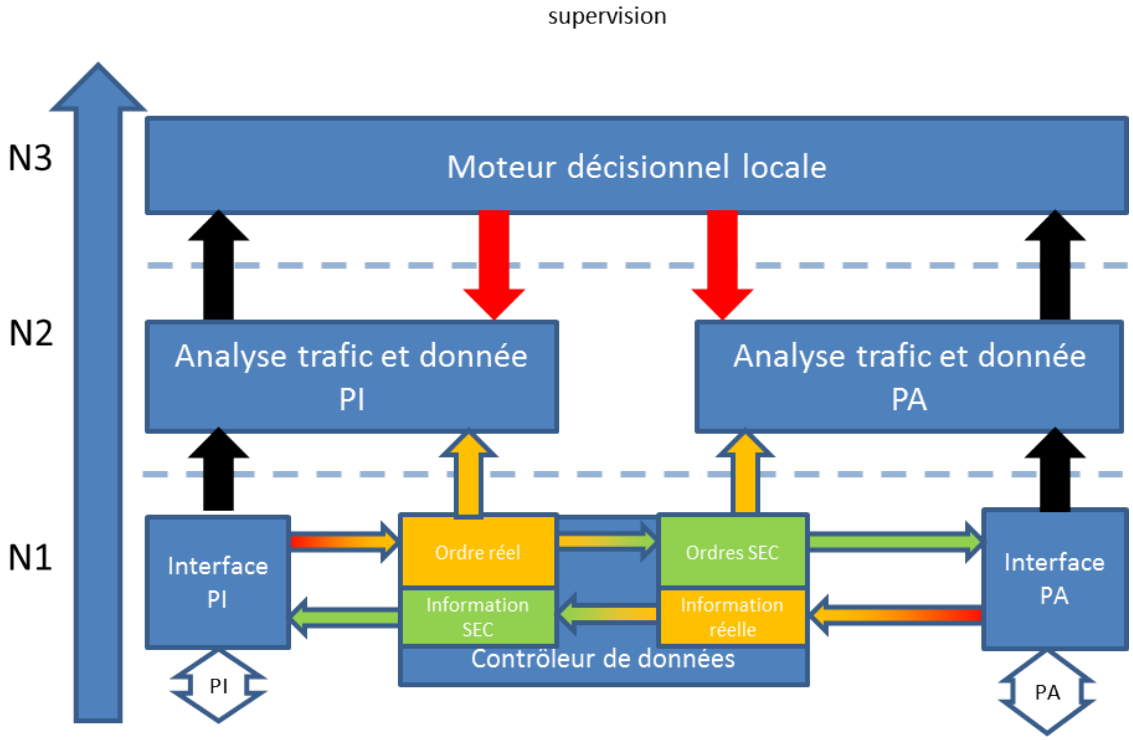


FIGURE 3.6 – Concept de la passerelle

Nos travaux se sont donc intéressés au concept de passerelle (nœud réseau) pour le réseau de terrain, car il permet de s'insérer dans le flot de communication en coupure et de manière invisible. Aussi la passerelle observe les ordres en dernier avant l'actionneur et les informations en premier avant l'automate. Ce placement permettra la mise en place de réactions plus sécurisantes vis-à-vis de l'environnement physique, ainsi que de meilleures détections, car les variables observées sont plus significatives point de vue du comportement du système (elles font sens).

3.2 Le fonctionnement de la passerelle ASSecIN

La passerelle sécurise le fonctionnement d'un système dont elle a la charge. L'objectif est d'observer en perturbant seulement de manière minimale le système afin de détecter des anomalies. Cette phase met en œuvre plusieurs méthodes de détection. Elle s'établit sur l'observation d'un comportement contraire aux exigences de sécurité. Après cette phase, une phase de remédiation intervient. La remédiation met en œuvre des méthodes de réaction. La réaction s'établit sur les interfaces de la passerelle, afin d'éviter que les effets de l'anomalie influent sur le système et d'informer de l'occurrence d'une anomalie,

3.2.1 Le concept de détection

La perception, est l'activité "d'extraire de donnée collectée par acquisition des indicateurs, qui seront utilisés pour déterminer si les évolutions constatées correspondent à un fonctionnement normal ou non" [Niel and Craye, 2002]. La détection, "caractérise le fonctionnement du système de normal ou d'anormale" [Niel and Craye, 2002] selon les indicateurs perçus.
Dans cette section, trois détections sont reprises et détaillées avec la perception qu'a la passerelle. En effet en raison de son placement décrit précédemment, elle perçoit les informations du système. Cela fournit différents indicateurs, qui renseignent différentes détections possibles. Deux différentes approches pour la sécurisation des IACS sont donc possibles.

3.2.1.1 L'approche comportementale

Nous avons entrepris d'utiliser l'approche comportementale qui se base sur l'observation du comportement réel. Une comparaison de celui-ci avec une référence permet la détection d'anomalie. Le choix de la référence déterminera la méthode de détection. Deux méthodes sont ainsi présentées :

La comparaison par contraintes logiques décrite dans la section 2.3 faite appel à un référentiel figé. C'est un ensemble de règles permettant de vérifier si le comportement d'un système reste dans son état nominal. À la mise à jour d'une variable, on observe l'état des autres variables afin de déterminer si elle est normale. Les variables sont de deux types les ordres envoyés par l'intelligence du processus et les informations de l'environnement remontées par les acteurs de celui-ci. Il y a donc quatre ensembles de règles :

- par rapport aux ordres sur les autres ordres,
- par rapport aux ordres sur les informations,
- par rapport aux informations sur les ordres,
- par rapport aux informations sur les autres informations.

On peut ainsi déterminer si l'anomalie est issue de l'intelligence ou des acteurs du processus. En effet, nous prenons comme hypothèse qu'aucune des entités n'est de confiance.

La comparaison à signatures temporelles décrite dans la section 2.2.1.2. Elle fait appel à un référentiel figé dans le temps. C'est une représentation du principe d'action réaction appliquée à un système. C'est un ensemble de signatures qui décrit des séquences d'événements et leur temporalité par rapport à un événement maître. En effet, le but de tout système de production est de transformer la matière d'œuvre. Or pour le faire, il utilise des acteurs du processus, les actionneurs qu'il contrôle par des ordres. Ces actionneurs influent sur l'environnement du système, qui est mesuré par les capteurs. Ils remontent sous forme d'information l'influence sur l'environnement c'est ce que l'on appelle la "boucle de feedback". Ces informations sont induites par le comportement du système. Donc elles sont causées par les actionneurs et leur occurrence à une temporalité plus ou moins fixe dans le temps.
Ces deux méthodes peuvent être cumulées, car leurs références n'ont pas la même échelle temporelle. La première est réflexe, son horizon temporel est T=0. La seconde détecte dans le temps donc son horizon temporel est T=$(1 \rightarrow \infty)$

3.2.1.2 L'approche informationnelle

Cette approche majoritairement répandue à haut niveau dans le SAP est aussi intéressante à bas niveaux. En effet, c'est à ce niveau que les informations issues du monde physique sont les plus "de confiance". Car aucune autre entité n'a pu les modifier ou les abstraire. Les informations sont des variables mises à jour par le système pour l'intelligence du processus. Elles rendent compte

en partie de l'état du système et doivent être prises en compte dans le bon ordre. Les informations peuvent être de différents types et ce sont eux qui définiront le paramètre à surveiller. La surveillance des informations peut se faire de plusieurs manières nous en présentons une :

L'historisation de variable permet de garder une trace des variables. Ainsi plusieurs paramètres peuvent être extraits selon le type de variable :

- Booléenne, ces variables n'ont que deux états, mais leur fréquence d'update est souvent la même ou dépendante de la production.
- Numérique, ces variables ont plusieurs états correspondants à la grandeur physique qu'elle traduise, cependant l'hystérésis entre deux valeurs peut être seuillé et la fréquence d'update est elle aussi souvent la même ou dépendante de la production.
- Message, ce sont des ensembles de caractère, on peut placer les communications réseaux dans ce type. Pour ce type, on cherchera plus à vérifier les messages précédents et leur cohérence.

Ces paramètres fournissent de l'information supplémentaire déduite des informations physiques historisées.
Les détections sont indispensables aux réactions, car ce sont elles qui vont les déclencher. En effet c'est en détectant puis observant les effets que la passerelle identifie l'anomalie, c'est cette identification qui va permettre la mise en œuvre de réactions appropriées.

3.2.2 Les concepts de réaction

Dans cette section, nous allons détailler les concepts pour quatre réactions avec le point de vue qu'a la passerelle. Deux types sont présentés :

- Les réactions actives influent sur les variables et donc sur le comportement du système.
- Les réactions passives n'influent pas sur le comportement, mais génèrent de l'information pour d'autres entités du SAP.

Elles sont des stratégies permettant de remédier des anomalies détectées, cette remédiation peut intervenir sous différentes formes :

- assurer les exigences de sécurité.
- fournir de l'information.

3.2.2.1 Les réactions passives

Ce sous-ensemble de réaction intervient de manière cyclique ou événementielle. Il réagit en remontant les informations pour différents "outils". C'est le choix de la destination qui détermine la méthode de réaction.

L'alerte est décrite dans le chapitre 2. Cette méthode permet d'inclure l'humain dans la boucle, Cependant l'humain ayant des limitations dans l'analyse de l'information. Une synthèse des résultats de détection est formulée pour faciliter son diagnostic et/ou sa prise de décision. L'alerte intervient à l'occurrence d'une anomalie en mettant en forme l'information pour atteindre l'humain à travers les IHM.

La remontée d'information permet d'informer les entités cyber de haut niveau, elle se déclenche de manière cyclique et/ou avec l'occurrence d'une anomalie.
Le système étant hiérarchique ces réactions permettent surtout d'informer le haut niveau du système exemple un SIEM. Cette entité participe ensuite à la remédiation de l'anomalie en corrélant les informations pour permettre la prise de décision. Le point de vue plus global sur tout le site de production permet de décider quelle réaction déclencher et où dans le système. Tandis que d'autres réactions locales cette fois-ci sont directement mises en œuvre dans la passerelle.

3.2.2.2 Les réactions actives

Ce sous-ensemble de réaction intervient après des détections lors de la surveillance du système. Il réagit en le supervisant avec des méthodes commandées par un moteur de réaction. Les méthodes déterminent comment superviser le système qui subit une anomalie.

Le filtrage de commande est une méthode permettant de filtrer les ordres envoyés à la partie opérative. Il intervient après une détection et selon une référence. La référence fournit un modèle de supervision du système exemple automate à état, qui en fonction des résultats des détections déterminera si l'ordre doit être filtré. L'objectif est de garantir les exigences de sécurité tout en permettant un fonctionnement minimal, mais aussi d'interdire des anomalies typées physiques d'interagir avec le monde physique. En effet, celles-ci peuvent causer des détériorations sur le système, le produit ou les personnes.

La mise en repli est une méthode permettant de modifier tous les ordres envoyés aux équipements, pour forcer la totalité du système dans un état voulu sans tenir compte de l'intelligence de processus [Toguyéni et al., 2003]. Ainsi on verrouille les interactions physiques de l'IACS sans pour autant modifier les interactions cyber des équipements.

Les concepts de notre dispositif ayant été introduits, nous allons maintenant présenter les fonctions mettant en œuvre nos méthodes et concepts.

3.3 La description fonctionnelle de la passerelle

La passerelle ASSecIN implémente plusieurs méthodes de sécurisation pour les IACS. Nous avons choisi de les mettre en œuvre dans différentes chaines d'information. En effet, les réactions sont pour la plupart commandées par la détection, qui dépend elle-même des informations capturées sur le réseau. Ces chaines sont dépendantes des niveaux temporels. Elles représentent les chemins de l'information dans la passerelle et sont présentées avec les interfaces dans la figure 3.7.

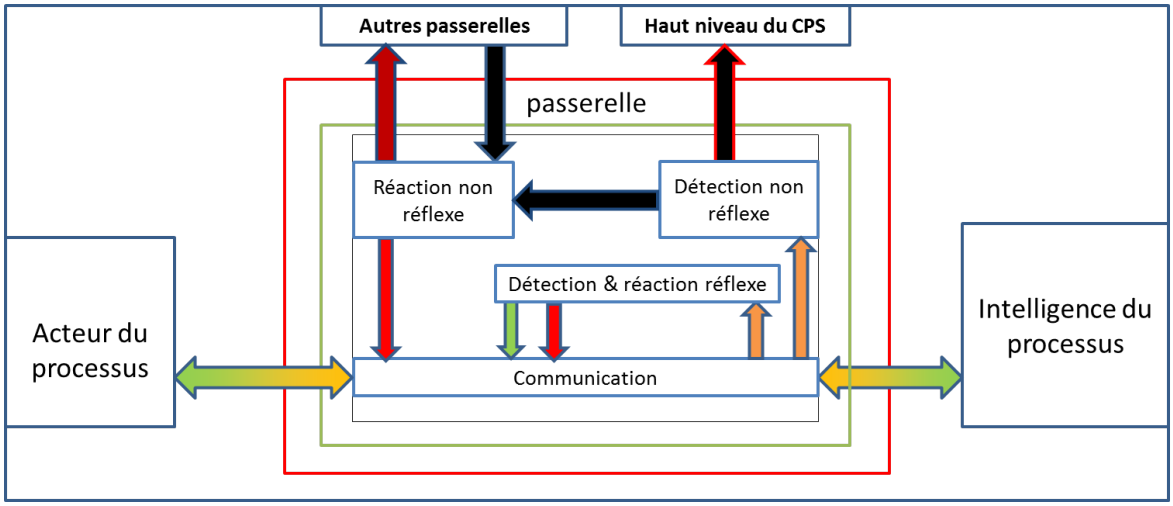


FIGURE 3.7 – Les chaines d'information de la passerelle

Quatre chaines d'information ont ainsi été organisées :

— La chaine de communication qui gère les informations transitant par les interfaces de la passerelle.

— La chaine de détection réaction "réflexe" qui garantit les exigences de SdF du système

— Les deux chaines de détection réaction "asynchrone" qui garantissent les exigences opérationnelles du système, ainsi que l'intégrité des informations de celui-ci.

Elles regroupent les différentes fonctions traitant l'information du réseau premièrement, mais aussi des informations transformées ou générées par la passerelle en elle même.

3.3.1 La chaine de communication

La chaine traite les informations ou requêtes des différentes interfaces du système. Les informations proviennent du réseau qu'elle observe. Les requêtes viennent de son interface pour le haut niveau du SAP ou des autres passerelles.
Quatre fonctions entrent en jeux dans cette chaine :

- L'updater qui met à jour les informations du système dans le modèle interne de la passerelle,
- Le writer qui réécrit les informations du système en fonction de celle dans le modèle interne,
- Le filtre qui sélectionne les communications à prendre en compte par la passerelle,
- Le contrôleur de données qui gère le modèle d'information interne à la passerelle, il permet de remonter les informations vers le haut niveau du SAP et de déclencher les réécritures ou l'envoi des communications.

Ces fonctions sont illustrées avec leurs flots d'informations en figure 3.8.

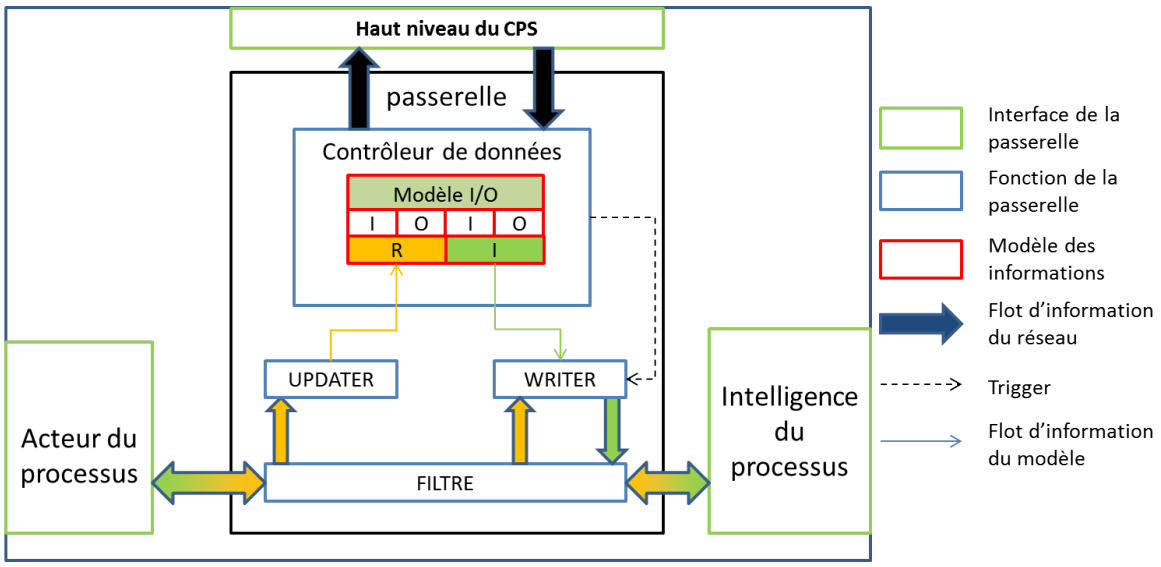


FIGURE 3.8 – Décomposition chaine de communication

Un diagramme de séquence est donné en figure 3.9, qui montre les flots d'information entre les fonctions de cette chaine. Le diagramme montre neuf séquences :

- Les communications non filtrées sont représentées par deux séquences.
- Les communications filtrées de la partie commande (serveur) vers la partie opérative (client) sont représentées par :
  - une séquence sans réaction déclenchée.
  - une séquence avec réaction déclenchée.

  On constate ici l'incidence du parcours de la chaine réflexe sur la latence induite par la passerelle, qui est augmentée par la mise en œuvre de réaction active.
- Les communications filtrées de la partie opérative (client) vers la partie commande (serveur) sont représentées par :
  - une séquence sans réaction déclenchée.
  - une séquence avec réaction déclenchée.

  On peut observer que le parcours des chaines d'information non réflexe n'a pas d'incidence sur la latence induite par la passerelle.

— Les communications avec le haut niveau sont représentées par trois séquences.
  — une séquence avec requête du haut niveau.
  — une séquence avec un déclencheur temporelle.
  — une séquence avec un déclencheur par la réaction.

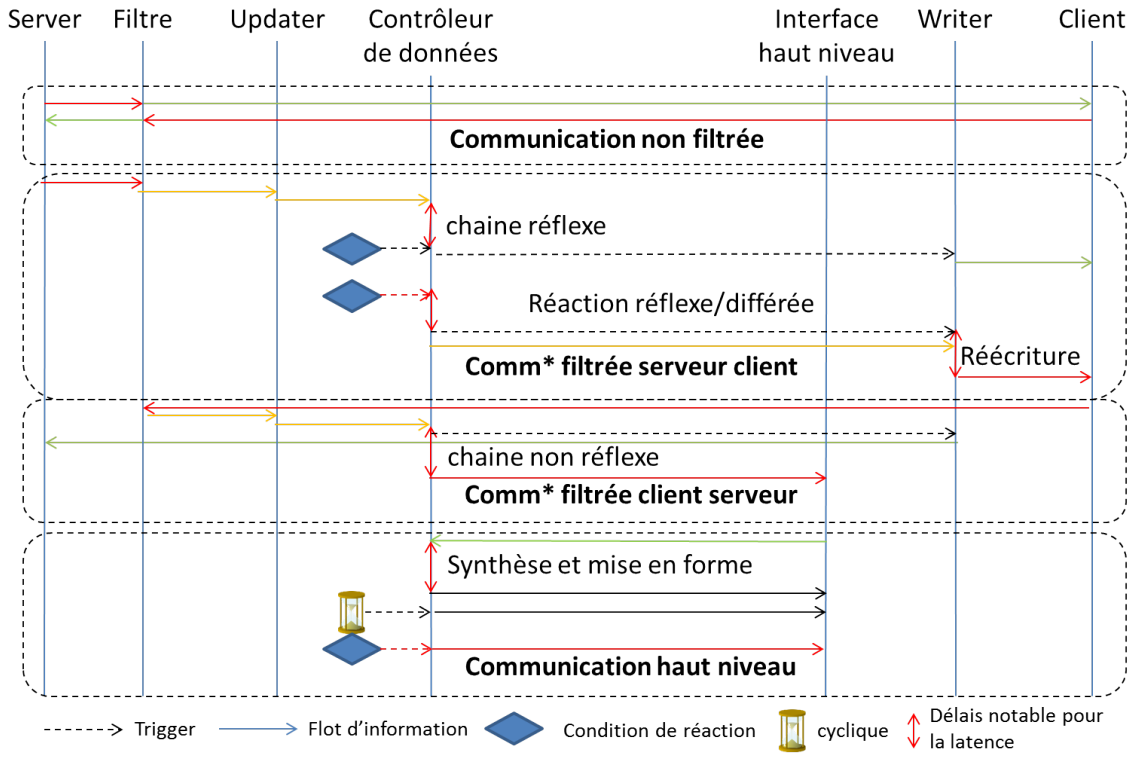


FIGURE 3.9 – Séquences de la chaine de communication

Ce sont les fonctions de réaction des chaines d'information qui commandent le contrôleur de données, celles-ci selon leurs résultats commandent l'envoi de la communication, sa réécriture ou l'envoi d'information au haut niveau.

3.3.1.1 L'observation des réseaux

La passerelle est un équipement du réseau placé en coupure sur celui-ci. Elle capture donc toutes les informations transitant sur le média de communication ce qui en fait un outil d'observation. Plusieurs métriques en plus des informations peuvent ainsi être observées. Elles rendent compte de l'état du réseau comme :

— la charge du réseau (nombre de communication / milliseconde),
— la taille des communications (nombre d'octets d'une trame ou d'un paquet),
— la perte de communication,
— les protocoles utilisés,
— la latence,

La fonction qui exploite ces métriques est le filtre. Il sélectionne par un algorithme les communications à prendre en compte. Nous avons étudié des protocoles applicatifs et observé les communications avec l'outil Wireshark. Cela nous a permis de trouver les métriques et paramètres à prendre en compte pour l'algorithme de filtrage. On a déterminé que l'identification des communications est la fonction principale de l'observation. Notre algorithme prend en compte la taille

des communications ainsi qu'un index de hachages. Le hachage des communications suit une équation :

$$Hachage_{Index} = \sum_{i=1}^{n} \begin{cases} Variable^{i}_{adresse} + 2^{i} & \text{if} \quad Variable^{i}_{valeur} = 1 \\ 0 & \text{if} \quad Variable^{i}_{valeur} = 0 \end{cases} \tag{3.1}$$

Cette équation doit être adaptée au protocole se trouvant sur le réseau observé. Cependant, elle ne se limite pas à un nombre ou un type de variable et est mise en œuvre dans l'algorithme 1 à chaque nouvelle communication.

Algorithm 1 Algorithme de filtrage des communications

Require: x est une nouvelle trame capturée de taille y, x' est un booléen informant d'une capture, u est la liste des index de hachage des précédentes trames par taille, t est la taille des données dans la trame/paquet, l est la liste des différentes tailles de trame à analyser

```
1:  function COMPARAISON(x, u)
2:      δ ← x ⊕ y                                   ▷ ⊕ : 2nd niveau de filtrage 3.1
3:      if δ ≠ u then
4:          δ ← u
5:          return TRUE
6:      else
7:          return FALSE
8:      end if
9:  end function
10: while x' = TRUE do
11:     if y ≠ l then
12:         laisse passer la communication
13:     else
14:         σ ← COMPARAISON(x, u)
15:         if σ = TRUE then
16:             transmet la communication à l'Updater
17:         else
18:             laisse passer la communication
19:         end if
20:     end if
21: end while
```

Cette technique assure un filtrage des communications par la passerelle pour des protocoles cycliques ou événementiels. Aussi elle diminue fortement le volume d'information à traiter pour les chaines de détection et réaction. Nous apportons donc en partie une solution à l'un de nos verrous "La passerelle ne doit pas perturber le réseau".

3.3.1.2 Le modèle d'information

Nous avons pris comme hypothèse que la passerelle connaît le système qu'elle sécurise. Cette connaissance est dynamique, car à chaque instant elle a la représentation du système par ses variables. C'est le modèle des informations interne à la passerelle qui apporte cette connaissance. Aussi nous avons décidé qu'elle aurait un double point de vue. Pour accomplir cela, nous avons mis en place une séparation de l'état des variables :

— La partie réelle de l'état de la variable, elle correspond à l'état de la variable vue sur le réseau.

— La partie imaginaire de l'état de la variable, elle correspond à l'état de la variable une fois celui-ci validé par la chaine assurant les exigences de la SdF. Cela nous permet de caractériser cette partie "de confiance".

Ce modèle d'information est géré par une fonction :

Le contrôleur de données a pour rôle de servir d'interface entre les chaines d'information et le modèle. Dans la chaine de communication, il s'interface avec :

— Le writer, le contrôleur de donnée déclenche la fonction du writter.

— le haut niveau du SAP, cette fonction implémente les méthodes de réaction passive, pour la remontée d'information sur demande. Elle reçoit des requêtes de lecture du haut niveau ou des autres passerelles et y répond. Elle le fait aussi de manière cyclique en prenant en compte les mises à jour entre deux émissions de rapport.

3.3.1.3 L'interprétation et la modification des communications

Le dispositif ASSecIN doit sécuriser des systèmes par leurs réseaux. Il nécessite donc l'interprétation des communications qui y transitent. Certaines méthodes de réaction nécessitent de pouvoir modifier les communications. Deux fonctions jouent ce rôle :

L'updater interprète les communications que le filtre sélectionne. Il a connaissance de l'architecture des communications, ce qui lui permet d'isoler les informations qui y sont contenues. Puis il a la charge de mettre à jour la partie réelle du modèle d'information avec ce qu'il a isolé. Cette mise à jour est directe et autonome.

Le writer reforge les communications sélectionnées par le filtre. Son rôle est d'assurer que les informations émises par la passerelle soient relatives à la partie "de confiance" du modèle d'information. Cette réécriture des communications est directe, mais elle est déclenchée par le contrôleur de donnée.
La chaine de communication a donc pour rôle de faire transiter les communications à travers la passerelle. Elle sélectionne celles qui doivent être sécurisées et met à jour la partie réelle du modèle d'information. Cette chaine a donc une interface vers le haut niveau du SAP ou les autres passerelles afin de les informer.

3.3.2 La chaine de détection réaction safety

La passerelle doit garantir les exigences de SdF. Dans cette section, on s'intéresse plus particulièrement à la "Safety", qui est la condition assurant qu'un système ne pourra ni dégrader son environnement, lui-même, ou le produit, ni blesser des personnes. Cette garantie doit être assurée en tout temps et à chaque instant. La passerelle n'observant que les informations du réseau en devant le perturber le moins possible, nous en avons déduit les spécificités et la mise en œuvre d'une chaine détaillée en figure 3.10.
Un diagramme de séquence est donné en figure 3.11, qui montre l'agencement des flots d'information entre les fonctions de cette chaine. Deux séquences y sont présentées :

— La première séquence décrit une mise à jour de variable sans détection : La fonction de détection est déclenchée par le contrôleur de données, elle analyse les variables selon le modèle de surveillance. Quand l'analyse est terminée elle valide le passage de la variable de réelle à sécurisé et déclenche la réaction, qui déclenche le contrôleur de données.

— La deuxième séquence avec détection : Le commencement de cette section est le même que sans la détection. Cependant quand la fonction de détection détecte une anomalie, elle ne valide pas le passage de la variable et déclenche la réaction. Celle-ci analyse le rapport de détection et décide d'une réaction qu'elle impose par l'intermédiaire du contrôleur de données.

Cette chaine d'information assure les exigences safety et traite les informations du modèle interne à la passerelle. Deux fonctions réalisent le travail de cette chaine :

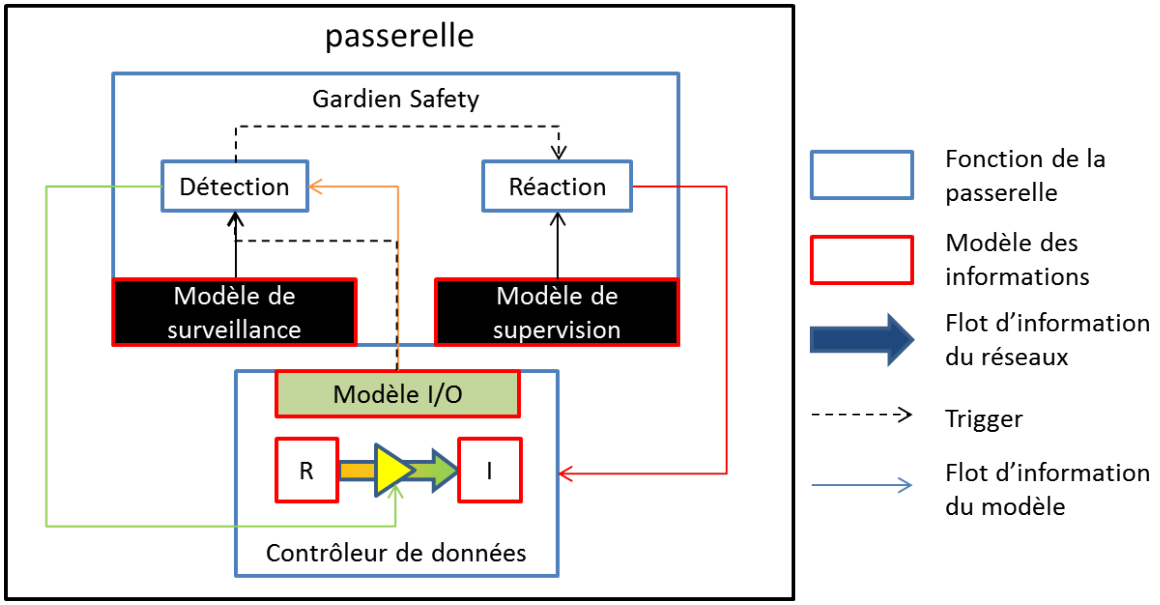


FIGURE 3.10 – Décomposition fonctionnelle pour garantir les exigences de SdF

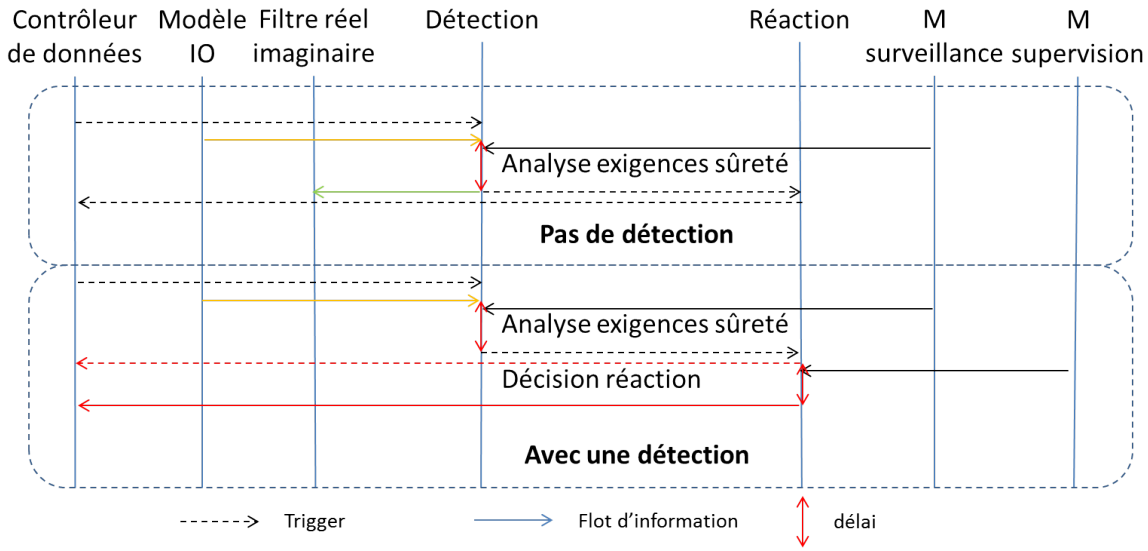


FIGURE 3.11 – Séquences de la chaine pour les exigences de safety

3.3.2.1 Le gardien de sécurité

Il implémente les méthodes de détection et de réaction et est illustré en figure 3.10. Pour cette chaine, la détection consiste en une comparaison avec référence décrite dans la section 3.2.1.1. Cette détection est binaire, elle viendra déclencher une réaction si détection il y a. La détection se fait en comparant les parties réelles et imaginaires des variables du modèle interne à la passerelle, avec un ensemble de contraintes. Celui-ci est dans un modèle de surveillance et sert de référence. La réaction a pour rôle de corriger la conséquence d'une anomalie afin de garantir le comportement normal du système. Elle est déclenchée par une détection. C'est un moteur d'exécution qui met en œuvre l'automate à état fini spécifique à la variable qui pose problème. Cet automate est dans un modèle de supervision interne à la passerelle. Le moteur d'exécution mettra en œuvre une remédiation en déclenchant le contrôleur de donnée

Le modèle de surveillance contient un ensemble de contraintes constitué de différents ensembles d'équations booléennes liées aux changements d'état d'une variable, qui selon l'état d'autres variables du système le valideront ou non. Ces équations sont sous forme d'arbre binaire (BT) illustrer en figure 3.12. La détection est mise en œuvre par des parcours d'arbre qui vérifieront les parties réelles et imaginaire des différentes variables. En effet l'analyse se faisant variable par variable il est possible que des variables aient été mises à jour dans le modèle, mais pas encore validées.

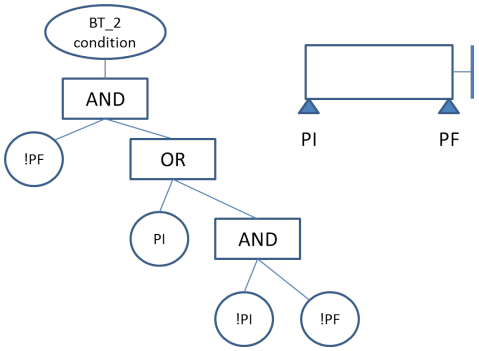


FIGURE 3.12 – Exemple de modèle d'arbre binaire de surveillance

Le modèle de supervision contient un ensemble d'automates à état fini. Chacun d'eux décrit les stratégies de remédiation possibles en fonction de la synthèse des validations. Un exemple est donné figure 3.13. En effet le modèle de surveillance ayant plusieurs équations pour vérifier le changement d'état d'une variable. La détection en fait une synthèse qui est mise en œuvre dans l'automate à état. Chaque résultat de vérification de la synthèse est pris en compte au niveau des transitions de l'automate. Les différents états de celui-ci correspondent à une solution de contrôle pour le contrôleur de données ou d'alertes.

3.3.2.2 Le contrôleur de données

Dans cette chaine d'information, il joue le rôle de filtre commandable et est illustré en figure 3.10. Il autorise ou non la mise à jour d'une variable dans le modèle interne de la passerelle. Celle-ci s'effectue en faisant passer la valeur de la variable de réel à imaginaire "sécurisé". Le filtre est commandé par la réaction, mais aussi par la détection. La différence est que la détection ne contrôle que le passage de la variable analysée. Alors que la réaction va aussi contrôler le passage d'autres variables, selon le modèle de supervision. Cette chaine est mise en œuvre dans le flot de communication. Ce flot étant déterministe nous devons le respecter et y instaurer le minimum possible de latence. La comparaison et le moteur d'exécution avec automate à état sont des solutions tout indiquées, car on peut connaitre ou paramétrer la latence qu'ils vont induire.

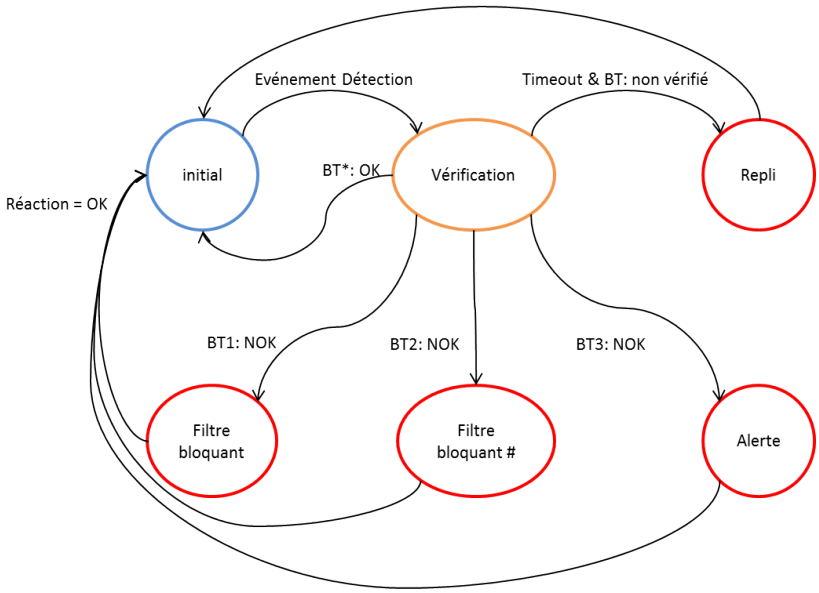


FIGURE 3.13 – Exemple de modèle automate de supervision

Les mécanismes d'analyse et décision doivent être rapides et fiables (déterminants). Cependant, d'autres mécanismes peuvent être mis en place hors du référentiel pseudo temps réel du réseau.

3.3.3 La chaine de détection réaction opérationnelle

Cette chaine d'information garantit une autre exigence de la SdF "la disponibilité", qui est la condition assurant qu'un système réalise ses fonctions dans des conditions données, à un instant donné. En effet dans de précédents travaux la nécessité d'avoir des systèmes reconfigurables a été abordée. Ces systèmes ont la capacité de se reconfigurer en reroutant des produits et en allouant de nouvelles ressources aux fonctions de gamme. Cependant l'activation de ses mécanismes bien que simplifiée nécessite in fine l'homme qui déterminera ou validera l'équipement à neutraliser. Cette chaine est mise en œuvre pour détecter les défaillances. Elle est présentée en figure 3.14.

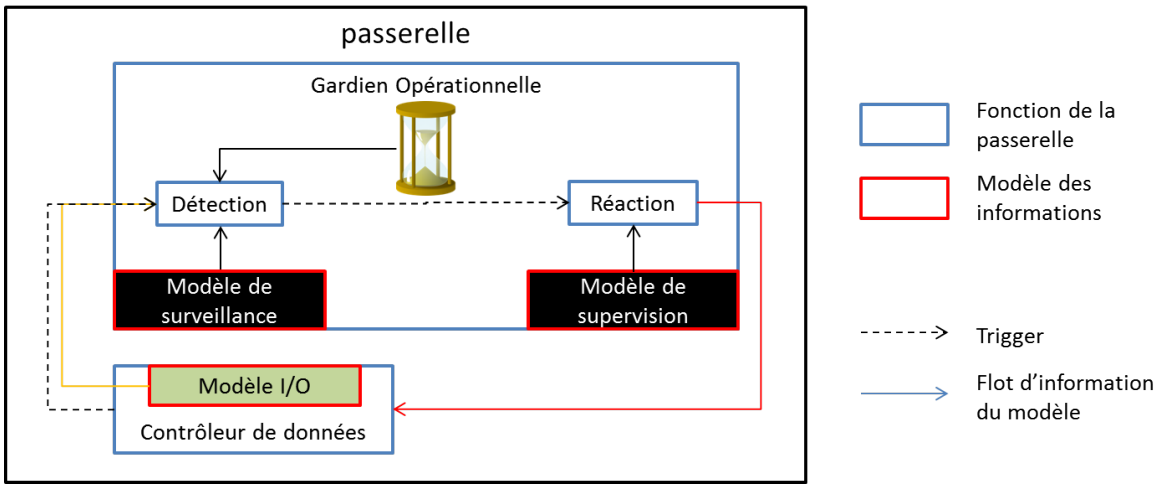


FIGURE 3.14 – Décomposition fonctionnelle pour garantir les exigences opérationnelles

Un diagramme de séquence est donné en figure 3.15, qui montre l'agencement des flots d'information entre les fonctions de cette chaine. Deux séquences y sont présentées :

— Une première séquence sans détection, qui nous montre le déclenchement de la détection par le contrôleur de données. Puis l'analyse par les chroniques en observant la mise à jour des variables si aucune défaillance n'est observée l'analyse est stoppée

— Une deuxième séquence avec détection, qui nous montre l'analyse comme la première, mais aussi le flot d'informations une fois une défaillance détectée. On voit ici que la réaction déclenche le contrôleur de données. Puis après consultation du modèle de supervision, elle prend une décision et l'impose par le contrôleur de données.

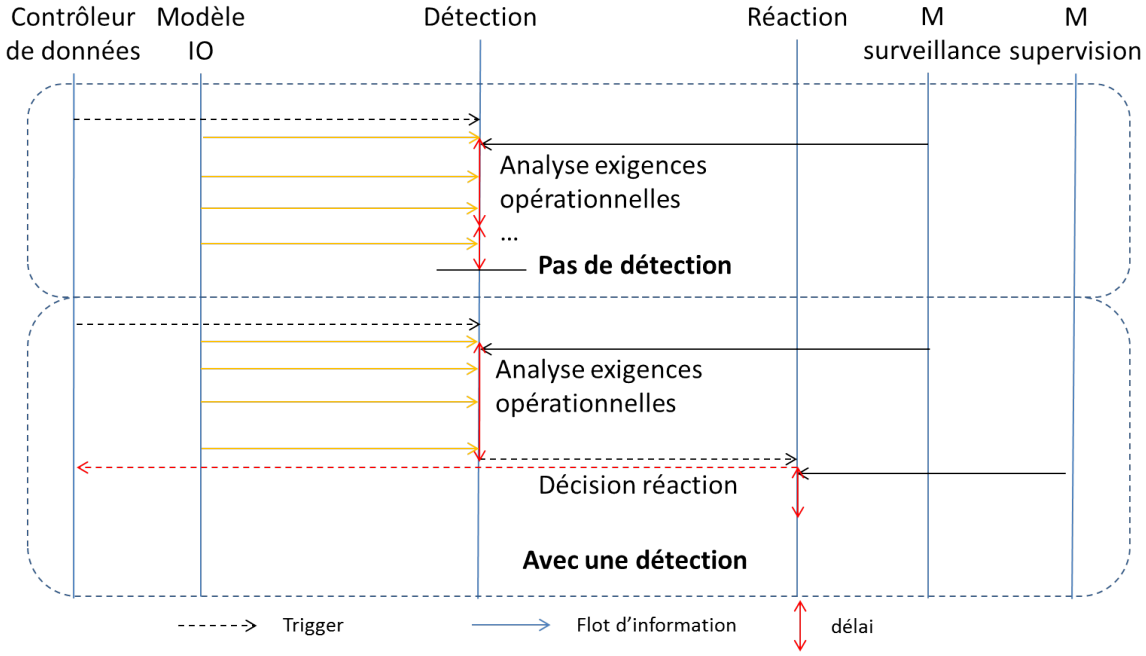


FIGURE 3.15 – Séquences de la chaine pour les exigences opérationnelles

Elle permet le diagnostic des équipements qui subissent une anomalie et informe des mécanismes haut niveau.

3.3.3.1 Le gardien opérationnel

Il a pour rôle de détecter les défaillances du système. La défaillance d'un équipement est définie comme la non-occurrence des événements qu'il génère et donc comme une perte partielle ou totale de ses fonctionnalités. Ce gardien met en œuvre la méthode de détection par signature décrite dans la section 3.2.1.1. Elle se fait en comparant les temporalités des changements d'état des variables, avec une chronique. Il met aussi en œuvre des réactions :

— la remontée d'information : En effet, ce gardien remonte directement les informations de détection au haut niveau du SAP ainsi qu'aux autres passerelles. L'information remontée est l'identification d'une défaillance.

— la mise en replis : En effet les conditions de ces contraintes sont aussi prises en compte dans la réaction réflexe, sous certaines conditions la neutralisation d'une partie ou de la totalité de l'équipement s'établit.

Le modèle de surveillance Ce modèle est basé sur le principe d'action réaction. Lors de la mise à jour d'un ordre, un ensemble de variables changera d'état à différentes temporalités. Cet état de fait est traduit en différentes chroniques pour chaque ordre du système.
Une chronique est une représentation des suites d'événement avec des temporalités menant à des défaillances. Elle fixe des intervalles temporels entre des événements de référence et des événements à observer. Ces intervalles bornent la fenêtre d'observation d'un événement. Une représentation est donnée en figure 3.16.

Le modèle de supervision Le modèle de supervision est composé d'une liste de log et de message pré formaté, afin d'informer les entités haut niveau du système ou l'homme de l'occurrence

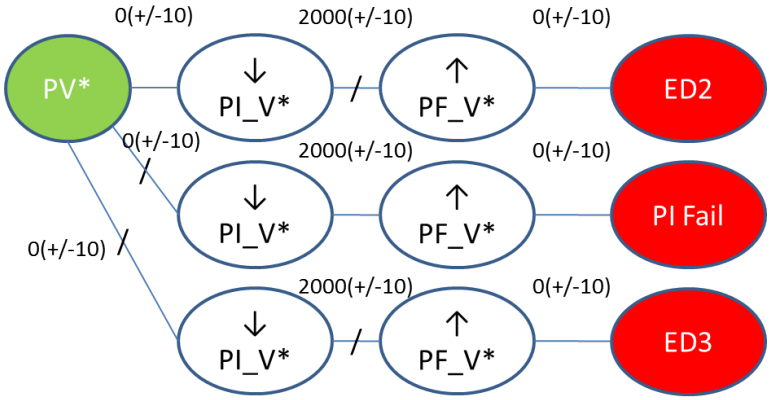


FIGURE 3.16 – Représentation d'une chronique

d'une défaillance. Il établit aussi une correspondance avec les différentes réactions possibles en fonction des résultats de détection.
La chaine de détection réaction temporelle permet donc l'analyse comportementale du système avec le temps comme référence. Elle permet d'observer les dérives ou de diagnostiquer les défaillances de celui-ci et d'y réagir. Cependant, cette analyse s'appuie sur les variables ; or celles-ci peuvent ne plus être intègres.

3.3.4 La chaine de détection réaction d'intégrité

Cette dernière chaine d'information a le rôle de détecter des incohérences dans les valeurs des variables. Aussi elle détecte les dérives dans les mises à jour de celles-ci et y réagit. La dérive d'un équipement est définie comme un décalage dans le temps des occurrences d'événement qu'un équipement génère, mais acceptables fonctionnellement. Elle est présentée en figure 3.17 et détaillée par la suite

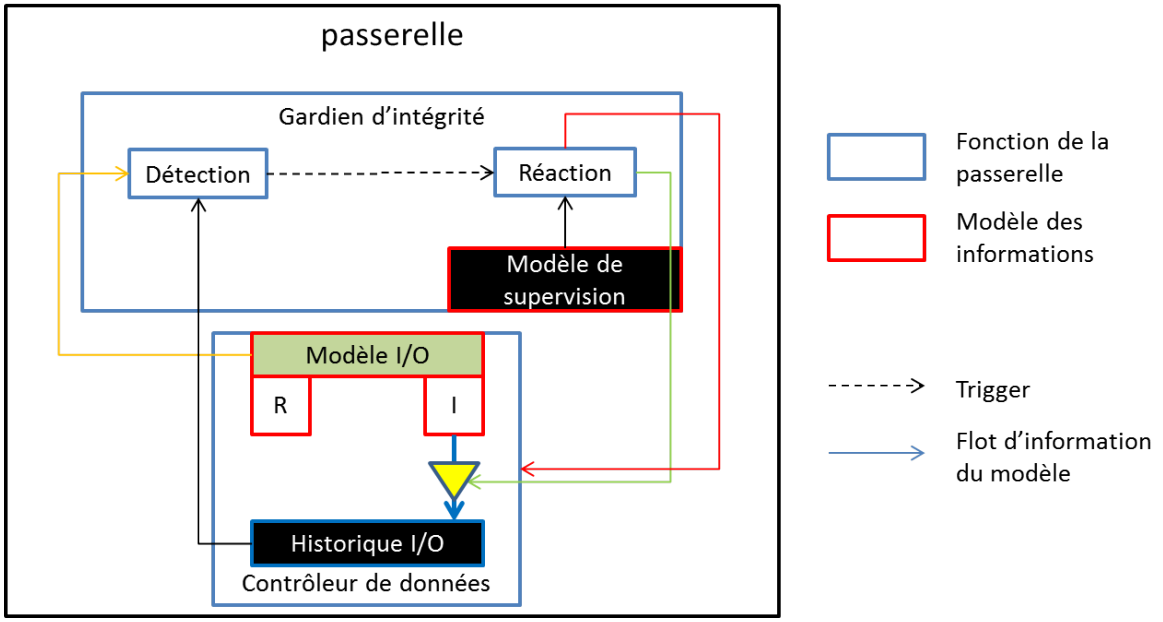


FIGURE 3.17 – Décomposition fonctionnelle pour les exigences d'intégrité

Un diagramme de séquence est donné en figure 3.18, qui montre l'agencement des flots d'information entre les fonctions de cette chaine. Deux séquences y sont présentées :

— La première séquence est sans détection. Le contrôleur de donnée déclenche la détection, qui s'établit avec le modèle de surveillance et l'historique de la variable. Puis la détection

déclenche la réaction et valide l'inscription des informations dans l'historique. La réaction déclenche le contrôleur de données sans réaction à établir.

— La deuxième séquence est avec détection. Elle est similaire à la première au commencement, mais la détection détectant une anomalie, elle ne met pas à jour l'historique et déclenche la fonction de réaction. La réaction déclenche le contrôleur de données. Puis après consultation du modèle de supervision, elle prend une décision et l'impose par le contrôleur de données.

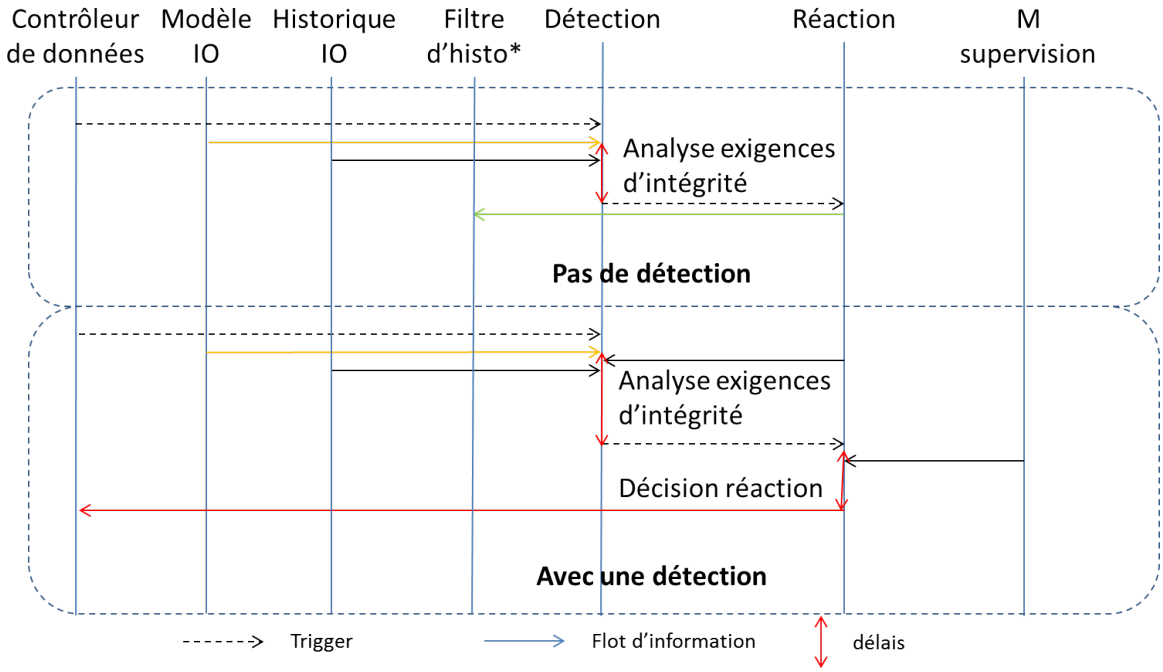


FIGURE 3.18 – Séquences de la chaine pour les exigences d'intégrité

La détection réaction avec historique a pour rôle de garantir les exigences d'intégrité des informations. C'est la condition qui vérifie qu'une information relate la bonne mesure de l'environnement sans altérations. Cette vérification est duale avec la variation de la valeur par rapport au temps. Deux fonctions sont mises en œuvre dans cette chaine :

3.3.4.1 Le gardien d'intégrité

Il met en œuvre une méthode de détection typée information. La détection repose sur l'observation des valeurs des variables et leur fréquence de mise à jour. Il compare ses observations à une référence dynamique interne à la passerelle, l'historique des I/O selon un algorithme 2.

Algorithm 2 Algorithme d'analyse d'intégrité

Require: x est une nouvelle information, t le temps écoulé depuis la dernière mise à jour, y la nouvelle valeur de la variable, u la dernière valeur mémoriser pour la variable, d la dernière variation mesurée, l la dernière période mesurer,

```
1:  function COMPARAISON(y, u, t, l)
2:      δ ← y − u
3:      if δ ≥ d(+marge) ⊕ δ ≤ d(−marge) then
4:          δ ← t − l
5:          if δ ≥ (marge) then
6:              if (y−u)/t = d/l then
7:                  return t
8:              else
9:                  return NOK
10:             end if
11:         else
12:             return t
13:         end if
14:     else
15:         return t
16:     end if
17: end function
18: while x = TRUE do
19:     σ ← COMPARAISON(y, u, t, l)
20:     if σ ≠ NOK then
21:         pas d'anomalie
22:     else
23:         anomalie détectée
24:     end if
25: end while
```

Ainsi une comparaison entre l'hystérésis observée et la précédente variation plus un seuil est réalisée, Puis selon le résultat de cette comparaison, on vérifie que l'évolution de cette valeur par rapport au temps entre les deux observations est égale à la précédente variation par rapport à la précédente période. Selon le résultat de cette comparaison, différentes réactions sont déclenchées.
Les réactions sont principalement passives (remontée d'information pour le haut niveau du SAP et les autres passerelles). Cependant, une réaction active est présente, mais elle ne concerne que la commande du contrôleur de donnée.

Le contrôleur de données implémente le mécanisme d'historisation des données. Ce mécanisme permet d'avoir une représentation dynamique dans le temps du système protégé. C'est cette représentation qui sert de base à l'analyse du gardien. Celui-ci commande ensuite la mise à jour de l'historique par sa réaction.
La dernière chaine d'information permet surtout de renseigner le haut niveau du système ou l'utilisateur. C'est lui qui diagnostiquera la cause et appliquera la stratégie complète de remédiation.

3.4 Conclusion

Ce chapitre nous a permis de présenter la passerelle ASSecIN. Nous avons introduit les concepts et méthodes qui garantissent la sécurité de l'IACS, mais aussi les contraintes qu'il faut prendre en compte. Aussi nous avons introduit son fonctionnement avec différentes chaines, qui garantissent

les exigences de sécurité pour le système. Ces chaines mettent en œuvre des méthodes de détection et réaction avec des références. Ces références étant complexes à formuler pour configurer la passerelle, nous avons mis au point des méthodes pour aider les concepteurs à configurer la passerelle. Elles sont présentées dans le prochain chapitre.

4 L'aide à l'intégration de la cybersécurité

« *Le poète ne doit avoir qu'un modèle, la nature ; qu'un guide, la vérité.* »

Victor Hugo

Sommaire

Figures

Le chapitre précédent a introduit une solution de détection et de réaction déployée sur les réseaux de terrain industriels. Il adresse notre premier verrou : Comment sécuriser un SAP? Différentes chaines d'information sont mises en œuvre pour atteindre cet objectif. Cependant elles se basent sur des références qui sont spécifiques au système protégé et qui sont complexes à renseigner.
Nous avons donc décidé de traiter un second verrou : Comment faciliter l'intégration de la cybersécurité dans les systèmes?
Le Lab-STICC et notre partenaire industriel ont déjà collaboré sur différents projets. La dernière collaboration a proposé une méthode générative pour la commande multi version de système transitique reconfigurable. Cette méthode a une approche hybride originale qui fait la synthèse de deux autres :

— l'ascendante, basée sur les composants d'un système.

— la descendante, basée sur les opérations d'un système.

Cette synthèse a été rendue possible par des travaux sur la génération conjointe de commande et d'interface de supervision [Bignon, 2012]. Cependant la prise en compte de la sûreté de fonctionnement ne s'établissait qu'au niveau du contrôle dans le code automate.
Or cela ne couvre pas nos problèmes. Les notions de surveillance et de supervision ont cependant été abordées. Nos travaux vont donc s'inscrire dans la continuité de ces travaux, avec la nécessité de fournir une vision orientée cybersécurité.
Dans ce chapitre, nous détaillons d'abord les besoins de la solution de sécurité. Puis nous présentons le flot outillé servant de base à notre démarche. Nous allons ensuite introduire nos contributions sur le modèle en entrée de ce flot.
Enfin nous introduisons la représentation sous forme de modèle de la configuration de la solution. Nous proposons une méthode de génération de configuration pour la passerelle. Ainsi qu'une méthode originale d'obtention de modèle de supervision et de surveillance s'appuyant sur le modèle en entrée du flot.

4.1 Les besoins de la passerelle

La passerelle ASSecIN sécurise un système en faisant passer les informations capturées sur le réseau entre PLC et IO à travers des chaines d'information. Ces chaines d'informations vont analyser l'information avec des cribles de détection. Si détection il y a, une réaction sera alors déclenchée dépendant de la méthode et du niveau de détection.
Or les méthodes que nous employons se servent de références pour détecter et réagir :
La première référence utilisée par toutes les chaines d'information est le modèle d'information de la passerelle. Ce modèle est une représentation de l'ensemble des variables du système à protéger. Il a un formalisme spécifique que nous avons défini séparant la valeur capturée sur le réseau et celle sécurisée . Aussi il contient des liens avec les références décrivant la sécurité pour le système. C'est par lecture de ce modèle que les méthodes servant à sécuriser ont connaissance de la/des contraintes ou du comportement à sélectionner.
Les références décrivant la sécurité ont des contraintes ou des comportements avec des formalismes exécutables par notre passerelle :

La référence de surveillance pour la sûreté est un ensemble d'arbres binaires sécurisant des variables. Chacun d'eux représente les états dans lesquels peuvent être les variables du système lors d'un événement sur la variable à sécuriser. La première version de ces arbres était avec plus de 4 niveaux de profondeur ils étaient donc complexes à concevoir pour la configuration.

La référence de surveillance pour les exigences opérationnelles est un ensemble de chroniques sous forme de graphes. Chacun d'eux représente des relations temporelles entre des variables observables dont la finalité est une défaillance. La première version de ces arbres était elle aussi complexe à concevoir pour la configuration.

La référence de supervision pour la sûreté est un ensemble de comportements sous forme d'automate à état fini. Chaque automate définit la réaction de la passerelle selon le résultat de la détection. Les conditions des transitions sont pour la plupart des automates, les états de validation des arbres binaires.

La référence de supervision pour les exigences opérationnelles est un ensemble de messages. Les messages sont formatés pour être intelligibles par les entités de niveaux supérieurs du SAP. Dans notre cas nous faisons remontons l'information à un humain.
Ces références doivent être configurées à l'initialisation de la passerelle. Cependant leur conception peut être ardue. C'est pourquoi nous nous orienté vers l'ingénierie diriger par le modèle. Afin de simplifier la génération de la configuration pour toutes les références de la passerelle.

4.2 Présentation du flot existant ComGEM

Nos travaux se basent sur ceux de [Bévan, 2013], dont le but était la génération automatique de code de contrôle multi-plateformes pour système reconfigurable. Ces travaux ont mis en place un flot de conception avec une approche architecture dirigée par les modèles, *Model Driven Architecture* (MDA) outillé et décrit dans la figure 4.1.

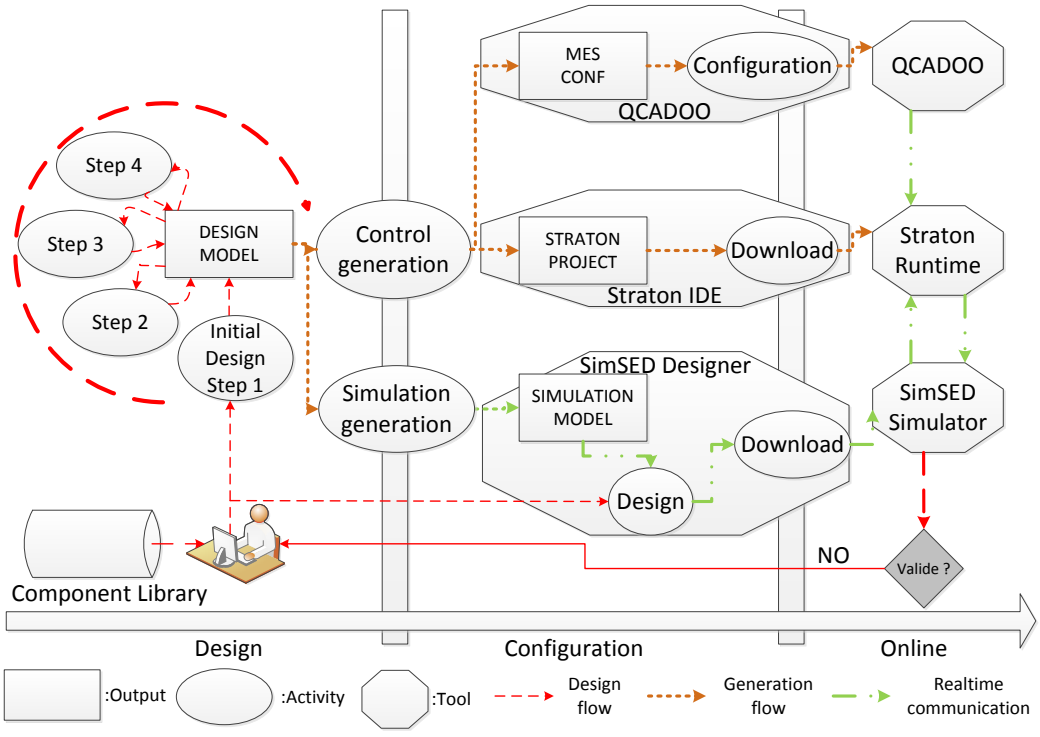


FIGURE 4.1 – Le flot de conception historique du projet

Les outils mis en place dans ce flot sont :

— ComGEM, il permet l'instanciation d'un système sous forme d'un modèle conforme à un méta modèle. Celui-ci a été élaboré par rapport à un DSL, il s'appuie sur des travaux antérieurs tout en intégrant la modélisation d'un MES. L'outil permet aussi la génération automatique du code de contrôle multi-plateforme par transformation.

— SimSED quant à lui permet la simulation du système modélisé. Il sera détaillé dans le chapitre traitant du démonstrateur .

4.2.1 Introduction à l'IDM

L'ingénierie dirigée par les modèles se démarque depuis quelques années des autres approches de l'ingénierie logicielle [Kuhn, 1989], En effet celle-ci propose de générer toute ou partie d'une application logicielle à partir de modèle. "Un modèle est une description ou une représentation conçue pour montrer la structure ou le fonctionnement d'un objet ou système [Landau and Besançon-Voda, 2001]" [Miller et al., 2001] ont quant à eux proposé la (MDA) permettant le suivi de l'évolution de plateforme. Puis en 2003 ils définissent une nouvelle approche pour les Nouvelles Technologies de l'Information et de la Communication (NTIC)[Miller et al., 2003], qui sépare la mise en œuvre de fonctionnalités sur une plateforme de leurs spécifications. Cette seconde approche fournit un ensemble d'outil, de concept, et de langages pour créer et transformer des modèles. Notamment le langage *Unified Modeling language* (UML) qui est le langage tautologique servant de référence ou de métalangage à d'autres, plus spécifiques. Le DSL quant à lui est adapté à un domaine métier particulier avec les concepts et notations utilisés par les experts, celui-ci permet la capitalisation du "savoir-faire métier" et d'apporter de la précision ainsi que de la compréhension dans le modèle, tout en étant conforme au métalangage. La conformité a aussi un intérêt dans les modèles, c'est ce que l'on se propose de vous présenter dans cette section.

4.2.1.1 L'organisation des modèles

Les modèles sont des représentations de ce qu'ils décrivent et sont hiérarchisés, comme montrés en figure 4.2. Un modèle fils est conforme à son père le méta modèle qui définit le langage d'expression de son fils. Le méta modèle est lui même un fils conforme à son père le méta-méta modèle. La limite dans les ascendants d'un modèle est un modèle tautologique (qui se définit lui même) [Fleurey, 2006] :

— *Meta Object Facility* (MOF) défini par l'*Object Modeling Groupe* (OMG) [MOF, 2006]

— EclipseMOF une variante de MOF définit pour l'*Eclipse Modeling Framework* (EMF) [Steinberg et al., 2008]

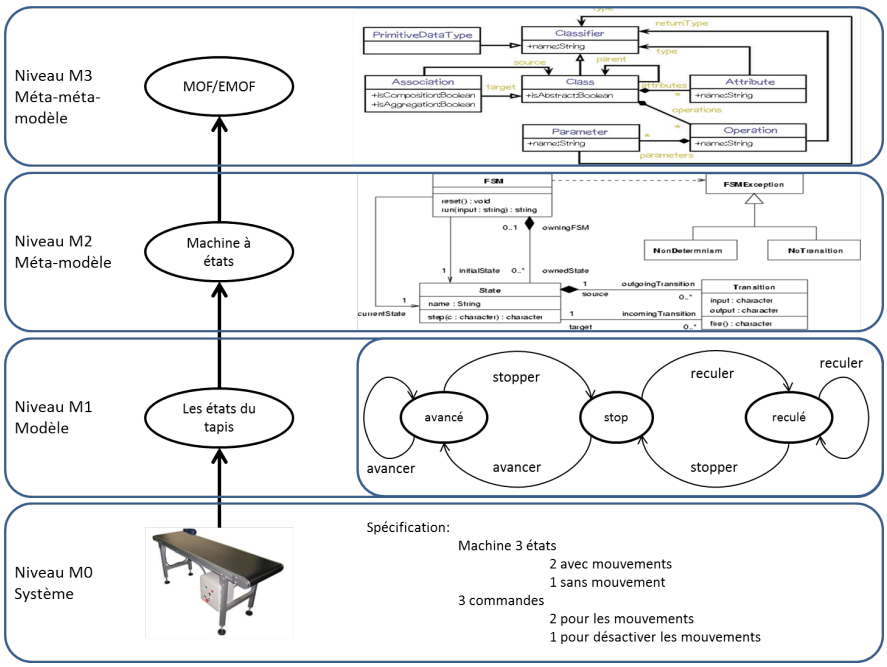


FIGURE 4.2 – La hiérarchie des modèles

Cette organisation hiérarchique est l'un des piliers de l'IDM. Elle permet la mise en place de méthode et d'outil rapidement et efficacement.
De grands formalismes existent comme les machines à état finis, l'algèbre (max, +) [Le Corronc et al., 2017], les chaines de Markov, les réseaux de Petri, les bond graphs [Pichard et al., 2017].

Chacun d'eux permet d'exprimer une problématique et de la résoudre avec des approches par simulation ou formelles. Cependant, l'intérêt d'un modèle est qu'il soit le plus complet possible. En effet ces formalismes étant décrit au niveau méta, plus un modèle est complet plus on peut avoir de formalisme pouvant s'y prêter [Aliyu, 2016].

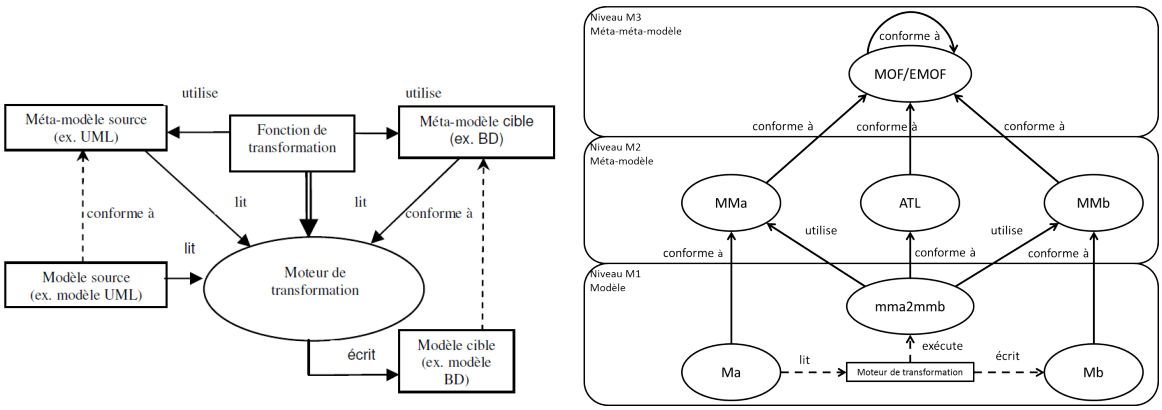
4.2.1.2 Introduction aux transformations de modèles

Les transformations de modèle sont un concept clef de l'IDM. Leur rôle est par parcours d'un modèle source, de fournir un modèle cible. Cette opération peut être de 2 types dont les objectifs sont différents [Allègre, 2012] :

— Endogène : les modèles sources et cibles sont conformes au même méta modèle. Le langage étant le même entre les deux modèles la sémantique est conservée.

— Exogène : les modèles sources et cibles sont conformes à des méta modèles différents. Les langages n'étant pas les mêmes entre les deux modèles, des règles syntaxiques doivent être mises en place pour la traduction. Il est aussi fréquent d'employer un modèle ou un langage transitoire "Glue", qui fait office de liant entre les deux modèles.

Chaque transformation à un coût qu'il soit temporel ou informationnel. Le processus général de transformation est présenté dans la figure 4.3a



(a) Le processus de transformation générale (b) Le processus de transformation ATL

FIGURE 4.3 – Le processus de transformation général et celui spécifique à ATL.

L'*ATLAS Transformation Language* (ATL) étant le langage de transformation que nous avons choisi d'utiliser nous le présentons en 4.3b Le moteur de transformation interne à l'outil parcourt un modèle source Ma conforme à un méta modèle MMa. Il exécute ensuite des règles stockées dans le modèle de transformation mma2mmb conforme au langage ATL. Ces règles utilisent MMa et MMb comme base et décrivent le comportement du moteur pour le niveau M2. Le moteur de transformation applique les règles pour chaque élément de Ma fournissant ainsi Mb Il écrit ensuite le modèle Mb conforme à MMb.
L'IDM est avant tout une approche permettant la mise en place d'outil et de méthode, pour représenter et solutionner des problématiques. Nous allons maintenant présenter nos outils de modélisation, qui posent les bases pour le flot de conception.

4.2.2 Présentation du modèle

Celui-ci décrit les différentes entités présentes dans un système industriel (plus particulièrement les systèmes transitiques) Ce modèle très complet décrit l'entièreté d'un système à des fins de conception : la partie physique, la partie contrôle ainsi que la partie commande ou MES. Nous allons nous intéresser plus particulièrement à trois éléments principaux de ce modèle :

4.2.2.1 Le composant

Cet élément du M2 est présenté en figure 4.4. Il représente les entités physiques d'un système industriel. C'est donc la description de l'interface physique de l'IACS :

— Les entrées sorties de commande : ce sont les variables du composant, celle d'information pour un capteur ou d'ordre pour un actionneur. Les variables globales servent d'interface avec les composants de niveaux hiérarchiques supérieurs.

— Les entrées sorties du flux physique : c'est le flux de matière ou produit décomposer en zone. Un composant peut être lié à plusieurs zones cela dépendra de l'opération qu'il réalise.

Le composant est aussi décrit selon des points de vue représentés par des vues détaillées en section 4.2.2.2 :

— La vue partie opérative : décrit le point de vue physique du composant.

— La vue topologique : décrit le point de vue topologique ou positionnement.

— La vue contrôle : décrit le point de vue du pilotage.

— La vue contrainte : décrit les comportements vis-à-vis d'autres composants.

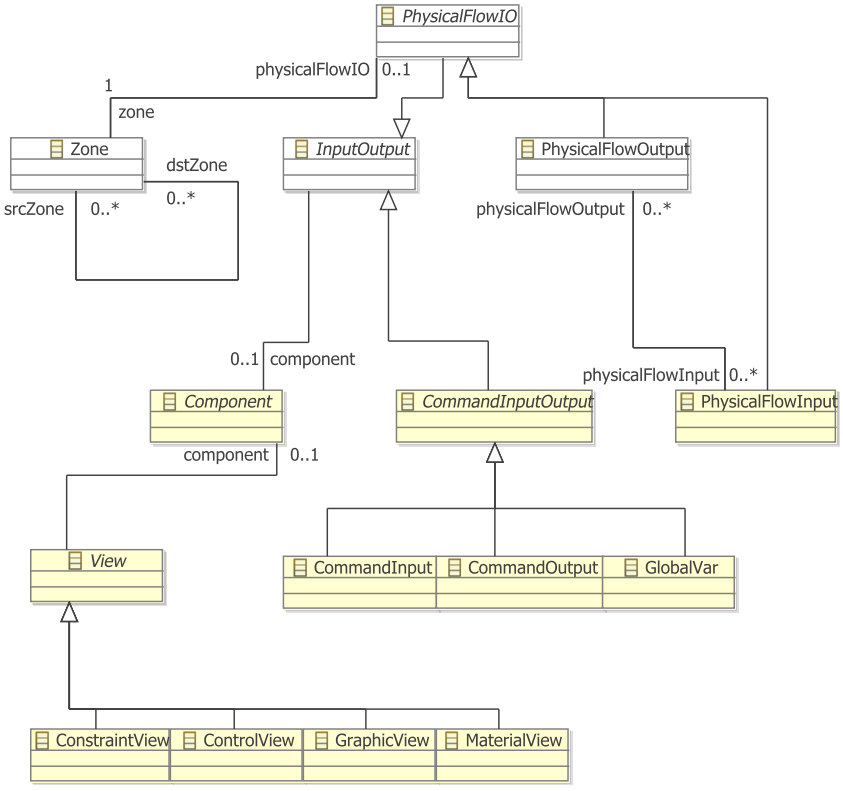


FIGURE 4.4 – Présentation de l'élément composant

Dans l'objectif de simplifier la mise en œuvre de la démarche de conception. L'élément composant générique a été décomposé en 2 types et plusieurs sous-éléments, cela est illustré en figure 4.5 :

— Les composants simples : ce sont les composants de bas niveau, déclinés en trois classes :

  — le Composant de Manutention, *Handling Components* (CM/HC) : c'est un composant déplaçant le produit avec un support comme un élévateur.

  — le Composant Support, *Support Component* (CSu/SuC) : c'est le composant qui supporte tous les autres ainsi que le produit.

  — le Composant de Base, *Basic Component* (CB/BC) : c'est un composant réalisant les opérations de base du système.

Ces trois composant sont de niveau 0.

— les composants agrégés : sont des compositions de plusieurs composants de niveau hiérarchique inférieur. Ils sont déclinés en trois classes :

  — le Composant de Base Enrichi, *Enriched Base Component* (CBE/EBC) : c'est un composant composé de CB et les liants les uns aux autres. Ce composant est défini comme de niveau 1.

  — le Composant Contextuel Effectif, *Effective Contextual Component* (CCE/ECC) : c'est un composant composé de CB ou CBE et d'un CSu. Ils les lient les un aux autres et est défini comme de niveau 2.

  — le Composant Système, *System Component* (CSys/SysC) : c'est un composant composé de CCE. C'est un composant unitaire défini comme le niveau 3.

Cette décomposition a permis de constituer une bibliothèque de composants et de mettre en place une hiérarchie. En effet des règles sont établies pour l'agrégation de composants dont voici quelques exemples :

— Un CBE ne peut être constitué que de CB, qui deviennent ses fils.

— Un CCE est obligatoirement constitué d'un CSu et de CBE ou CB, qui deviennent ses fils.

— Le CSys est le composant père, tout composant du système doit être lié à celui-ci par hérédité.

Cette hiérarchie est importante, car elle établit des règles pour la navigation dans un futur modèle de niveau M1, nous y reviendrons dans la section 4.4.

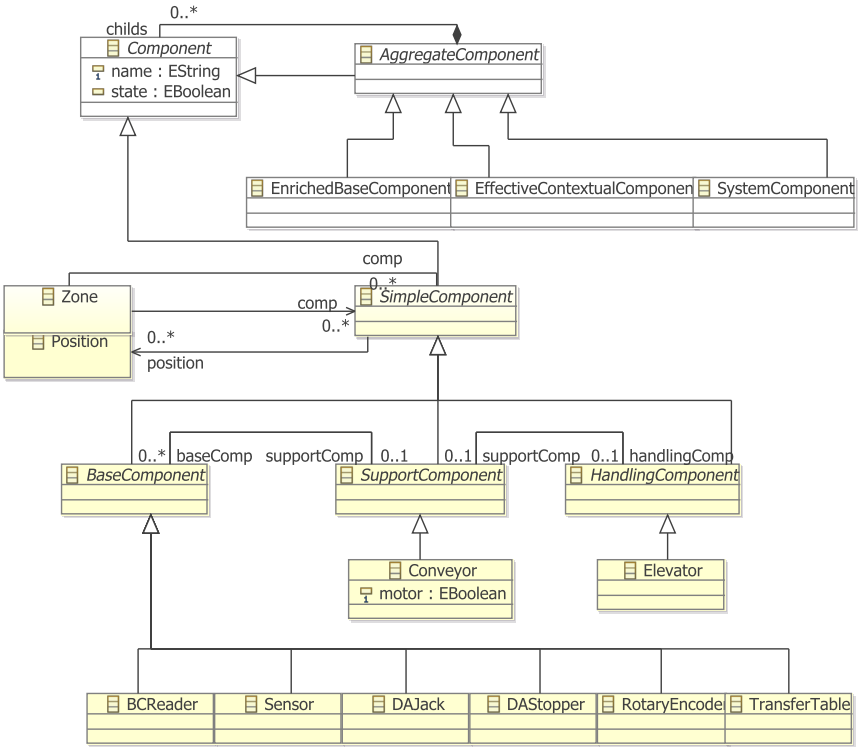


FIGURE 4.5 – Présentation de la hiérarchie des composants

"Un composant est un élément paramétrable, réutilisable et modulaire modélisant une partie d'un système. Il utilise un formalisme "boite noire" qui inclut des vues et des opérations pour le décrire."[Bévan, 2013]

4.2.2.2 La vue

Cet élément est spécifié sur l'hyper classe composant il est donc générique à toutes ses sous-classes. Quatre types de vue vont être détaillés dans cette section :

— La vue partie opérative illustrée en figure 4.6, représente les différents ensembles paramétriques pour chaque CB au niveau M2. Elle caractérise donc le composant qui lui est associé avec différents paramètres dans le niveau M1 :

  — des paramètres dimensionnels : hauteur, longueur, largeur.

  — des paramètres de position : position par rapport à un repère orthonormé (CSu)

  — des paramètres particuliers : caractéristique dépendante du composant : vitesse, accélération, décélération, longueur de course ....

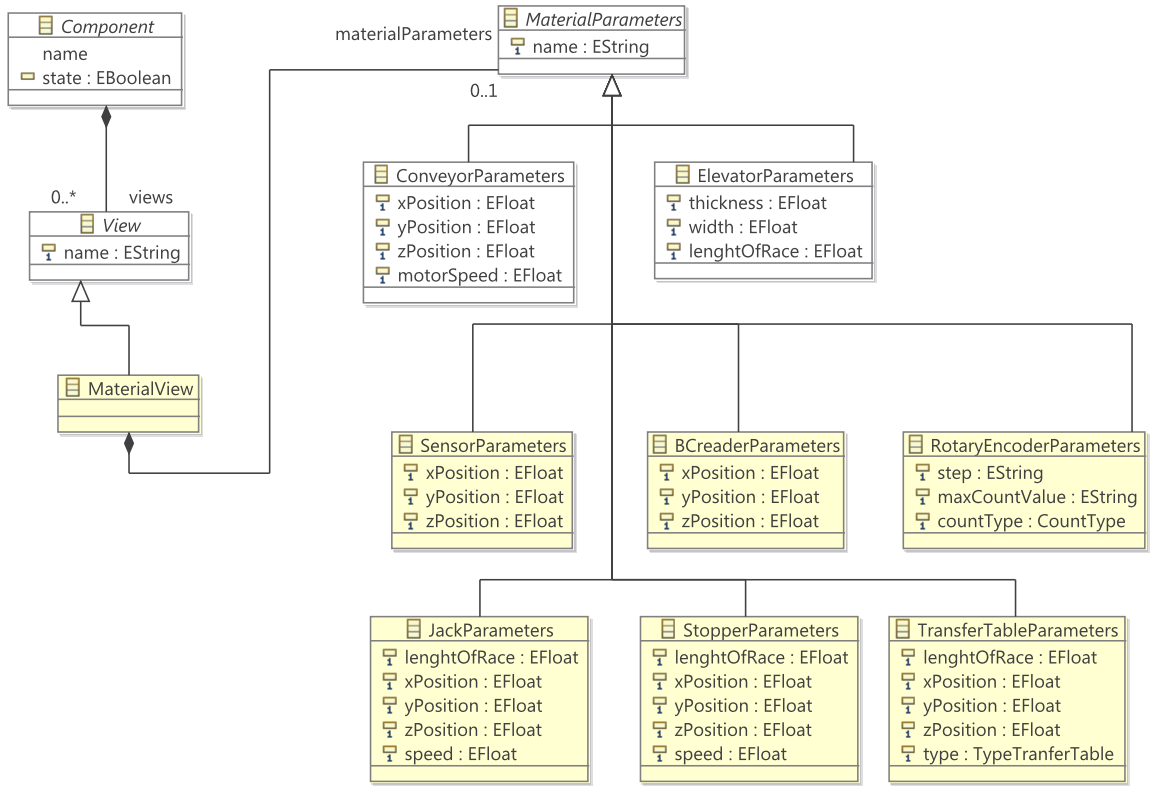


FIGURE 4.6 – Le méta modèle de la vue partie opérative

— La vue topologique présentée en figure 4.7, donne un point de vue topologique pour le composant. Seules deux sous-classes sont concernées par cette vue le CBE et le CCE. Le modèle spécifie des positions pour le CBE et des zones pour le CCE. Les positions et les zones sont virtuelles, mais elles aident dans l'expression des contraintes et la compréhension des opérations.

— La vue contrainte illustrée en figure 4.8, décrit un modèle de contrainte comportementale à haut niveau. Au niveau M1 c'est un ensemble de contraintes définissant le comportement du système, sous forme d'équation à condition pour l'activation ou la désactivation d'une opération. Les conditions sont simples ou doubles, les conditions doubles sont liées à d'autres conditions (doubles ou simples). Les conditions simples font référence à des opérations et leur état ou à des temporisations. Cette vue ajoutée par Romain Bévan permet par transformation d'enrichir la vue contrôle du modèle M1.

— La vue contrôle présentée en figure 4.9, décrit comment un composant peut être contrôlé (programme automate pour le composant). Au niveau M1 on trouvera deux types de modèles de contrôle :

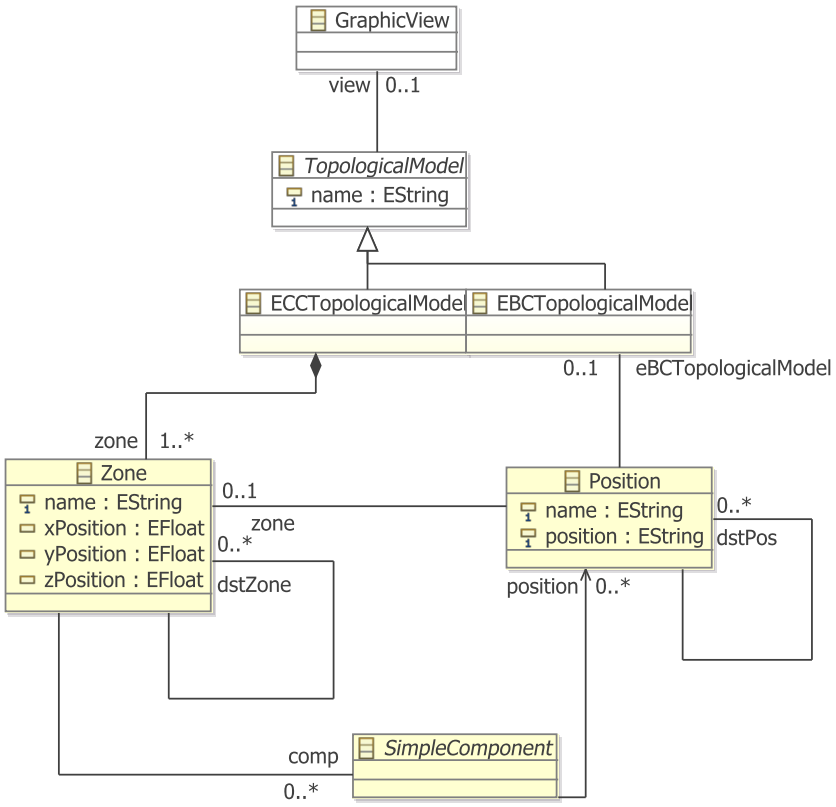


FIGURE 4.7 – Le méta modèle de la vue topologique

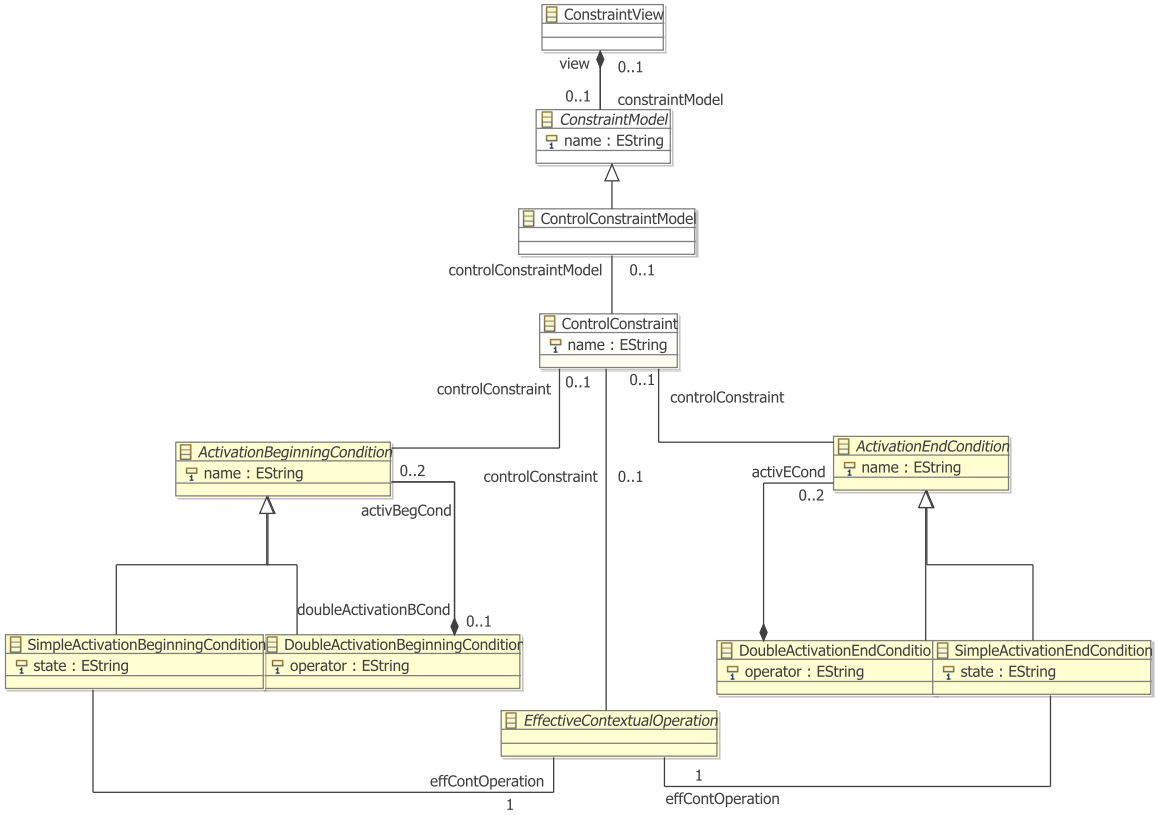


FIGURE 4.8 – Le méta modèle de la vue contrainte

— Le contrôle basique pour un CB.

— Le contrôle hiérarchique pour tout autre composant.

La différence réside dans l'interfaçage, en effet pour un composant de base le modèle de contrôle est interfacé avec les variables de commande. Alors que pour les composants hiérarchiques cet interfaçage est entre des variables globales. Aussi deux modèles de contrôle sont décrits le grafcet et le FBD. Il sera choisi par le concepteur ou par les transformations pour représenter le contrôle du composant au niveau M1.

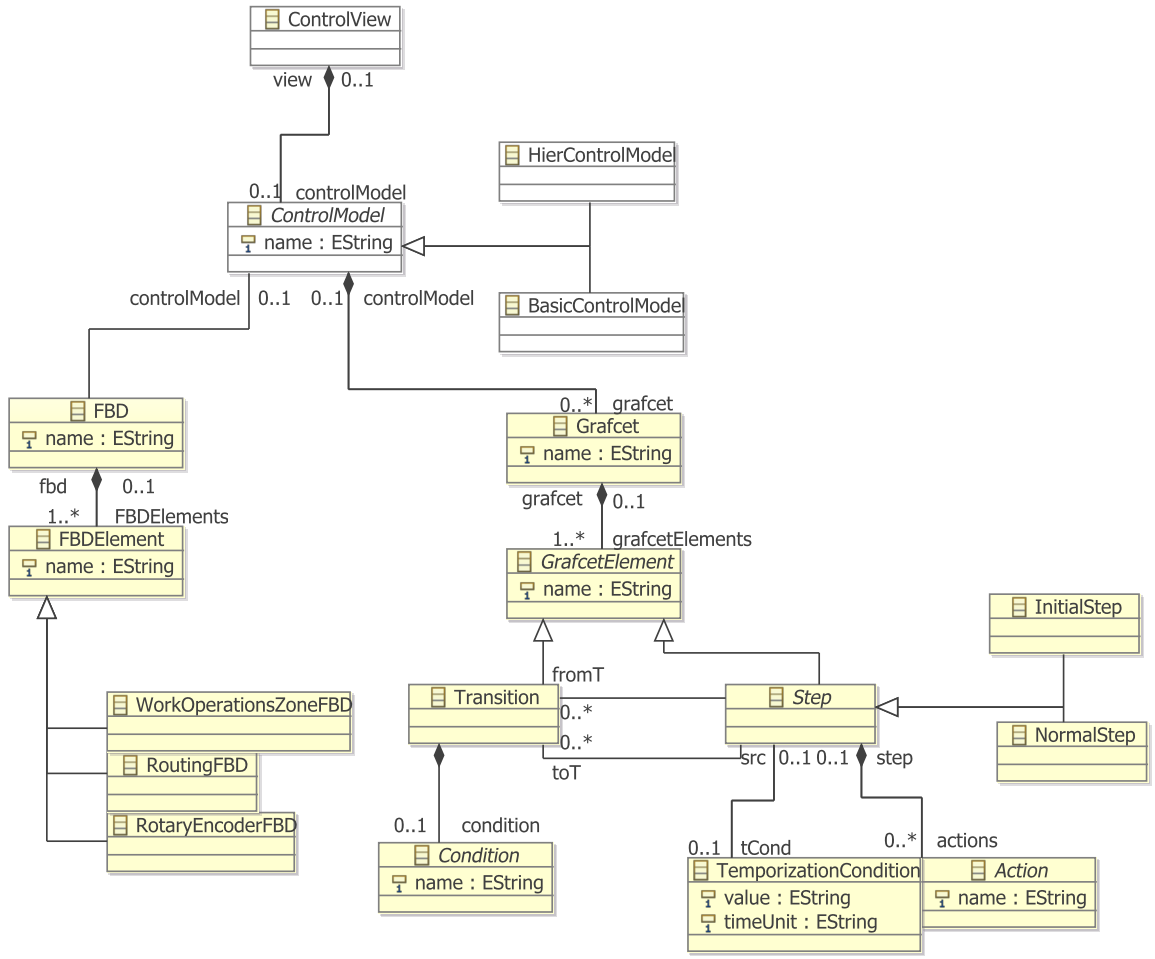


FIGURE 4.9 – Le méta modèle de la vue contrôle

4.2.2.3 L'opération

C'est le troisième élément qui nous intéresse. Il est présenté en figure 4.10. Il représente les fonctions mises en œuvre par la ressource physique représentée par le composant de manière hiérarchique. Ces opérations sont commandables par des variables globales. C'est donc la description de l'interface cyber de l'IACS. Celle-ci aussi a des attributs génériques pour sa classe et se décompose en 2 types :

— l'Opération Basique, *Basic Operation* (OB/BO), c'est l'opération que réalise un CB elle est principalement dépendante des entrées sorties de commande.

— les opérations agrégées : elle se décompose par niveau hiérarchique du composant.

— l'Opération Contextuelle, *Contextual Operation* (OC/CO), c'est l'opération que réalise un CBE et qui est dépendante d'un contexte opérationnel.

— l'Opération Contextuelle Effective, *Effective Contextual Operation* (OCE/ECO), c'est l'opération que réalise un CCE, elle est dépendante d'un contexte opérationnel et elle effectue une action sur le produit.

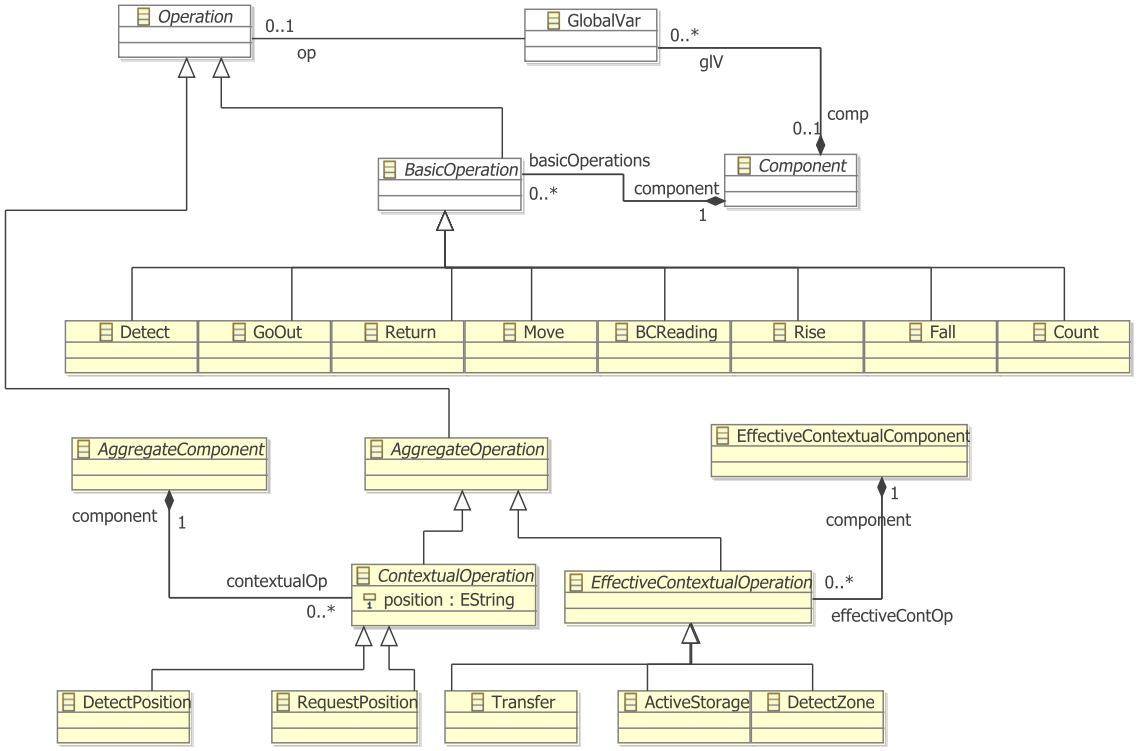


FIGURE 4.10 – L'élément opération du méta modèle

4.2.3 Les transformations du flot de conception

Le flot de conception ComGEM met en œuvre les deux types de transformation. Elles sont présentes à l'étape une, avec l'instanciation d'un site de production dans l'outil. Ainsi qu'à l'étape deux, la génération des différentes configurations pour la simulation.

Les transformations d'enrichissement permettent de générer de l'information de manière autonome. La lecture d'une première instance incomplète du modèle d'un site permet de l'enrichir. Les transformations ajoutent de l'information dans la vue contrôle des composants (les programmes de contrôle) par niveaux hiérarchiques. Au dernier niveau hiérarchique, la vue contrainte est exploitée afin de générer le programme de commande avec les contraintes du concepteur.

Les transformations de migration permettent de générer de nouveaux modèles conformes à des méta modèles différents. En effet in fine ComGem permet la validation d'une conception par simulation. Cependant il faut transformer le modèle issu d'un langage de conception en trois autres modèles :

— un modèle conforme à un langage de simulation pour le simulateur SimSED

— un modèle conforme à un langage de programmation pour une cible.

— un modèle conforme à un langage de configuration pour un MES.

Une fois ces transformations réalisées et leurs résultats mis en œuvre, la validation de la conception par simulation peut commencer.

Les trois principaux éléments ont été illustrés au travers du méta modèle des composants. Les transformations du flot ont aussi été décrites. Nous avons posé la base de nos travaux, qui vont maintenant vous être présentés avec les ajouts de notre démarche. Ainsi que la définition d'un nouveau langage décrivant la configuration de la passerelle.

4.3 Notre démarche pour l'intégration de la cybersécurité

Notre démarche vise à nous inscrire dans le flot de conception existant, outillé par ComGEM. Cependant elle ne doit pas interférer avec les précédentes réalisées sur le flot. Nous présentons en section 4.3.2 les ajouts au modèle d'entrée pour la prise en compte de la cybersécurité. Aussi nous introduisons en section 4.3.3 notre langage décrivant le modèle des références de la passerelle. Mais premièrement nous présentons le nouveau flot de conception.

4.3.1 Le nouveau flot de conception

Ce nouveau flot de conception reprend le précédent, mais en y intégrant la cybersécurité à chaque étape de celui-ci, comme montré dans la figure 4.11.

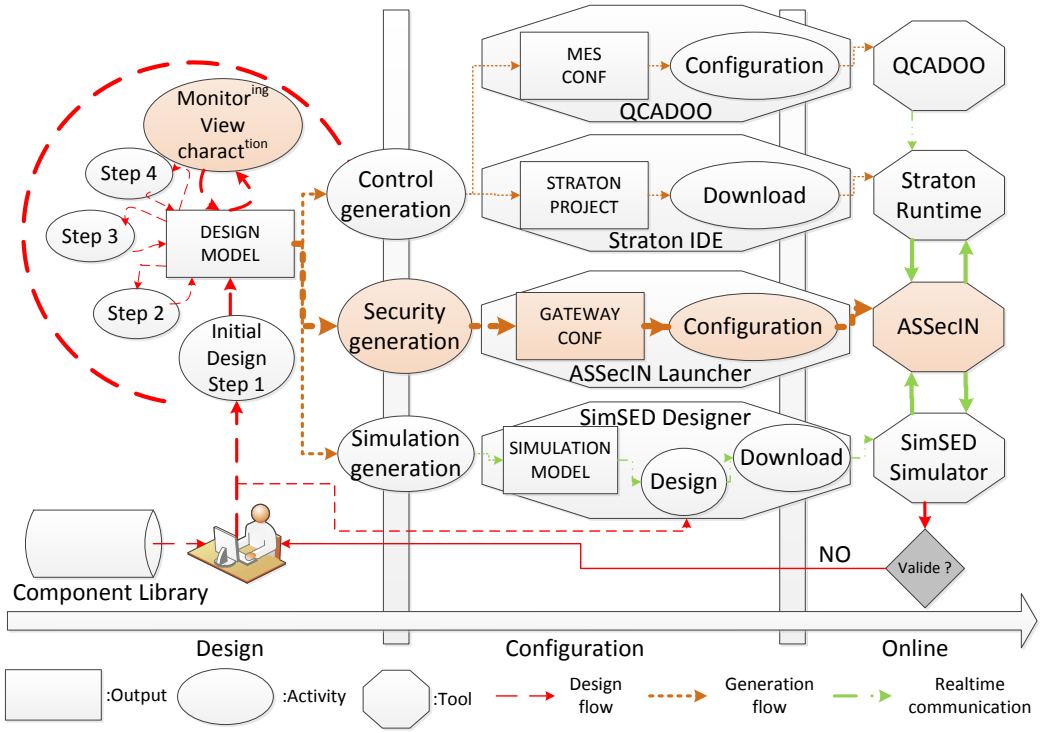


FIGURE 4.11 – Flot de conception actuel avec ASSecIN

Dans l'étape de conception, le nouveau modèle enrichi par la prise en compte de la cybersécurité est établi. Une méthode permettant l'auto-génération de cet enrichissement est donnée en section 4.4.1. Dans l'étape de génération, une nouvelle méthode est mise en place afin de générer la configuration de la passerelle. Cette génération du modèle des références est donnée en section 4.4.2. Dans l'étape de simulation, on y voit le placement de la passerelle dans le flot.
Cependant l'intégration de la cybersécurité nécessite des modifications dans le modèle en entrée du flot. Ainsi que la définition d'un modèle de sortie pour la configuration de notre passerelle.

4.3.2 Les ajouts au modèle d'entrée du flot

L'approche hybride par composant et opération du flot permet la mise en place de point de vue de sécurité. Ces points de vue vont nous permettre de renseigner les informations nécessaires à l'intégration de la cybersécurité dans les systèmes. Le modèle composant existant a donc été mis à jour, on a défini et précisé le contenu des vues surveillance et supervision comme montré en figure 4.12.

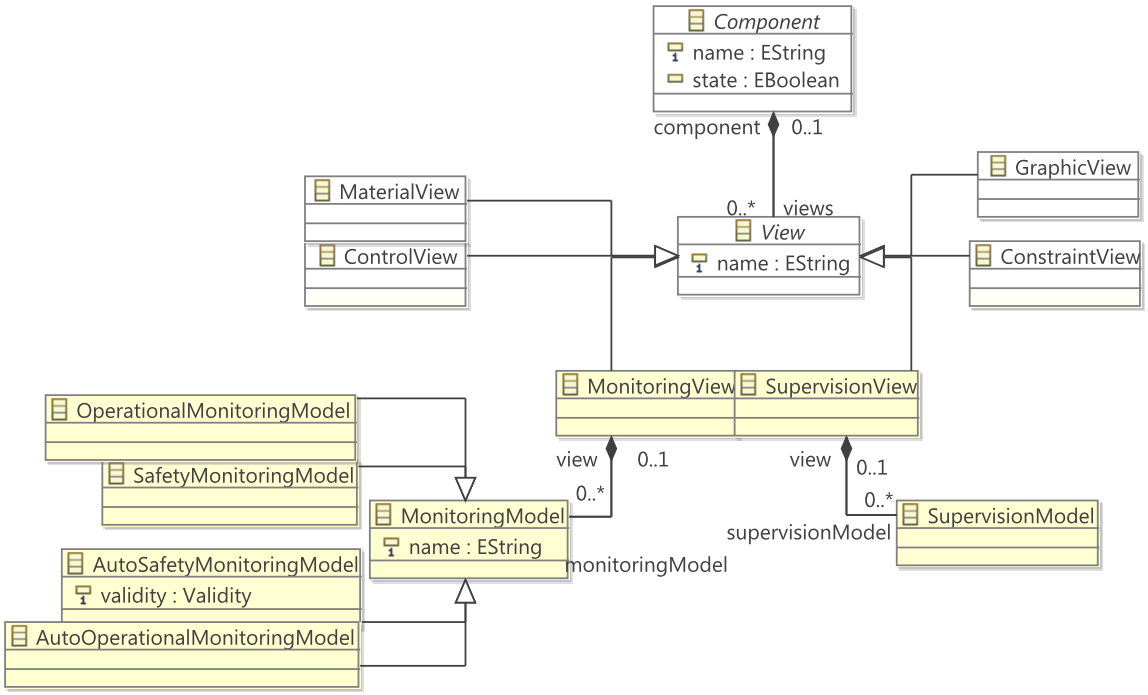


FIGURE 4.12 – Les vues d'un composant

4.3.2.1 La vue surveillance

La vue surveillance décrit pour chaque composant du système ce qui doit être observé, ainsi que la condition de sécurité à vérifier. Nous avons pris en compte que les modèles peuvent être instanciés par un concepteur, mais aussi par une transformation d'enrichissement.

Le modèle sûreté exprime les exigences de sûreté d'un composant et est présenté en figure 4.13. C'est un ensemble hiérarchique de contraintes lié aux variables I/O de celui-ci et de ses fils. Les contraintes sont des équations booléennes mises sous forme d'arbre binaire. Cependant celles-ci sont hiérarchisées selon la hiérarchie des composants. L'intérêt est que l'on peut définir le comportement selon le niveau hiérarchique. Ces contraintes sont instanciées par le concepteur ou générées automatiquement par transformation 4.4.1.1 et validées par celui-ci.

Le modèle opérationnel est présenté en figure 4.14. Il exprime les exigences opérationnelles d'un composant. C'est un ensemble de contraintes hiérarchique lié vérifiant la temporalité d'un événement. Les contraintes sont exprimées selon des Signatures Temporelles Causales décrites dans les travaux [Saddem et al., 2012]. Cependant celles-ci sont hiérarchisées selon la hiérarchie des composants. L'intérêt est que l'on peut contextualiser les événements non observables. Ces contraintes sont générées automatiquement par transformation 4.4.1.2 et validées par le concepteur. Elles sont aussi en partie imbriquées, car l'un des événements non observables n'est "Aucune défaillance observée". Celui-ci sert dans le niveau hiérarchique supérieur.

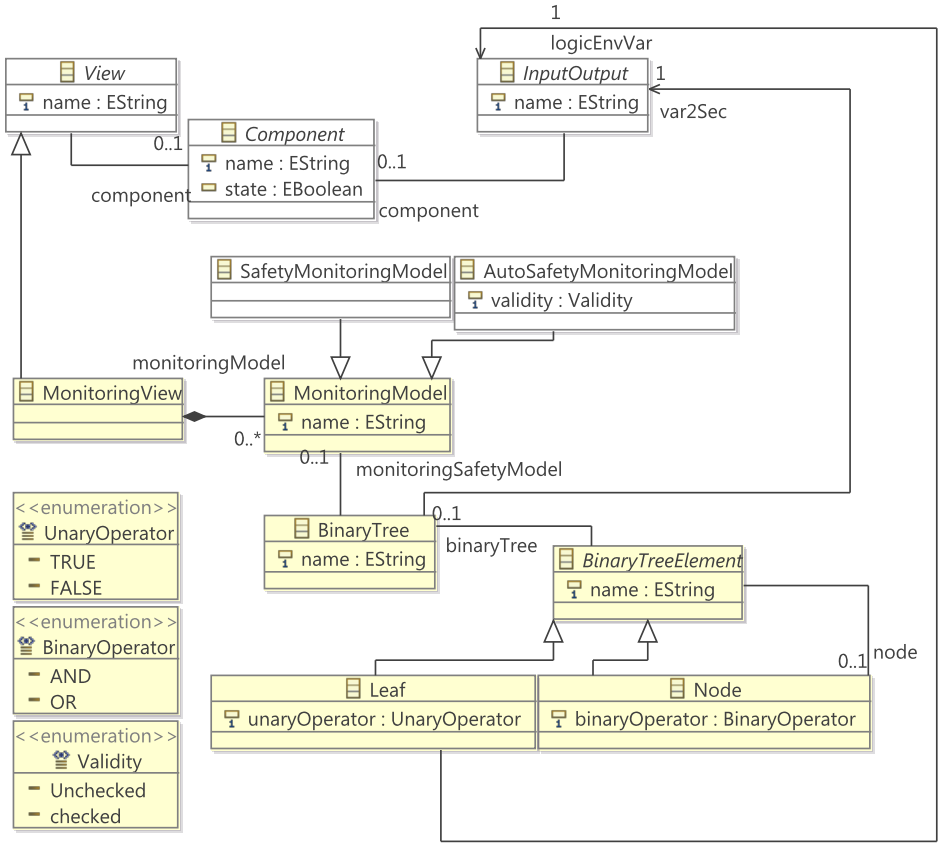


FIGURE 4.13 – Le méta modèle de la vue surveillance pour la sûreté

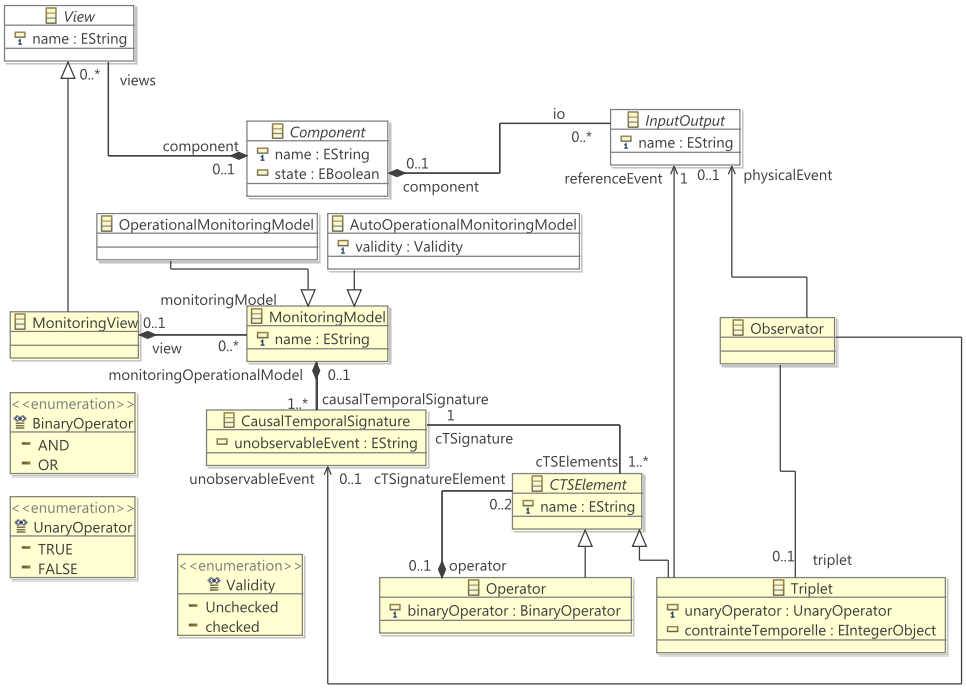


FIGURE 4.14 – Le méta modèle de la vue surveillance pour l'opérationnelle

4.3.2.2 La vue supervision

La vue supervision décrit pour chaque composant du système, les différentes options possibles pour la remédiation en cas de violations des contraintes. Elle est présentée sous forme méta dans la figure 4.15

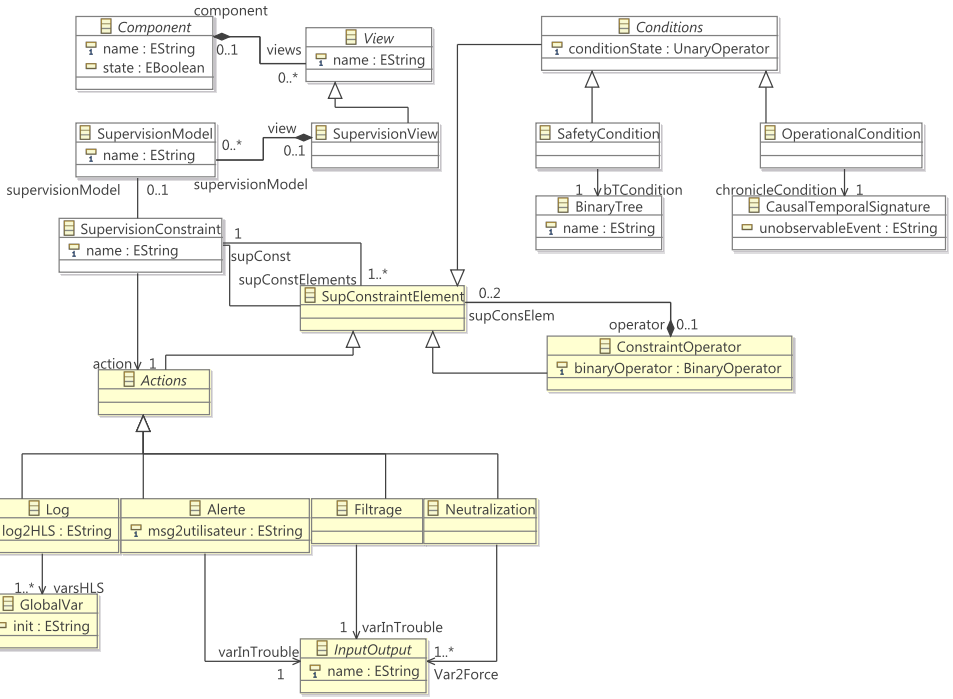


FIGURE 4.15 – Le méta modèle de la vue supervision

Le modèle de supervision est un ensemble de contraintes, regroupant logiquement des conditions, qui si celles-ci sont validées engendrent une action. Les conditions dépendent des contraintes de surveillance précédemment présentées :

— Signature temporelle causale.

— Arbre binaire.

Les actions correspondent aux réactions attendues dans la passerelle :

— l'alerte : c'est un message configuré par le concepteur qui indique le problème et la variable qui le posent.

— le filtrage : c'est un blocage de l'état d'une variable

— la remontée d'information : ce sont des log qui permettent à plus haut niveau dans le système de prendre des décisions.

— la neutralisation : c'est un forçage de certaines/toutes les variables dans un état.

L'ajout des vues surveillance et supervision au langage de conception permet la mise en place de contrainte de sécurité. Ces contraintes ont des formalismes différents qui spécifient :

— Les comportements autorisés du système (modèle de sûreté).

— Les défaillances possibles du système (modèle opérationnel).

— Les moyens de réponses face à la violation d'une ou de plusieurs contraintes (la vue supervision).

Ces formalismes sont spécifiques à la conception du système et peuvent changer pour la configuration de l'outil de sécurisation.

4.3.3 Le langage décrivant la configuration de la passerelle

La passerelle ASSecIN utilise des références pour la mise en œuvre de ses méthodes de détection et réaction. Or ces références sont spécifiques au système qu'elle protège et doivent être mémorisé dans passerelle. En effet l'objectif est de passer de l'outil de conception à celui de sécurisation. Cependant comme dit précédemment les transformations de migration ont besoin de deux méta modèles. Nous avons dû mettre en place une modélisation de ces références ainsi que leurs descriptions. C'est ce que nous proposons de vous présenter dans cette section.

4.3.3.1 La référence physique

Nous avons pris pour hypothèse que notre passerelle connait le système qu'elle protège. Une partie du modèle de référence servant à la configuration représente donc la partie opérative du système. Cette représentation se fait sous forme de variables dans le référentiel physique. Elle a ensuite été méta modélisée comme montré en figure 4.16.

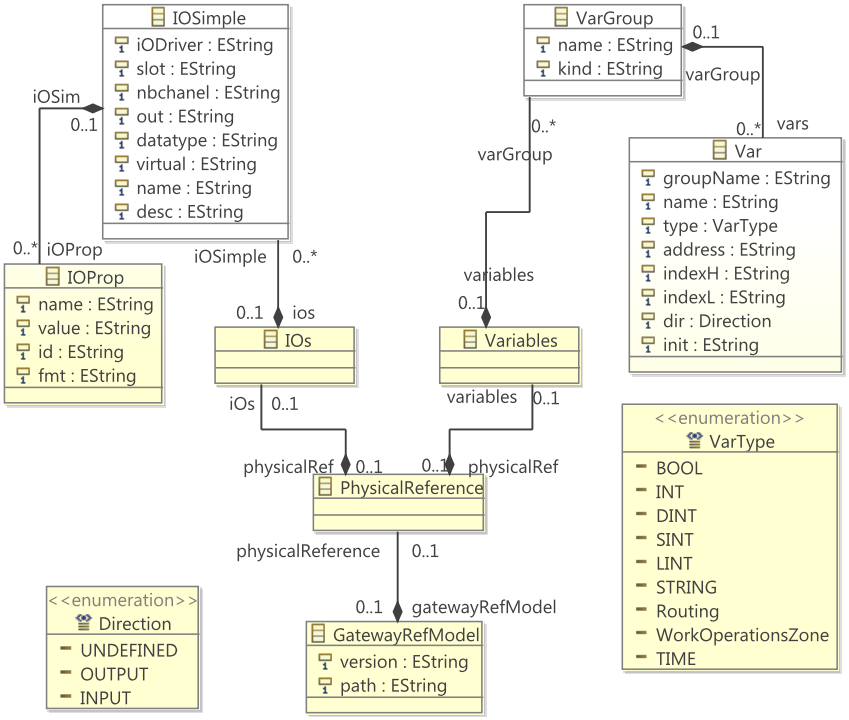


FIGURE 4.16 – Le méta modèle de la référentiel physique

On y constate plusieurs éléments, tout d'abord l'élément variable qui correspond aux informations et à leur regroupement point de vue cyber. Aussi on y trouve l'élément IOs, qui fait référence à l'architecture du FB et à comment recueillir l'information sur le FB. Le référentiel physique est donc la configuration pour le modèle I/O présent dans toutes les chaines d'information.

4.3.3.2 Les références de détection

Ces références servent à la configuration des modèles de surveillance présents dans les chaines d'information. Elles ont ensuite été méta modélisées :

La référence point de vue de la sûreté est utilisée dans la chaine de détection et réaction réflexe présenté en section 3.3.2. Cette chaine détecte la violation des contraintes de sûreté pour un système. La référence formalise ces contraintes en graphe de liaison quantitative, décrite dans les travaux de [EL OSTA, 2005]. Nous les avons formulées en arbre binaire comme montré dans

la figure 4.17, ce choix a été principalement motivé par la vitesse d'analyse et l'exactitude du résultat. En effet la vitesse d'analyse dépend de la profondeur de l'arbre, du nombre d'arbres et de la méthode de parcours. Le résultat d'analyse est l'ensemble des états des arbres. La chaine de détection réaction réflexe mettant en œuvre cette référence est exécutée pendant le flot de communication. Donc la vitesse d'analyse est indispensable pour ne pas engendrer de latence. Aussi cette chaine vérifie la sûreté du système. Donc l'exactitude du résultat est indispensable. En effet une contrainte de sûreté évalue si le système est dans un état permettant la mise à jour de la variable. Or s'il ne l'est pas, il faut le détecter de façon certaine, pour éviter que le système blesse un opérateur ou se détériore ou détériore le produit ou l'environnement.

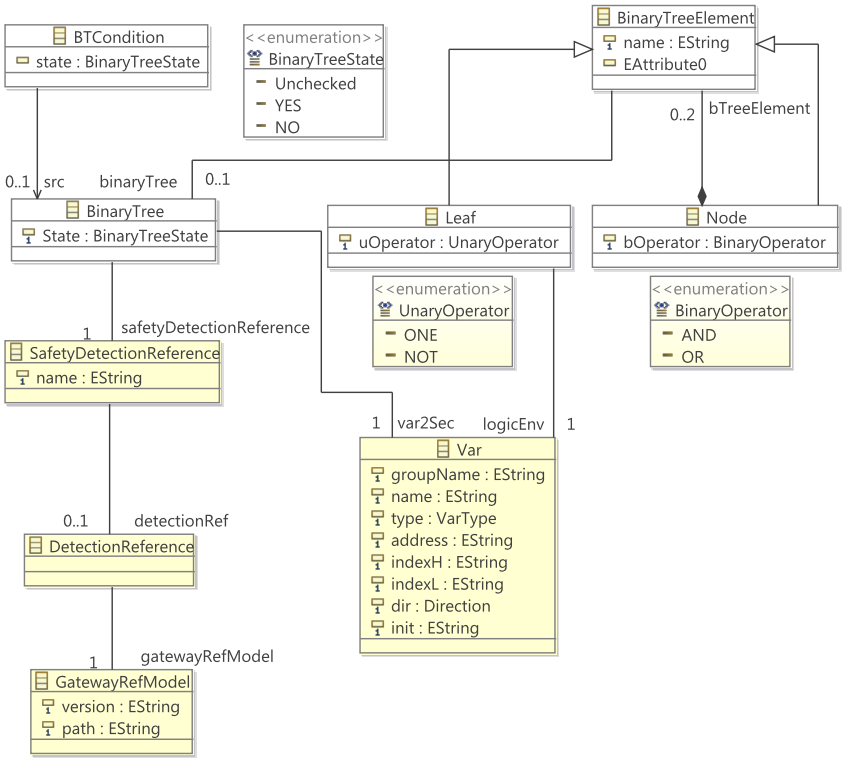


FIGURE 4.17 – Le méta modèle de la détection de sûreté

La référence point de vue opérationnel est utilisée dans la chaine de détection et réaction présentée en section 3.3.4. Cette chaine détecte une STC relatant une défaillance. Elle formalise ses STC en chroniques présentées dans la figure 4.18. Pour les modéliser, nous avons choisi un graphe avec place et arc semblable aux réseaux de Petri.Les places représentent des événements relatifs à une variable et son changement d'état. Les arcs quant à eux sont des intervalles temporels durant lesquels l'événement peut arriver. Les chroniques permettent principalement de vérifier la bonne exécution des opérations du système et d'aider au diagnostic lorsque ce n'est pas le cas.

La référence d'intégrité est formulée dans le modèle présenté en figure 4.19. C'est une liste de différents ensembles de paramètres par type de variable. Au niveau M1 chaque variable aura un ensemble de paramètres à vérifier dépendant de son type.

4.3.3.3 Les références de supervision

Les références de supervision définissent quelles réactions peuvent être entreprises, lorsqu'une violation des exigences de sécurité est détectée. Elles sont décrites par catégories d'exigences au niveau MM :

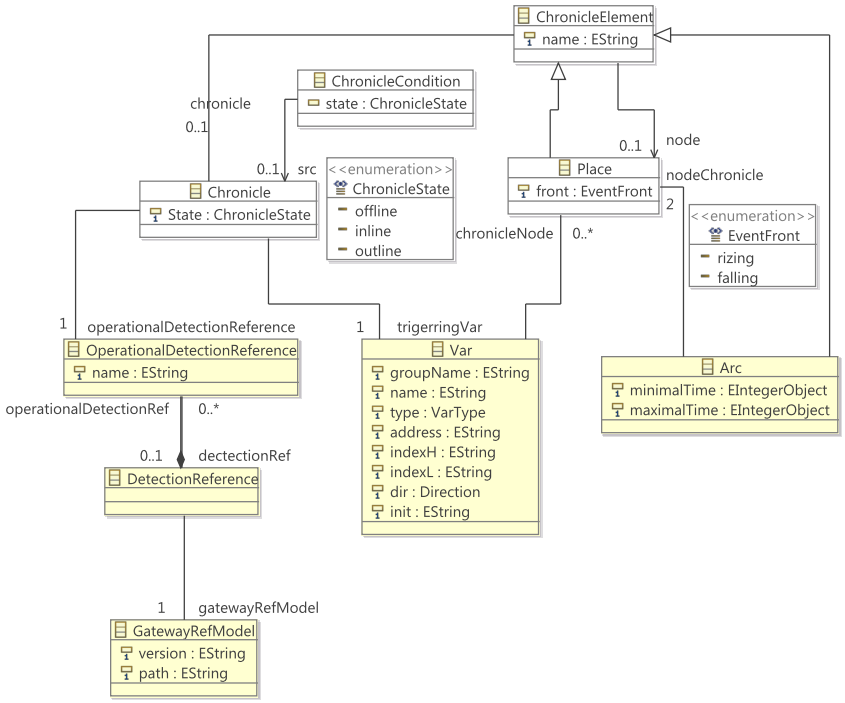


FIGURE 4.18 – Le méta modèle de la détection opérationnelle

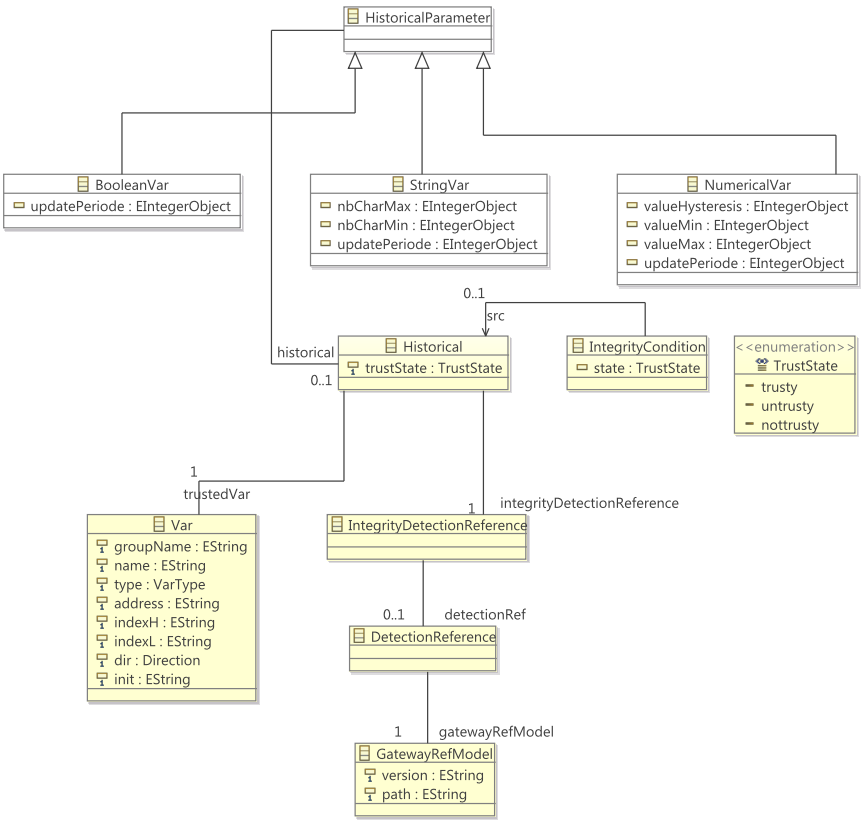


FIGURE 4.19 – Le méta modèle de la détection d'intégrité

La référence de sûreté est un automate à état décrit en figure 4.20. Celui-ci représente le comportement (les réactions) induit par la détection. trois types d'états sont définis : l'initial, le checking, la réaction Les états sont liés entre eux par des transitions dépendantes du résultat d'analyse ou d'une condition temporelle. La condition temporelle a pour but d'entrainer une réaction même si l'analyse n'aboutit pas.

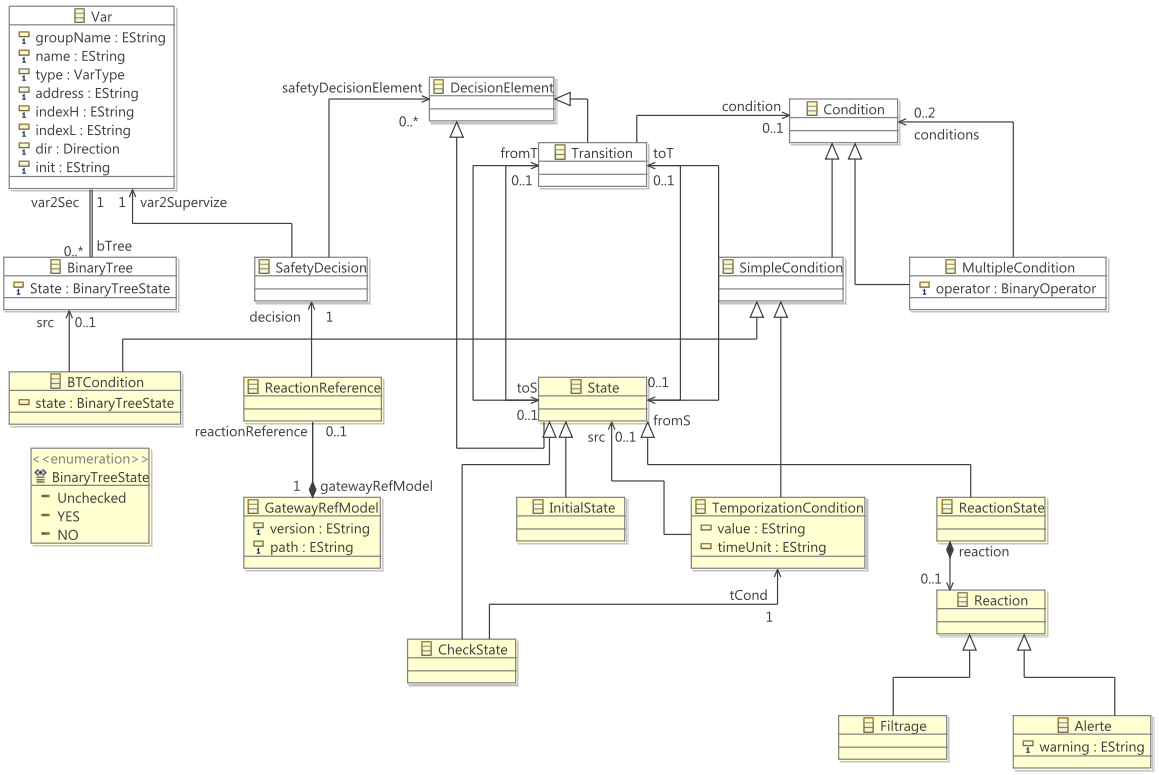


FIGURE 4.20 – Le méta modèle de la réaction pour la sûreté

La référence avec un point de vue opérationnel décrit la remontée d'information induite par la détection. Elle est présentée en figure 4.21. Elle décrit les différentes informations remontées : les défaillances d'opération, de composant actif, de composant passif. La sélection de l'information à remonter est faite par la chaine de détection réaction opérationnelle. Ici nous décrivons les différentes informations à remonter en fonction de l'analyse des chroniques.

La référence d'intégrité représente la remontée du niveau de confiance en une variable au haut niveau du système. Elle est présentée en figure 4.22. Cette information est purement qualitative cependant elle améliore le diagnostique d'une anomalie.
Nous vous avons présenté nos ajouts dans le langage de conception des systèmes industriel. Nous avons aussi présenté un nouveau langage permettant de décrire la configuration de la passerelle. La suite de nos travaux présentera l'obtention de la configuration de la passerelle dès la conception d'un système.

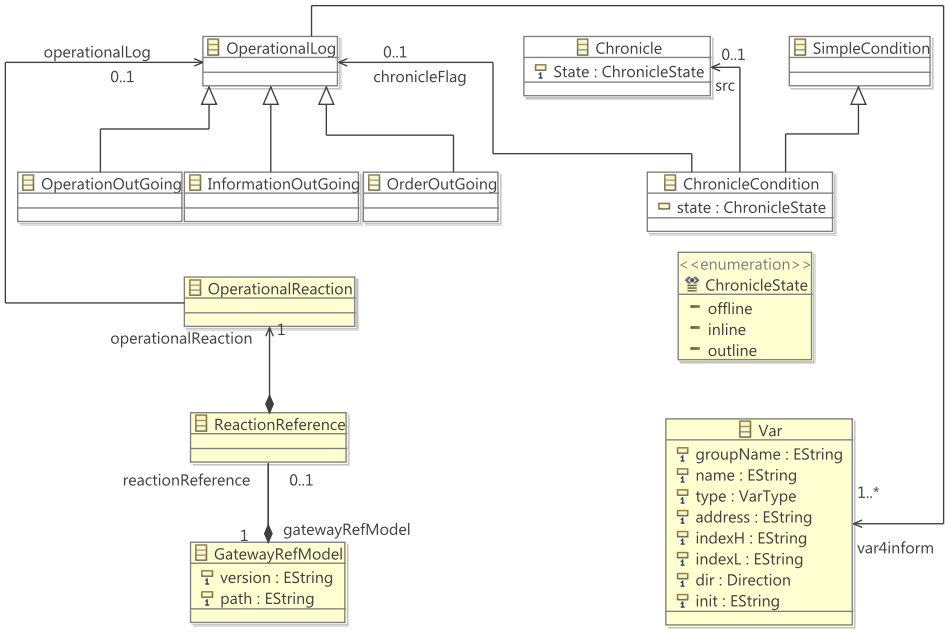


FIGURE 4.21 – Le méta modèle de la réaction opérationnelle

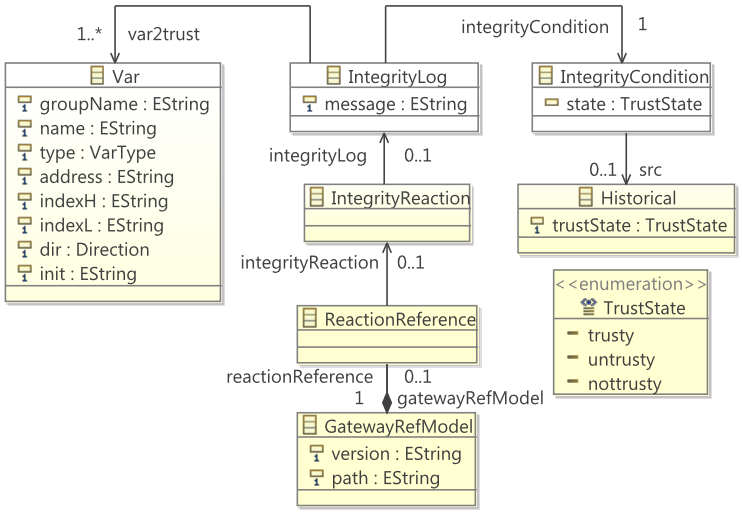


FIGURE 4.22 – Le méta modèle de la réaction pour l'intégrité

4.4 La génération de la configuration de la passerelle ASSecIN

Le modèle d'information interne à la passerelle décrit les I/O du système, ainsi que les références de détection et de réaction. À l'initialisation de la passerelle, le modèle interne est configuré pour la cible à protéger. Comme dit précédemment nous nous sommes inscrits dans un flot de conception existant pour générer la configuration de la passerelle. Nous avons présenté dans la section précédente le nouveau modèle en entrée du flot et le modèle en sortie. Dans cette section nous allons détailler les étapes du flot ayant pour finalité de générer la configuration. Une vision macro des étapes est donnée en figure 4.23. Deux types de transformation sont présents, celui d'en-

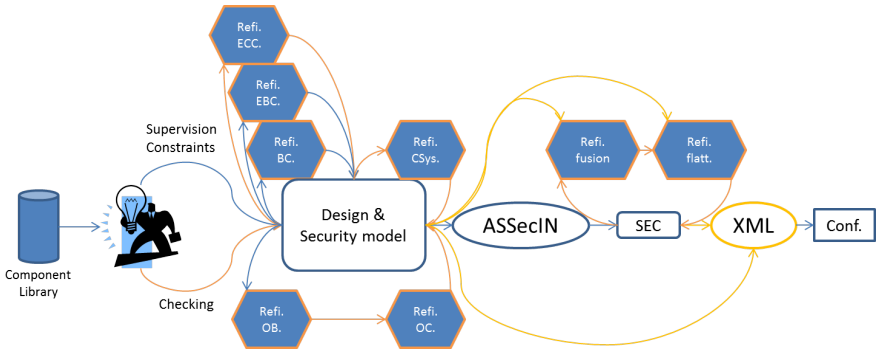


FIGURE 4.23 – Description macroscopique des étapes de génération

richissement qui auto-génère les modèles dans les vues du composant et celui de migration qui génère la configuration. Les enrichissements pour le modèle de sûreté dans la vue surveillance des composants sont hiérarchisés, mais indépendants les uns des autres, Une transformation instancie le modèle de sûreté pour tous les composants d'un même niveau hiérarchique. Les enrichissements pour le modèle opérationnel sont quant à eux hiérarchisés aussi, mais une séquence doit être respectée. En effet des événements observables à un niveau sont des événements de référence à un autre. Il faut donc réaliser les enrichissements selon une séquence, comme montré en figure 4.23. Après la validation du concepteur ainsi que l'instanciation du modèle de supervision, l'autre type de transformation intervient. La migration est elle aussi séquentielle en effet une première transformation réunie l'information et la met en forme conformément au modèle cible. Puis une deuxième traduit ce modèle en format XML pour que la passerelle puisse le lire et se configurer.
Le flot de générations de la configuration ayant été présenté de façon macroscopique. Nous allons maintenant détailler les transformations y prenant place.

4.4.1 L'enrichissement du modèle par auto génération

Ce type de transformation est utilisé principalement pour remplacer en partie l'humain dans le processus de conception. En effet les vues surveillance et supervision sont instanciable à la main, cependant la tache peu être complexe ou rébarbative. Nous avons donc mis en place une méthode en plusieurs étapes :

1. Une première version du modèle doit être instanciée par le concepteur :
   - avec tous les composants et leurs vues matérielles et topologiques.
   - avec toutes les opérations.
2. Puis les transformations d'enrichissement instancient automatiquement les vues surveillance et supervision des composants.
3. Le concepteur valide ensuite ces vues en validant toutes les contraintes de sécurité générées :
   - équation booléenne pour la sûreté.
   - STC pour les exigences opérationnelles.

4. Aussi le concepteur fera correspondre un élément de supervision aux contraintes de surveillance qui n'en ont pas.

Cette méthode faisant appel à de l'auto génération nous avons choisi de différencier les deux modèles de surveillance : le modèle de surveillance instancié manuellement et celui instancié par nos transformations. Nous nous proposons de les présenter :

4.4.1.1 L'enrichissement du modèle de surveillance de sûreté des composants

L'enrichissement est obtenu par une succession de transformations, celles-ci suivent toutes le même fonctionnement illustré en figure 4.24.

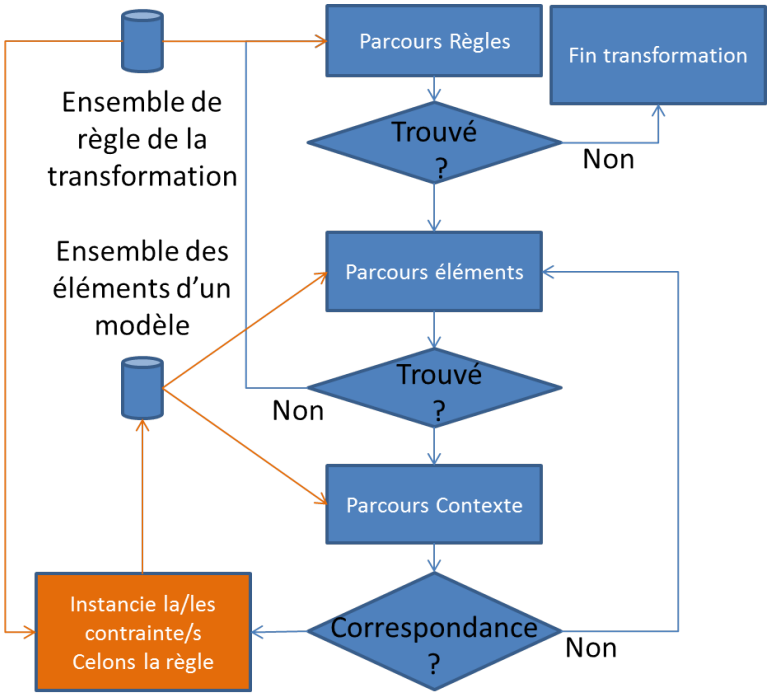


FIGURE 4.24 – Représentation fonctionnelle de l'enrichissement pour la sûreté

Chacune s'établit sur un niveau dans la hiérarchie des composants, pour caractériser la vue surveillance des composants de ce niveau, ce qui rend les transformations indépendantes. Les transformations Refi.(CSys/CCE/CBE/CB) en figure, 4.23 peuvent donc être exécutées dans n'importe quel ordre.
Ces transformations mettent en œuvre des règles dépendantes d'un contexte. Ce contexte permet d'identifier si la règle peut être appliquée sur le composant en fonction :

- de ses fils
- des opérations
- des positions
- des zones

Les règles sont donc des paternes préétablies pour un certain contexte, elles ne sont pas effectives pour tous les composants seulement ceux qui y correspondent.
Une fois la correspondance réglée, élément, contexte trouvé, la transformation se charge de remplir le paterne avec les informations extraites du modèle source pour former l'arbre booléen. Puis elle instancie le modèle auto-safety dans la vue surveillance du composant et continue son parcours du modèle source.
D'autres règles peuvent être ajoutées pour s'adapter au cas d'études que le concepteur désire sécuriser. Nous avons spécifié les règles relatives à notre cas d'étude, un exemple est donné en annexe 1. Cet exemple présente la règle qui définit l'ensemble des contraintes de sûreté pour le vérin

d'un CCE, avec comme contexte : que le composant réalise un transfert avec vérin dans l'axe du support, avec une butée en zone source et un capteur en zone source et destinataire. à ce niveau hiérarchique, les règles donnent 2 types de contraintes :

— l'une est relative aux ordres commandant les composants fils de l'ECC (butée)

— l'autre est relative aux informations remontées par les composants fils de l'ECC (cellule zone source et destinataire)

Le regroupement de ses règles est discuté dans les perspectives 6.2

4.4.1.2 L'enrichissement du modèle de surveillance opérationnel du composant

Il est obtenu par transformations successives. Elles suivent toutes le même fonctionnement décrit dans la figure 4.25.

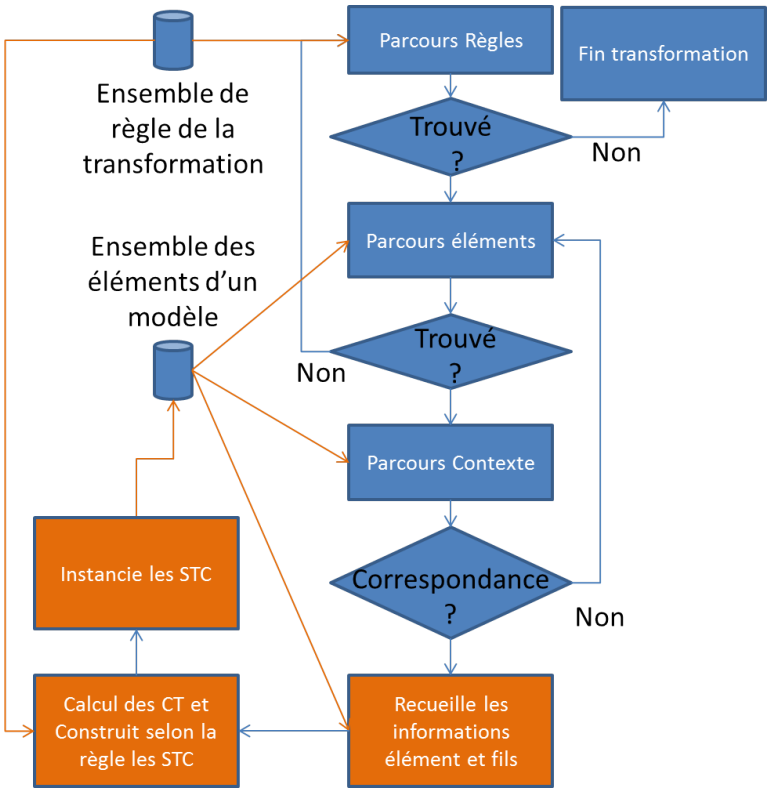


FIGURE 4.25 – Représentation fonctionnelle de l'enrichissement opérationnel

Comme la première elles sont cloisonnées à un niveau hiérarchique. Avec des règles spécifiques aux opérations de ce niveau et selon un contexte. Un exemple est donné en annexe 2.
Cet exemple traite de la génération des STC pour un vérin deux positions. Deux opérations contextuelles servent ici de référence : les requêtes pour les deux positions du vérin. Dix défaillances sont ici prises en compte :

1. la défaillance du capteur de position initiale de l'EBC pendant sa sortie
2. la défaillance du capteur de position initiale de l'EBC pendant sa rentrée
3. la défaillance du capteur de position finale de l'EBC pendant sa sortie
4. la défaillance du capteur de position finale de l'EBC pendant sa rentrée
5. la défaillance des capteurs de position finale et & initiale de l'EBC pendant sa sortie
6. la défaillance des capteurs de position finale et & initiale de l'EBC pendant sa rentrée
7. la défaillance de l'actionneur de l'EBC pendant sa sortie
8. la défaillance de l'actionneur de l'EBC pendant sa rentrée

9. la défaillance de l'actionneur de l'EBC à sa sortie
10. la défaillance de l'actionneur de l'EBC à sa rentrée

Cependant seulement 6 STC sont générées donc nous avons 10 inconnues pour 6 équations. L'identification de la défaillance est quasi impossible à ce niveau, car une STC peut décrire plusieurs défaillances. Seules les défaillances des capteurs position initiale pendant la sortie ou position finale pendant la rentrée peuvent être identifiées à ce niveau
Cependant des événements non observables de niveau inférieur sont utilisés dans la STC du niveau supérieur. L'intérêt est de pouvoir catégoriser avec plus d'exactitude la défaillance, en faisant transiter les événements non observables à travers les couches d'abstraction du système.
Ainsi avec l'observation de nouvelles variables la catégorisation est plus précise. Un exemple est donné en annexe 3. Une opération contextuelle effective sert ici de référence le transfert par vérin. Nous avons détaillé la STC détectant la défaillance du capteur de position finale. Elle observe les Événements de Défaillance (ED) des STC de niveaux inférieurs. L'ED 2 et & l'ED 4 sont relatifs aux défaillances 3,7 énumérées ci-dessus.
Une cohérence est donc établie entre les STC à travers la hiérarchie des opérations & des composants.

La vue supervision est instanciée manuellement par le concepteur, qui spécifie les contraintes de supervision. Cependant les conditions et actions sont déjà établies auparavant, mais non configurées.

4.4.2 La migration

Cette transformation suit un algorithme présenté en 4.26.

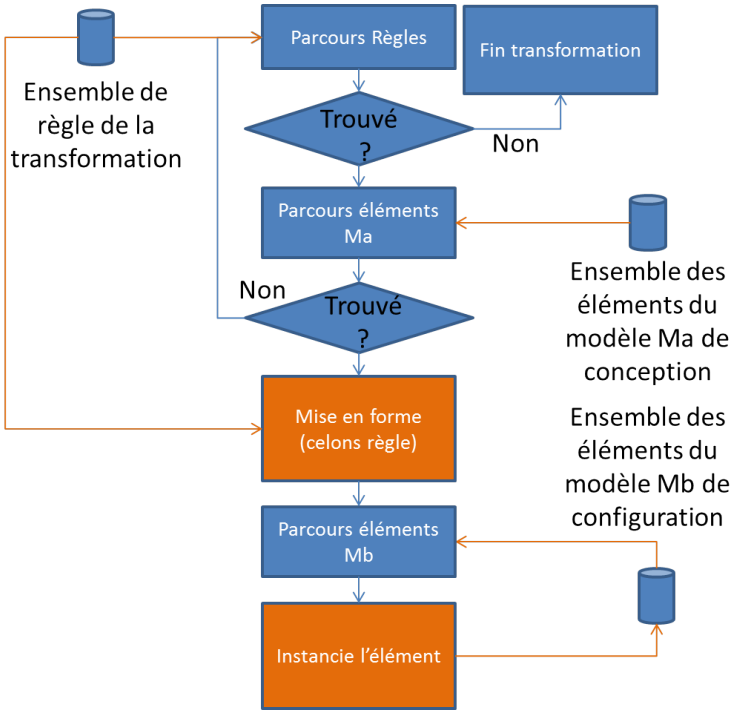


FIGURE 4.26 – Représentation fonctionnelle de la migration

Il permet la simplification du M1 à partir de règle et sa traduction en modèle pour configurer la passerelle. Ces règles sont établies par composant, elles permettent d'identifier l'élément souhaité dans le modèle source et de le traduire en l'élément du modèle cible. Les règles sont données en annexe 4. Elles nous montrent les éléments principaux et génériques, la référence physique (RP) avec les variables du système. Ainsi que les références de détection (RD) et réaction (RR) que nous allons détailler :

La référence physique a le même formalisme pour le modèle source et le modèle cible, car elle décrit les variables du système de la même façon avec les mêmes attributs. Ce sont d'ailleurs ces attributs qui servent de paramètre pour la méthode de détection par intégrité d'une valeur.

La référence de détection de sûreté ayant le même formalisme pour le modèle source et le modèle cible, l'objectif est le regroupement ainsi que la liaison avec la référence physique du modèle de configuration. Les règles sont données en annexe 5. Elle illustre le passage de l'élément du modèle source vers l'élément du modèle cible. Le résultat de cette transformation est présenté en annexe 8

La référence de détection opérationnelle n'a pas le même formalisme, En effet dans le modèle source c'est un ensemble de STC hiérarchisé correspondant chacune à une ou plusieurs défaillances, alors que dans le modèle cible c'est une chronique les réunissant toutes pour un composant. Les chroniques ont un événement déclencheur, des événements de défaillance unitaire et des événements à surveiller.
Plusieurs règles interviennent pour le changement de formalisme STC vers Chronique. Cependant nous avons constaté que des règles ne peuvent pas être exécutées durant la transformation de migration, car elle exploite le modèle cible "SEC" et source "Design" de la migration. Cela est montré en figure 4.23. Nous avons donc choisi de les placées dans la transformation permettant l'obtention d'un fichier de configuration en XML ou en raffinement du modèle "SEC", en effet celles-ci prennent en modèle source "SEC" et "Design". Nous allons détailler le principe des règles permettant le changement de formalisme.

Une première règle transforme les triplets , leur événement de référence et à observer deviennent des places dans le sous-graphe. La contrainte temporelle quant à elle est transformée en intervalle temporel, qui pondère l'arc reliant les deux places avec une marge de plus ou moins 10ms (le temps de cycle de notre automate). La règle est donnée en annexe 6 et décrite en figure 4.27.

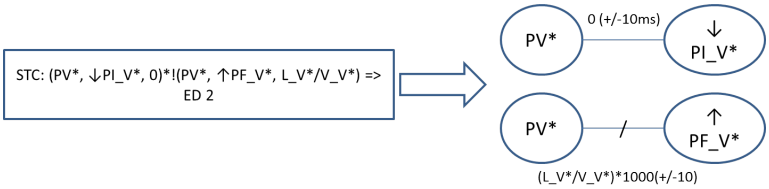


FIGURE 4.27 – Exemple d'application de la règle 1

Cette règle est exécutée durant la transformation de migration "ASSECIN", car elle transforme une STC en sous-graphe décorrélé.

Une deuxième règle gère la cohérence des sous graphes , la précédente générant un sous graphe par triplet, il faut rendre cohérent les sous graphes entre eux vis-à-vis de la STC qu'ils représentent. Les places identiques générées par la première règle de la transformation pour une même STC sont fusionnées. Cette fusion fait émerger la première place qui n'est liée à aucune autre en tant que destinataire d'un arc. Ce principe est illustré en figure 4.28

Une troisième règle gère la cohérence avec la STC , la précédente générant un sous graphe par STC, il faut rendre cohérent celui-ci vis-à-vis de la STC qu'il représente. La prise en compte des opérateurs de la STC permet de replacer les arcs tout en les recalculant. Cependant cette règle est dépendante de la précédente.
Ces deux règles sont exécutées durant des transformations de raffinement "Refi fusion" & "Refi flattening", car elles transforment les sous graphes décorrélés fournis par la transformation de migration, en un sous graphe corrélé représentant la STC.

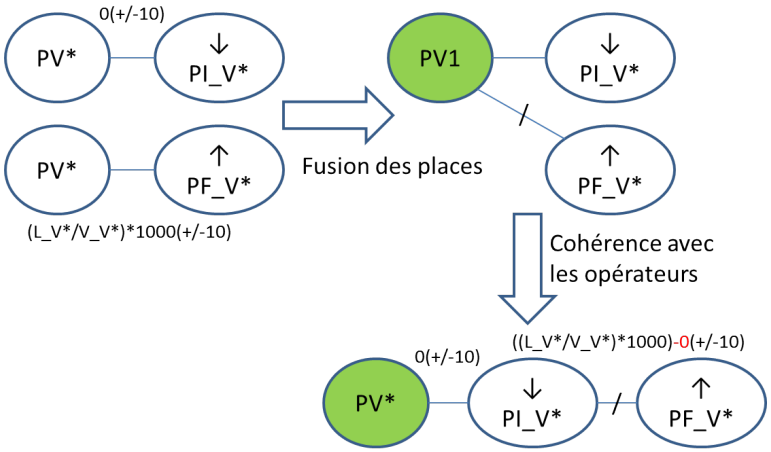


FIGURE 4.28 – Exemple d'application des règles 2 & 3

Une dernière règle gère la cohérence de la chronique , ajoute l'événement de défaillance de la STC à la fin du graphe la représentant. Aussi elle assemble les sous-graphes en chronique. Plusieurs chroniques sont générées par composant, car elles sont relatives aux opérations de celui-ci. La figure 4.29 présente la chronique pour l'opération requête position finale d'un vérin.

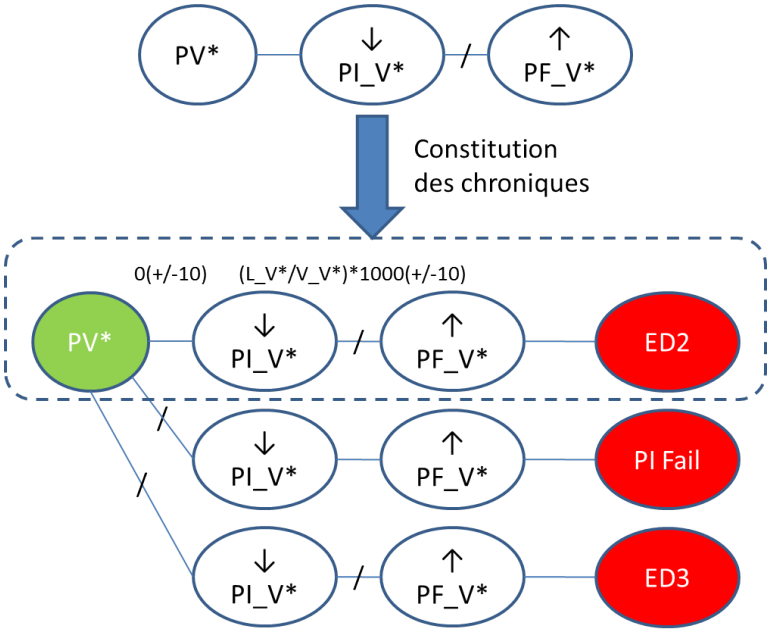


FIGURE 4.29 – Exemple d'application de la règle 4

La référence des réactions Les événements de défaillance sont traduits en log pour la partie réaction du modèle cible. Le résultat est présenté en annexe 10
Cette transformation est le pivot de notre travail. Elle génère la configuration de la passerelle en fonction de ce que le concepteur ou les transformations de raffinement ont spécifié.

4.5 Conclusion

Dans ce chapitre nous avons présenté une méthode orientée IDM, pour assister le concepteur dans la phase de configuration de la passerelle. Nous avons introduit l'approche l'IDM ainsi que les transformations, mais aussi ce sur quoi nous nous sommes basés pour construire la méthode. Nous avons ensuite présenté la méta modélisation du modèle d'information interne de la passe-

relle. Puis nous avons introduit nos modifications sur le méta modèle de conception, ainsi que les transformations permettant l'enrichissement automatique de celui-ci et la migration en modèle de configuration. Cette contribution a permis de simplifier l'instanciation de notre cas d'usage, afin de tester nos développements dans un démonstrateur. Celui-ci est présenté dans le prochain chapitre.

5 Le démonstrateur

« *Aucune investigation humaine ne peut s'intituler véritable science, si elle ne passe par la démonstration mathématique.* »

Léonard de Vinci

Sommaire

Figures

Tableaux

Le chapitre précédent a introduit une méthode assistant les concepteurs de système de production pour l'intégration de la cybersécurité. Cette méthode est issue de l'IDM et s'inscrit dans un flot de conception, fruit de précédents travaux. La méthode se décompose en quatre étapes :

1. l'instanciation des éléments primaires du système (Composant, Opération, Vue).
2. l'enrichissement semi-automatique du modèle.
3. la génération de la configuration.
4. la validation par simulation.

Les travaux décrits ici traitent du 3e verrou : Comment vérifier l'efficacité de la solution ?
Dans ce chapitre nous présentons le démonstrateur ASSecIN, il permet le test des méthodes de détection et réaction, ainsi que de la configuration contenant les références de ces méthodes. Nous présentons également le cas d'étude que nous avons conçu, le flot de conception, ainsi que les analyses et paramètres que nous avons déduits. Enfin nous introduisons nos scénarios d'attaque et/ou de défaillance et présentons nos résultats en termes d'aide à la génération et de sécurisation par la passerelle.

5.1 Présentation de la plateforme

Le démonstrateur dont le visuel est présenté en figure 5.1 est un assemblage de trois environnements :

1. Un environnement pour simuler le niveau 0 du CIM décrit en section 5.1.3.
2. Un environnement d'exécution pour le dispositif ASSecIn décrit en section 5.1.2.
3. Un environnement pour émuler les niveaux 1 et 2 du CIM décrit en section 5.1.1.

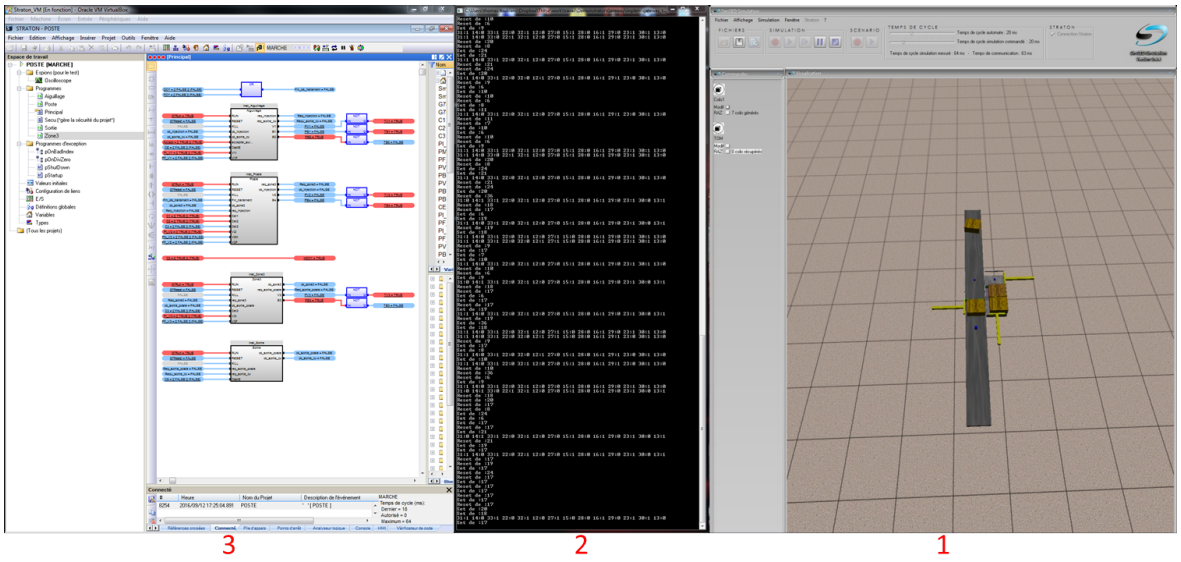


FIGURE 5.1 – Le visuel du démonstrateur ASSecIN

Ce démonstrateur virtuel a plusieurs intérêts : Il permet d'observer les effets d'anomalies sur l'environnement du système et sur le système lui-même par simulation. Il est proche de la réalité, car la décomposition respecte les niveaux hiérarchiques du CIM. Toutes les briques (niveaux hiérarchiques) sont présentes dans le démonstrateur : le MES, l'automate, les équipements, la station d'ingénieur.
La communication est l'une des originalités de notre démonstrateur. L'utilisation d'un protocole client-serveur se rapproche des implémentations cyberphysiques de l'I4.0. Aussi l'assemblage d'environnements, les cloisonnent les uns par rapport aux autres tout en les liant par des canaux

de communication. L'assemblage permet de rendre compte au plus près de la propagation possible d'une attaque. En effet les trois environnements s'exécutent sur des machines virtuelles et les communications passent par des interfaces réseau physiques. Une représentation est donnée en figure 5.2

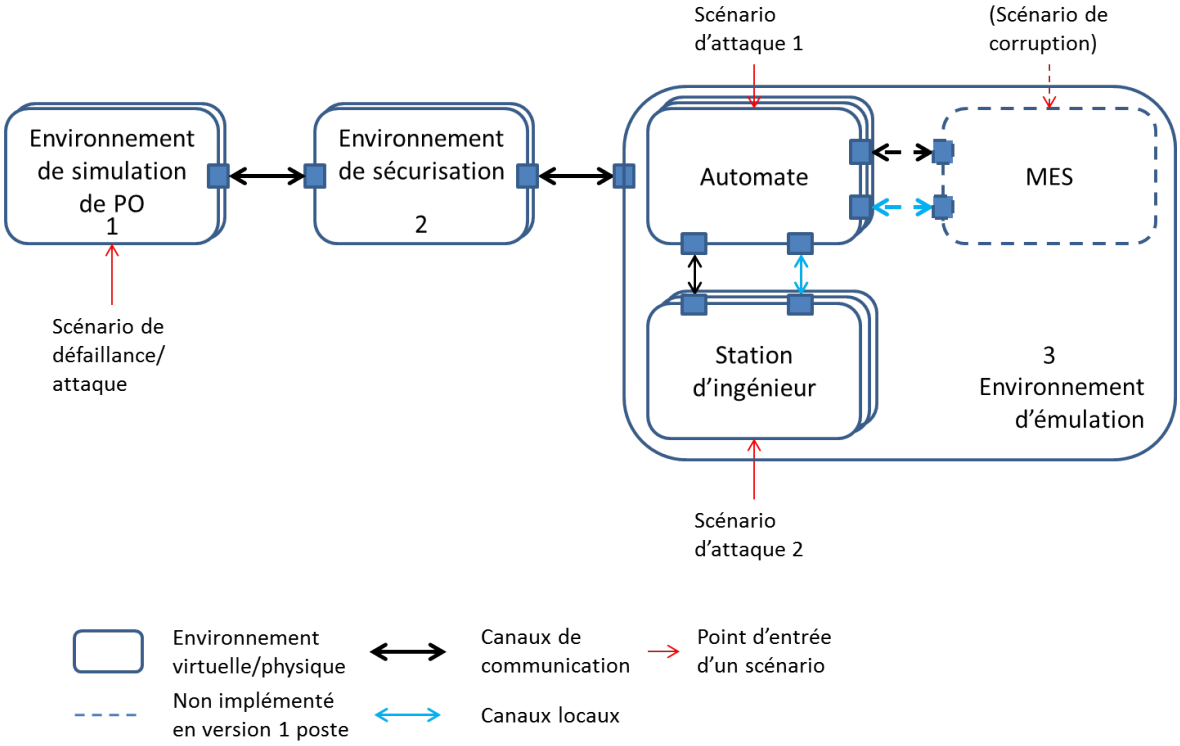


FIGURE 5.2 – Le démonstrateur ASSecIN

Nous allons maintenant présenter les trois environnements qui le composent :

5.1.1 L'environnement d'émulation

Il met en œuvre différents outils qui spécifieront le rôle de la ou des entités :

— un toolkit pour concevoir, exécuter des programmes et dialoguer avec des automates et runtime.

— un MES Qcadoo, qui a été introduit dans les travaux de Romain Bévan avec la reconfiguration dynamique de ligne.

Ces outils sont mis en œuvre dans des machines distinctes ou en une seule.

Qcadoo est l'outil de pilotage du système de production à haut niveau. Cet outil à la charge d'ordonnancer la production par rapport aux ordres de fabrication qu'il reçoit. Aussi il veille au respect de la recette permettant la réalisation du produit. Il représente le niveau 2 du CIM pour notre démonstrateur [Laksmana et al., 2013].

5.1.1.1 Le toolkit

C'est une suite d'outil le workbench Straton proposé par la société Copalp, qui représente le niveau 1 du CIM pour notre démonstrateur, avec le pilotage des équipements de production. L'IDE représente la partie ingénierie et débogue des programmes et automates.
Cette suite d'outils utilisée par les industriels a les mêmes "vulnérabilités" classiques des outils communément exploités.

La station d'ingénieur est la machine mettant en oeuvre l'IDE. Cet outil de conception ainsi que de débogue permet de concevoir les programmes avec des langages normalisés IEC61131-3 (*Sequential Functioning Chart* (SFC), *Functional Block Diagram* (FBD), littérale structuré ...) pour l'automate, ainsi que lier les entrées/sortie du programme à celle du monde physique ou cyber. Cet ensemble d'informations et de programmes est envoyé à l'automate en utilisant l'API et le protocole T5. L'IDE permet aussi un affichage et un contrôle de ses variables par ce même API et protocole. On constate donc deux flots d'information illustrés en figure 5.3.

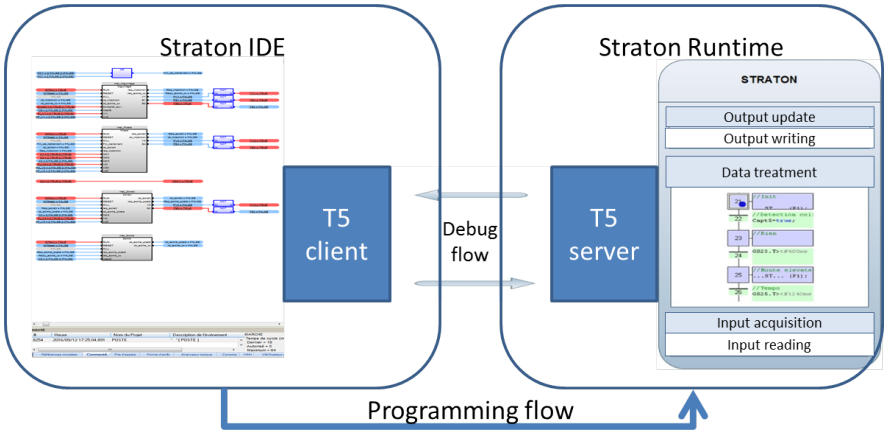


FIGURE 5.3 – Straton "workbench"

Le Runtime est l'outil exécutant le code précédemment conçu et mis en forme. Celui-ci est autonome, il implémente aussi un serveur TCP avec l'API du protocole T5. Plusieurs clients peuvent s'y connecter comme l'IDE, la passerelle ASSecIN, le simulateur, le MES ou un attaquant. Son cycle se décompose en 5 phases :

— la lecture des entrées externe.

— l'acquisition des entrées externe et interne.

— le traitement des informations.

— l'écriture des sorties externe et interne.

— la mise à jour des sorties externe.

La lecture prend en compte les mises à jour des différentes variables inscrites dans des piles de communication. L'acquisition met en exergue ces différentes mises à jour face au programme. Les variables internes sont relatives aux temporisations internes du programme, mais aussi aux variables globales mises à jour par la communication haut niveau (l'API du serveur). Les variables externes sont quant à elle les variables du ou des réseaux de terrain.

Le protocole T5 est le protocole propriétaire des cibles Copalp. Sont API permet la communication avec d'autres entités : l'IDE, le MES, les autres automates .... Ce protocole n'est pas sécurisé, il est donc attaquable une fois qu'on le connait.

5.1.1.2 Rôle

L'environnement d'émulation du niveau 1 et 2 du CIM a donc pour rôle d'être :

— la station d'ingénieur : où l'ont peut forcer des variables par un canal local de debugg.

— l'intelligence de processus : machine se chargeant de manière cyclique de fournir le comportement.

— l'un des hôtes de communication : machine répondant à des requêtes internes ou externes.

5.1.2 L'environnement de sécurité

La passerelle ASSecIn sécurise des IACS. Cet outil logiciel permet de capturer l'information transitant sur un réseau, de l'analyser et détecter des anomalies par des méthodes comportementales ou informationnelles. La détection d'anomalie engendrera des réactions passives ou actives afin de remédier à la situation. Ce dispositif implémente des chaines d'information avec différentes références temporelles. La première étant celle du réseau où la passerelle capture les informations.

5.1.2.1 Les choix de développement induit par le démonstrateur

Le démonstrateur ASSecIN nous a obligés à faire des choix pour la passerelle au cours de son implémentation. Le premier fut l'emploi de l'interface client-serveur TCP/IP, car c'est le protocole de couche basse utilisé. Le second fut l'utilisation du protocole propriétaire de couches hautes T5 de Copalp, qui doit être pris en compte pour l'interprétation de l'information. Nous avons aussi déduit par analyse de différentes captures, les paramètres du filtre présentés en section 3.3.1.1

L'analyse protocolaire nous a permis d'identifier les différentes communications. Nous avons capturé avec l'outil Wireshark [Wir, 1990] la communication entre le simulateur et le Runtime. Ces captures ont ensuite été traitées principalement dans Excel. L'identification de celle-ci nous a fourni plusieurs résultats :

TABLEAU 5.1 – Résultat issus de l'analyse de la capture des communications

ID trames	description	nombre par cycle	taille
T	simulateur commande ajout de temps	1	8
C	simulateur commande l'exécution d'un pas de cycle	1	11
OK	réponse "OK" serveur	1+1+X.1	9/10
RL	requête de lecture d'une information	1	52
RI	mise à jour d'une variable d'information	X	13
WO	écriture des variables d'ordres	1	80

Deux trames nous intéressent : RI & WO. Nous avons donc spécifié leurs tailles dans le filtre. On remarque qu'un certain pourcentage des communications ne nous intéresse pas au point de vue de l'analyse. En effet la/les trames RI, ne sont pas présentes à chaque cycle. Quand elles ne sont pas présentes 5/6 = 83% de la communication est inintéressante. Quand elles sont présentes l'équation : $(5+X)*100/6+2*X$ nous donne le pourcentage de la communication inintéressante selon leur nombre. Nous avons aussi remarqué que le protocole T5 est cyclique et semblable au cycle automate. Cependant les informations ne changent pas forcément. Nous avons donc utilisé le hachage pour identifier les informations identiques. Les résultats de notre étude sont présentés en figure 5.4 pour toutes les communications. Ainsi qu'en annexe 11 avec une décomposition pour les trames qui nous intéressent. Ils mettent en évidence :

— le nombre de trames de même taille avec un même index de hachage.

— trois mesures du temps de réponse en (µs) : maximum, minimum, moyen.

Nous avons filtré un grand pourcentage (+80%) des communications pour l'analyse par la passerelle. Ceci diminue fortement sa charge de travail et limite les analyses non pertinentes.
De ce fait on peut espérer une réduction de la latence induite par le dispositif.

Le développement de la passerelle a été réalisé avec le langage de programmation C. À défaut d'une implémentation matérielle sur réseau de portes programmables, *Field-Programmable Gate Array* (FPGA) de la passerelle, il nous semble pertinent d'aller vers ce langage, qui permet certaines optimisations en temps d'exécution. De plus des outils permettent rapidement de passer d'un code en C à une configuration FPGA comme [viv, 1990]. Cependant nos tests nous ont confortés par rapport à l'objectif de latence.

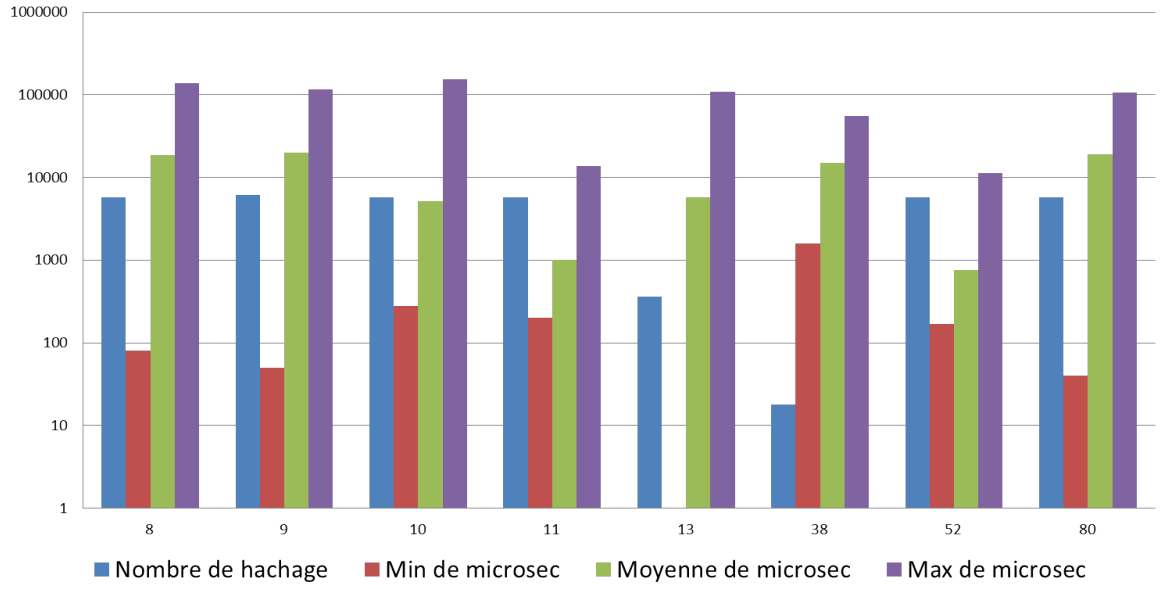


FIGURE 5.4 – Graphique dynamique de l'identification

5.1.2.2 Liens avec le flot précédent

La passerelle ASSecIN doit être configurée avant de pouvoir analyser les communications. Cette configuration prend place dès l'interprétation des communications jusqu'aux méthodes de sécurisation du système. Les références nécessaires sont générées à partir d'un flot de conception précédemment introduit. Nous nous y inscrivons de différentes manières comme présentées en figure 4.11 :

— dans l'étape de conception du système avec l'enrichissement automatique ou manuel 4.4.1.

— dans l'étape de génération avec la génération de la configuration de la passerelle 4.4.2.

— dans le flot de simulation avec l'intégration de la passerelle entre le Runtime et le simulateur.

5.1.2.3 Liens avec le flot précédent

Le démonstrateur ici présenté permet donc à la fois de tester les effets de défaillance et d'attaque sur un système, et de valider les fonctions de détection et réaction de la passerelle.
Le démonstrateur étant modulaire chaque partie peut être remplacées par son homologue réel. Une implémentation de la passerelle a d'ailleurs été réalisée sur une carte SAMA5D3 [ATMEL, 2014] avec un OS spécifique IOT YOCTO [YP, 2011] afin de tester l'outil dans un contexte réel.

5.1.3 L'environnement de simulation

SimSED est un ensemble d'outils permettant la conception et la simulation de la partie opérative d'un système, afin d'en valider le bon fonctionnement. Cet ensemble est présenté en figure 5.5 avec le placement de la passerelle.

Le designer est l'outil permettant la conception d'un site industriel en 3D. Il a son propre DSL intégrant une bibliothèque de composants, afin qu'un utilisateur les instancie avec des paramètres géométriques, des caractéristiques techniques comme la vitesse et l'accélération d'un moteur .... L'outil s'occupe aussi de faire correspondre les ordres des composants avec ceux de l'automate, en chargeant un fichier de configuration fourni par le toolkit Straton. L'instanciation de composants utilitaires comme le générateur de produit ou l'opérateur est faite dans cet outil. Durant nos travaux nous avons conçu notre système sur ComGEM, qui n'a pas le même DSL et que nous avons en plus modifié. Celui-ci génère un modèle pour le designer conforme à son DSL par migration.

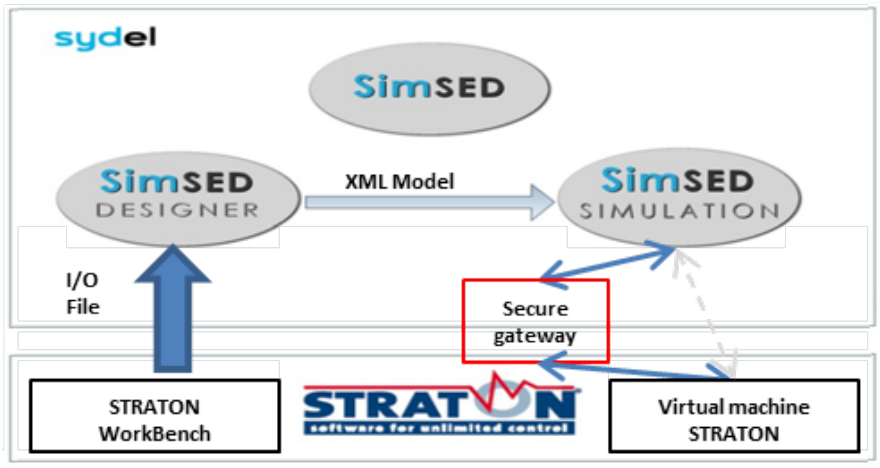


FIGURE 5.5 – Présentation de l'outil SimSED

L'utilisateur n'a donc plus qu'à instancier les composants utilitaires et à faire correspondre les E/S des composants avec celle de Straton. Puis le designer génère un fichier XML interprétable par le simulateur.

Le simulateur permet de simuler le site préalablement réalisé dans le designer. Il récupère l'information du fichier XML ainsi que le fichier contenant le scénario que l'on veut y jouer. Les scénarios permettent notamment d'instancier des aléas (défaillance de composant), en spécifiant la variable impactée et comment, le type d'aléa cyclique ou ponctuel et des paramètres temporels. Nous pouvons donc jouer différents scénarios que notre passerelle devra les détecter. Le simulateur met en œuvre un environnement de réalité virtuelle avec un moteur physique *Open Dynamic Engine* (ODE) [Smith et al., 2005]. Ce dernier s'approche au plus près de la réalité : il gère les collisions, les différents mouvements, ainsi que les effets de force, vitesse et rigidité. La simulation permet le test de programmes sur des cibles virtuelles ou réelles en les connectant en TCP/IP.

5.1.3.1 Rôles

Cet ensemble d'outils a plusieurs rôles :

— La simulation de la partie physique d'un système sous le contrôle d'une autre entité.

— La simulation des effets des menaces.

La finalité est principalement la validation des références de détection réaction. C'est aussi un outil pédagogique permettant de visualiser les effets d'attaque ou de défaillance dans un processus industriel.
Dans la section précédente, nous avons présenté les trois outils de notre démonstrateur. Cependant pour démontrer il faut prouver ou tester, nous avons donc mis au point un cas d'étude à simuler. Ce cas d'étude est basé sur un système réel présent à l'ENSIBS que nous nous proposons de vous présenter.

5.2 Présentation du cas d'étude

Le cas d'étude représente une version simplifiée de l'atelier de production présent à l'ENSIBS.

5.2.1 La partie opérative

La partie opérative de notre cas d'étude est composée d'un poste et un tapis d'alimentation. Il est illustré dans la figure 5.6. Ce poste réalise des traitements en Zone 6 (Z6) une fois le produit intercepté sur le tapis d'alimentation, puis il l'éjecte de nouveau sur le tapis d'alimentation.

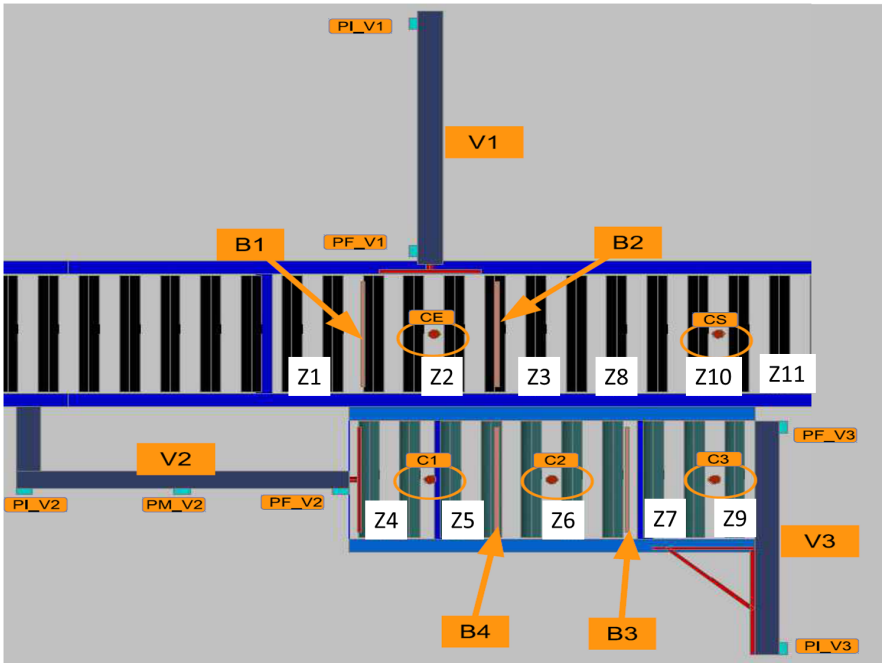


FIGURE 5.6 – Présentation d'un poste

5.2.1.1 Les composants du poste

La partie opérative est décomposée en composants de base. Leur agrégation forme le CBE/EBC puis le CCE/ECC. C'est illustré dans la figure 5.7. On y constate la répartition des différents composants ainsi que la hiérarchie.

- 1 CSys/SysC
- 3 CCE/ECC
  - intercepteur
  - traitement
  - éjecteur
- 3 CBE/EBC
  - vérin d'interception
  - vérin de transfert
  - vérin d'éjection
- 16 CB/BC
  - 3 vérins de base
  - 7 capteurs de position
  - 6 capteurs de présence
- 7 CSu/SuC
  - 3 convoyeurs, dont 2 motorisés
  - 4 butées

Seuls les composants de base et support ont des informations physiques. Les autres classes de la hiérarchie sont une abstraction de ceux de base, donc celles-ci ont leurs propres informations. Cependant elles reçoivent ou donnent des informations et des ordres aux composants de niveau hiérarchique inférieur.

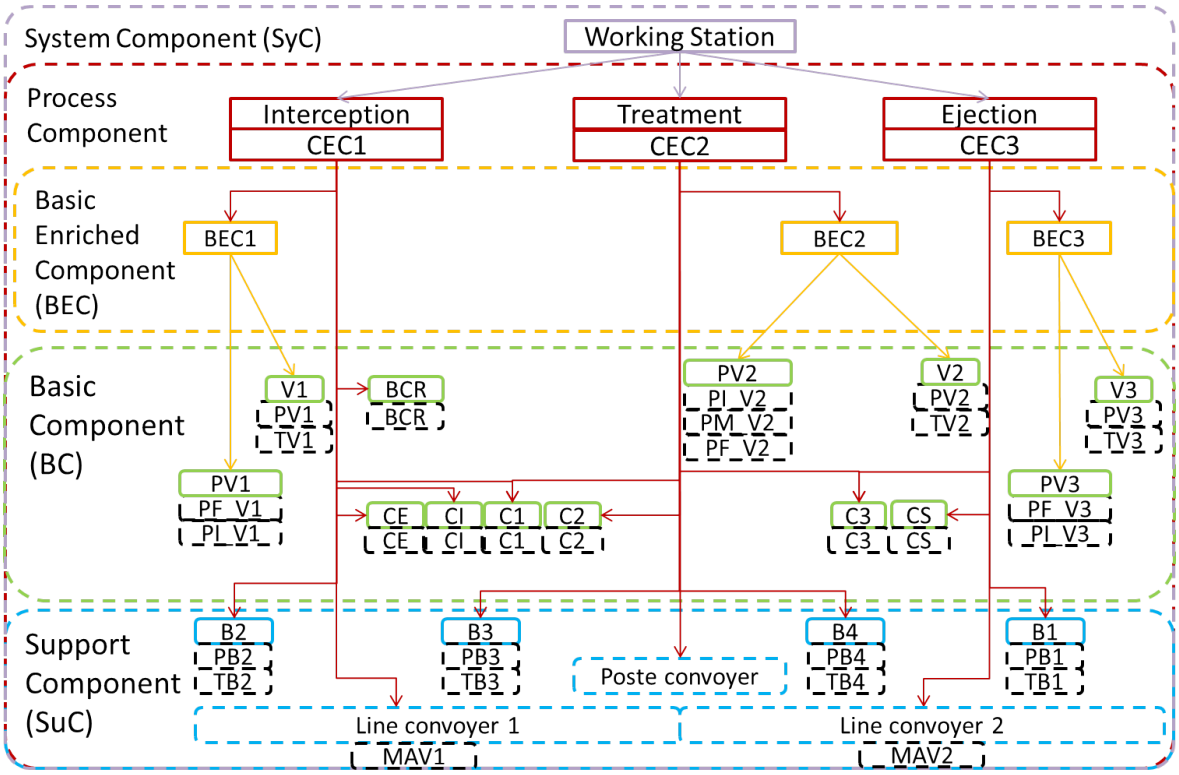


FIGURE 5.7 – Présentation de l'architecture composant du cas d'étude

5.2.1.2 Les composants utilitaires

Ces composants sont instanciés dans SimSED Designer ils servent principalement à la bonne mise en œuvre de la simulation. Ils permettent l'acquisition d'informations supplémentaires pendant la simulation pour vérifier des propriétés du cahier des charges.

— le générateur de produit : c'est le composant paramétrable et/ou commandable générant le produit.

— le chronomètre : c'est un composant permettant la capture précise d'un temps pour un mouvement du produit.

La partie opérative durant la simulation est pilotée avec les variables par une intelligence de procédé réel ou virtuel. Cependant cette intelligence de procédé a son cycle d'exécution commandé par le simulateur, garantissant ainsi le respect du cycle automate simulé.

5.2.2 La partie commande

La partie commande de notre cas d'étude est composée d'un PLC avec la possibilité d'avoir une version réelle ou virtuelle. Un programme s'y exécutant est illustré dans la figure 5.8. Celui-ci a été conçu sans l'outil de génération ComGEM dans un premier temps, puis nous l'avons réalisé avec l'outil. Celui-ci fournit par ComGEM est divisé en une multitude de grafcets représentant le comportement de chacune des opérations. Nous avons ensuite fait le choix de les regrouper par bloc fonctionnel. Il sera donc présenté dans sa globalité avec le regroupement par opération. Le programme a la charge de commander la partie opérative par les IO une fois qu'il est chargé dans une cible. Il respecte la hiérarchie des opérations dans le système et les contraintes de commande posées par le concepteur. On y constate des superpositions de zones exemple : 4 & 5 , 7 & 9, 3 & 8. Ces superpositions sont des interfaces entre CCE/ECC les OB/BO sont les mêmes, mais pas les OC/CO. Exemple : Stockage Actif, *Active Storage* (SA/AS)4 = SA/AS5, Détection de Zone, *Detect Zone* (DZ)4 = DZ5 , Détection de Zone Occupée, *Detect Ocupied Zone* (DZO/DOZ)4 ≠ DZO/DOZ5

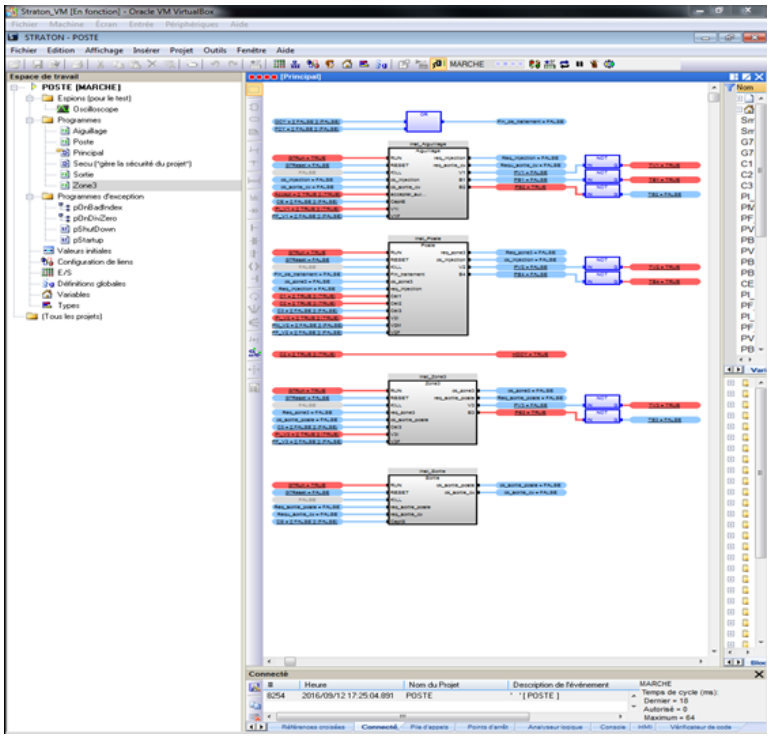


FIGURE 5.8 – Extrait du programme

5.2.2.1 Les opérations du poste

La partie commande est décomposée en niveaux : les opérations basiques puis leur agrégation en l'OC/CO puis l'OCE/ECO, comme illustré dans la figure 5.9.

- 5 Opérations du Système, *System Operations* (OSys/SysO)
  - 2 transferts par convoyeur
  - 3 transferts complexes
- 15 OCE/ECO
  - 4 transferts par vérin
  - 4 stockages actifs par butée
  - 4 détections de zone occupée et 2 partiellement occupées
- 10 OC/CO
  - 7 requêtes de position
  - 3 demandes de position
- 19 OB/BO
  - 3 ordres sortis et 3 ordres rentrés pour les vérins
  - les détections de position : 3 initiales, 3 finales et 1 milieu
  - 6 détections par cellule
- 10 Opérations Supports, *Support Operations* (OSu/SuO)
  - 4 ordres sortis et 4 ordres rentrés pour les butées
  - 2 ordres de marche pour moteur

On y constate que tous les composants ont des opérations. La navigation entre les niveaux hiérarchiques est établie par des variables globales qui sont les flèches sur la figure 5.9. Les informations physiques (ordres/information) sont les derniers & premiers maillons de la chaine. Le haut niveau hiérarchique est le point névralgique où tout converge, ce sont les contraintes de commande.

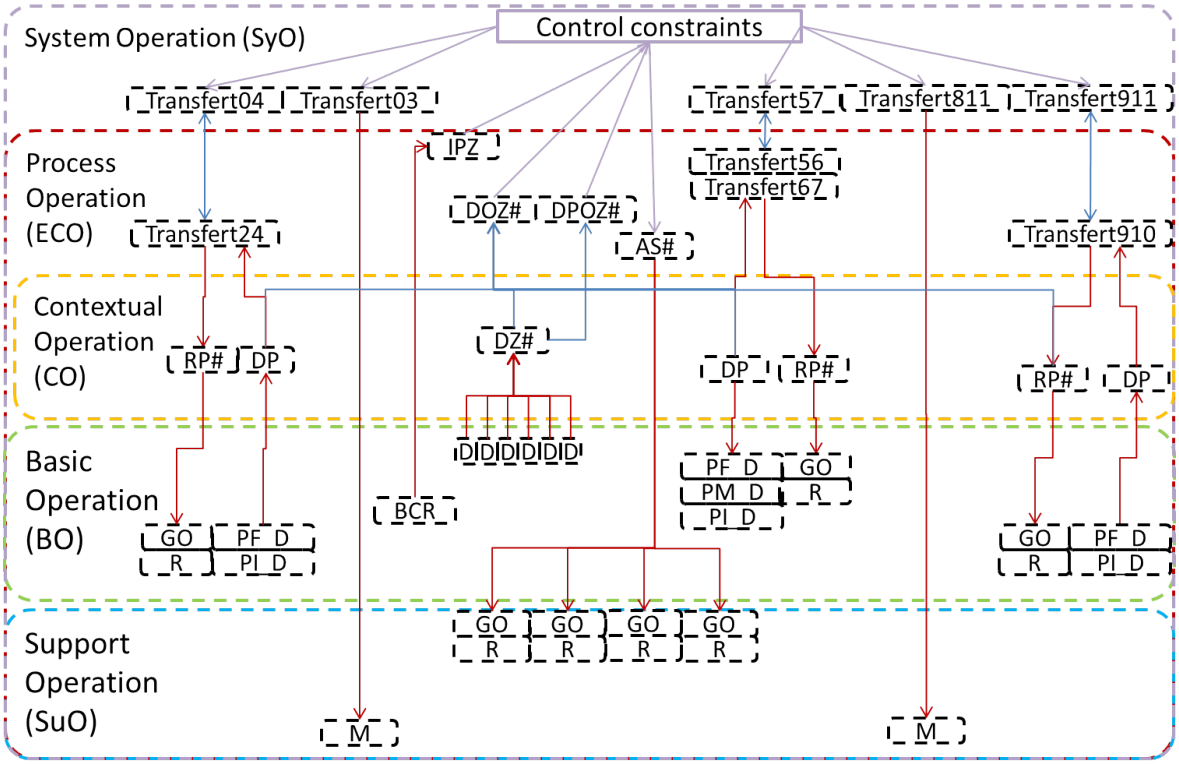


FIGURE 5.9 – Présentation des opérations du cas d'étude

5.2.2.2 Les contraintes de commande

L'ensemble des opérations est régi à haut niveau par ces contraintes. Elles donnent les conditions d'activation et de désactivation des opérations. Les conditions sont les états d'autres opérations ainsi que des temporalités.
Dans notre cas d'étude, l'objectif est de traiter un produit en le sortant de l'anneau d'alimentation, puis l'y replacer sans perturber le flux de produit sur l'anneau.

L'anneau d'alimentation est composé de deux convoyeurs à moteur, deux opérations de transfert commandent les variables qui les pilotent :

— T03 : Commence → 1 ; Termine → 0.

— T811 : Commence → 1 ; Termine → 0.

Les deux opérations s'activent, mais aucune contrainte n'est définie pour leur arrêt.

L'interception et l'éjection se font par l'intermédiaire d'un vérin. Deux opérations contextuelles effectives de transfert commandent des variables pour les OC :

— T24 : Commence → IPZ2 * AS2 * AS4 ; Termine →!DOZ2.

— T910 : Commence → DZ9 * AS2 *!DPOZ2 *!DZ10 ; Termine →!DOZ9.

Les deux opérations s'activent une fois que les conditions DZ pour les zones sources sont validées. Elles se désactivent avec les conditions DZO/DOZ pour les zones sources, en effet la DZO/DOZ est relative à la détection de zone, mais aussi à la position du vérin. Des conditions relatives à la sécurité d'exécution de l'opération sont aussi présentes. En effet les contraintes de commande spécifient les opérations qui commandent, mais aussi l'état des opérations la sécurisant. (Exemple, le verrouillage des zones sources et destination) :

— AS Zone 2 : Commence → DZ2 *!DPOZ2 ; Termine →!T24 + DPOZ2.

— AS Zone 4 : Commence → DZ2 *!DOZ4 ; Termine → DZ4.

— AS Zone 9 : Commence → DZ9 + T56 ; Termine → !T910 *!T56.

Ce verrouillage s'active lors de la DZ ou Détection de Zone Occupée partiellement, *Detect Parcel Ocupied Zone* (DZOP/DPOZ) et se désactive à la fin des opérations de transfert. Le verrouillage de la zone 4 qui s'active quand la zone est inoccupée et se désactive dès que l'on détecte dans la zone. Un dernier verrouillage de zone est présent :

— AS Zone 1 : Commence → DZ2 ; Termine → !T24 + !DOZ2.

Celui-ci n'est actif que pour sécuriser la zone 2 d'interception.

Le transfert vers la zone de traitement est réalisé par un vérin trois positions, qui déplace le produit sur un convoyeur non motorisé. La position milieux est une zone de travail dans laquelle le produit doit rester 1s pour être traité. Les deux mouvements correspondent aux deux opérations de transfert :

— T56 : Commence → (!DZ6 + (DZ7 * DPOZ6 *!AS7) *!AS5 ; Termine → DOZ5.

— T67 : Commence → DPOZ6*!AS5 *!AS7 ; Termine → DZ7.

Ces contraintes fixent donc le comportement de notre système à haut niveau. Les opérations de ce niveau fournissent et prennent l'information du niveau inférieur. ..., Ainsi de suite jusqu'aux ordres et informations de la partie opérative.
Ces ordres et informations transitent et sont traités par la passerelle ASSecIN afin de détecter des anomalies par rapport à des références.

5.2.3 La partie sécurité

Dans la passerelle, le cas d'étude est résumé dans les références mémorisées dans son modèle interne.

5.2.3.1 La référence de sûreté (fonctionnelle)

La référence de sûreté de la passerelle tire parti de la hiérarchie de composant pour sa génération. Nous en donnons une représentation pour le vérin 1 en figure 5.10

— 2 arbres binaires de niveau 1 (CB/CSu),
Vérin1_O_GO & Vérin1_O_R → ordres de jack1

— 2 arbres binaires de niveau 2 (CBE),
Vérin1_O_GO & Vérin1_O_R → informations de CBE1

— 4 arbres binaires de niveau 3 (CCE),
Vérin1_O_GO & Vérin1_O_R → ordres de CCE1
Vérin1_O_GO & Vérin1_O_R → informations de CCE1

— 4 arbres binaires de niveau 4 Csys,
Vérin1_O_GO & Vérin1_O_R → ordres de Csys
Vérin1_O_GO & Vérin1_O_R → informations de Csys

Les arbres binaires ici présentés sont établis par rapport aux variables d'ordres du vérin ou des autres composants. Pour notre cas d'étude nous en avons au maximum : $(3_{vérins})$ * 12 + $(4_{butées})$*10 + $(2_{convoyeurs})$ * 10 = 96 contraintes de sûreté qui peuvent donc être générées pour ce cas. Ce seront les transformations 4.4.1 qui les généreront.

5.2.3.2 La référence opérationnelle

La référence opérationnelle tire parti de la hiérarchie des opérations pour sa génération. Les chroniques sont mises sous forme de graphe à partir des STC décrites dans la section 4.3.2.1. L'état des informations est pris en compte par les fronts, c'est une particularité de nos STC. Nous avons constaté durant nos travaux que la hiérarchie des opérations permet une meilleure mise au point pour les STC

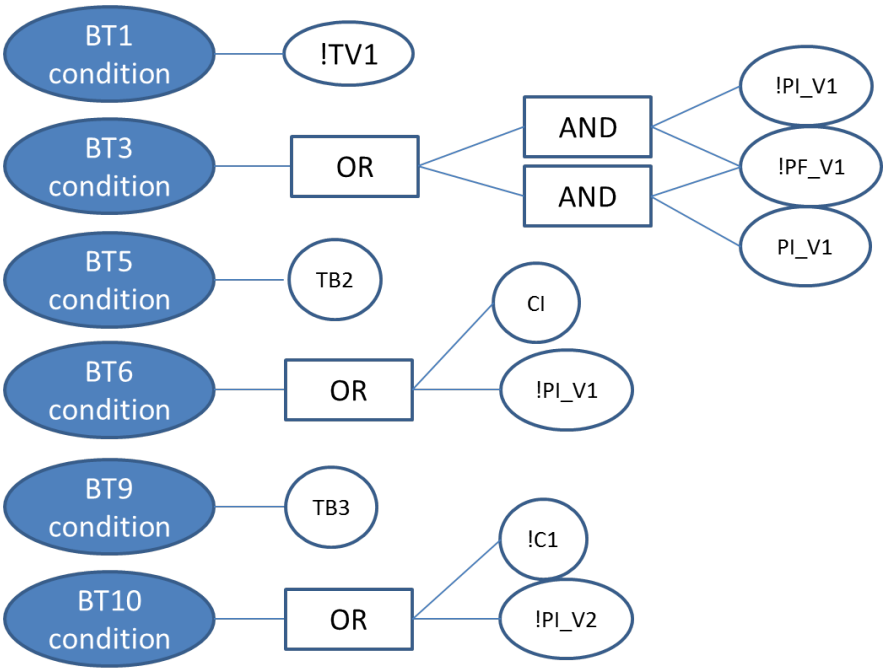


FIGURE 5.10 – Les arbres binaires pour le vérin 1

En effet : une défaillance indéterminée à un niveau hiérarchique peut être déterminée à un niveau supérieur. Nous allons exposer ce point avec l'exemple du vérin :

— STC de niveau (OC),
!(V1_GO & ↓PI_V1 & NCT) * (V1_GO & ↑PF_1 & (LV)) → Défaillance capteur position initiale en sortie (DOPIV1)
(V1_GO & ↓PI_V1 & NCT) *!(V1_GO & ↑PF_1 & (LV)) → Défaillance indéterminée en sortie (ED1)
!(V1_GO & ↓PI_V1 & NCT) *!(V1_GO & ↑PF_1 & (LV)) → Défaillance indéterminée en sortie (ED2)
!(V1_R & ↓PF_V1 & NCT) * (V1_GO & ↑PI_1 & (LV)) → Défaillance capteur position finale en rentrée (DOPFV1)
(V1_R & ↓PF_V1 & NCT) *!(V1_GO & ↑PI_1 & (LV)) → Défaillance indéterminée en rentrée (ED3)
!(V1_R & ↓PF_V1 & NCT) *!(V1_GO & ↑PI_1 & (LV)) → Défaillance indéterminée en rentrée (ED4)

— STC 1 de niveau (OCE),
(B2_GO & ↑CI & NCT) * (CI & ↑V1_GO & 40) * (V1_GO & ↓CI & 40) * (V1_GO & ↑C1 & (LV)) * (V1_GO & ↑ED1 & (LV)) * (C1 & ↓V1_GO & 40) + (C1 & ↑V1_R & 40) → Défaillance capteur position finale en sortie (DOPFV1)

On constate que les STC peuvent fournir des résultats indéterminés à leurs niveaux au point de vue du diagnostic, mais quand elles sont transmises sous forme d'Événement de Défaillance (ED) au niveau supérieur le diagnostic est plus clair.
Cependant nous avons préféré orienter nos travaux vers les méthodes génératives en commençant par les contraintes de sûreté. Afin d'avoir une solution complète pour une méthode de détection et de réaction.
Les STC ne sont générées complètement que jusqu'au niveau contextuel. Les Chroniques ne le sont que sous la forme d'ensembles de sous graphes.
La dernière référence correspond à l'ensemble des variables du système. Elle fait correspondre les noms de variable aux index dans la communication et fait référence aux ensembles de contraintes de sécurité.

5.2.4 Les scénarios de menace

Ce sont des scénarios où le haut niveau du système/l'automate est corrompu. L'intelligence de processus exécute du code malicieux, ou une entité de ce niveau en prend le contrôle, ce qui induit de mauvais comportements. Ces comportements peuvent engendrer tout comme ceux issus de vraies attaques des dégradations d'équipement ou de produit. De plus, ces mauvais comportements peuvent être commandables par des variables hauts niveaux.

5.2.4.1 Le scénario de corruption d'automate

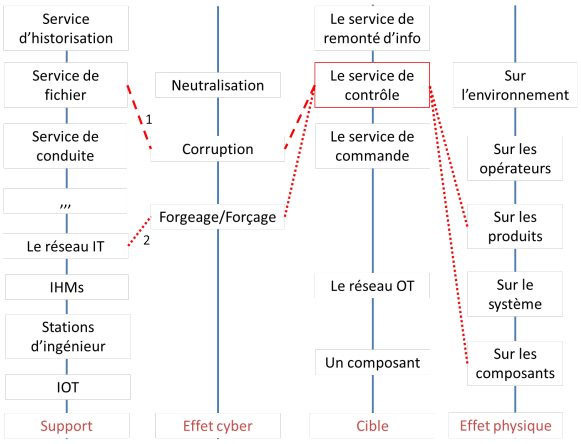
Ce scénario est en trois phases :

— la première phase est la récupération ainsi que la réécriture du programme automate (préliminaire).

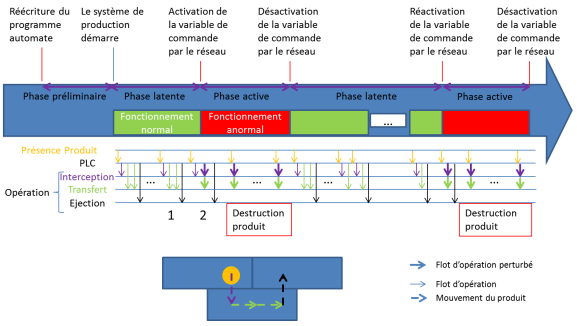
— la deuxième est une phase où le système fonctionne normalement (latente).

— la troisième est une phase, qui est commandée et cause un effet sur le système (active).

Nous illustrons le scénario dans la figure 5.11b. Il est montré le déroulement d'une attaque par corruption avec la phase préliminaire et les phases latente et active, illustré dans la figure 5.11a avec le mapping 2. Ces deux phases alterneront selon le signal de commande de l'attaque. La phase préliminaire s'établit "offline" et est illustrée dans la figure 5.11a avec le mapping 1. Cependant le chargement du nouveau programme corrompu nécessitera le redémarrage de l' PLC.



(a) Mapping de l'attaque par corruption



(b) Description du scénario d'attaque par corruption

Décomposition de l'attaque : Le système fonctionne correctement jusqu'à l'occurrence du déclencheur. Celui-ci est une variable haut niveau commandable par le réseau inter automate. La séquence de fonctionnement normale est illustrée en figure 5.11b avec l'indice 1.

1. la présence du produit est détectée et remontée à l'automate (flèche fine PP → PLC).
2. l'automate commande alors l'interception du produit (flèche fine PLC → intercept^{ion}).
3. une fois celle-ci réalisée l'automate commande une première fois le transfert (flèche fine PLC → transfert).
4. le premier transfert amène le produit en zone de traitement ou un opérateur réalisera une tâche.
5. une fois la tache réalisée, l'automate commande une nouvelle fois le transfert. Il a pour effet de déplacer un autre produit intercepté en zone de traitement, mais aussi d'évacuer le produit traité vers l'éjection.
6. l'automate enfin commande l'éjection du produit traité vers le tapis d'alimentation (flèche fine PLC → éject^{ion}).

Lors de l'attaque en phase active ce fonctionnement est perturbé. La séquence de fonctionnement anormale est illustrée en figure 5.11b avec l'indice 2.

1. la présence du produit est détectée et remontée à l'automate.
2. l'automate commande alors l'interception du produit, mais aussi le transfert (flèche en gras superposée). Si l'un est commandé en sortie alors l'autre le sera aussi, cela est dû à la modification du programme automate.
3. une fois celles-ci réalisées l'automate commande deux fois le transfert, mais aussi l'interception(flèche en gras superposée).
4. l'automate enfin commande l'éjection du produit traité vers le tapis d'alimentation s'il est présent (flèche fine).

La désactivation du déclencheur passe l'attaque en phase latente, elle fait reprendre au système son fonctionnement normal, mais peut donc être déclenchée à plusieurs reprises.
L'effet de l'attaque est montré en figure 5.12 avec les deux vérins sortis en même temps, cela détériore ou détruit le produit en l'éjectant du poste, mais aussi peut nuire à l'opérateur en commandant le vérin pendant que celui-ci réalise la tâche.

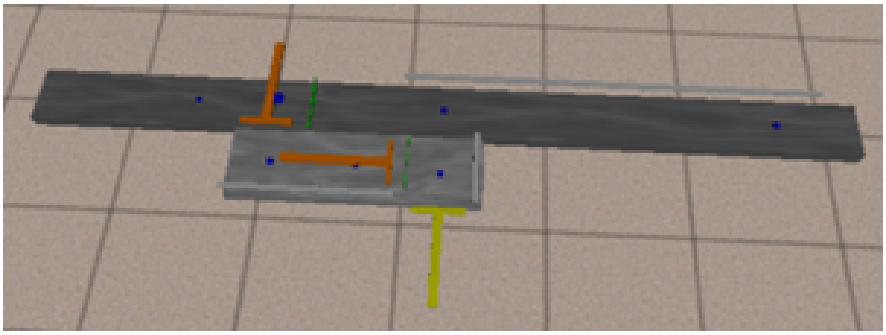


FIGURE 5.12 – Présentation des effets du scénario d'attaque

5.2.4.2 Le scénario de prise de contrôle de l'automate

Ce scénario est en deux phases :

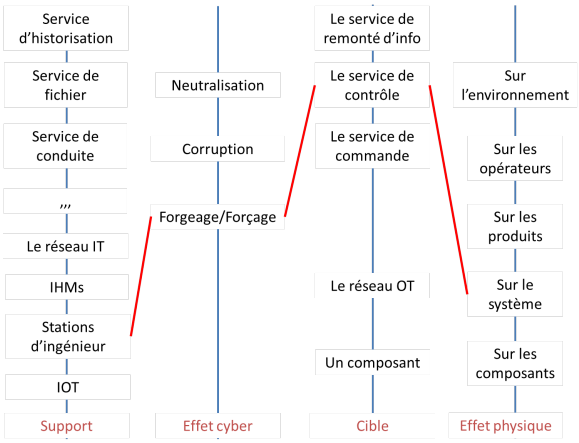
— la première phase vise à prendre le contrôle d'une entité pouvant piloter l'automate (préliminaire).

— la deuxième est une phase, qui est commandée et cause un effet sur le système (active).

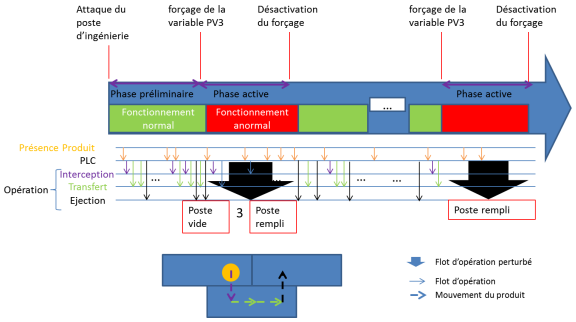
Nous donnons deux illustrations l'une montrant l'attaque sur notre cartographie 5.13a et l'autre donnant le scénario dans la figure 5.13b. Il y est montré le déroulement d'une attaque par "hijacking" avec la phase préliminaire et la phase active. La phase préliminaire est pendant le fonctionnement ou en dehors "offline". La phase active correspond à la durée d'un forçage.

Décomposition de l'attaque : L'attaque est visible sur l'entité dont on a pris le contrôle, mais elle fait fonctionner le système correctement avec la séquence décrite dans la section précédente. Cependant l'attaque s'établit avec une méthode différente. En effet c'est la station d'ingénierie par l'intermédiaire de l'IDE qui y est déployé, qui va forcer des variables dans l'automate et qui induisent un mauvais comportement. La séquence de fonctionnement anormal est illustrée en figure 5.13b avec l'indice 3.

1. la présence du produit est détectée et remontée à l'automate.
2. l'automate commande alors l'interception du produit si la zone n'est pas déjà occupée.
3. une fois celles-ci réalisées l'automate commande le transfert si la zone n'est pas déjà occupée.



(a) Mapping de l'attaque par hijacking



(b) Description du scénario d'attaque

4. l'éjection quant à elle est forcée depuis le départ de l'attaque (flèche large).

L'effet de cette attaque est illustré en figure 5.14. Il peut y avoir destruction de produit selon le moment d'activation, mais le principal effet est le blocage du poste. En effet le poste ne pouvant plus éjecter il intercepte les produits jusqu'à ce qu'il soit rempli puis est neutralisé.

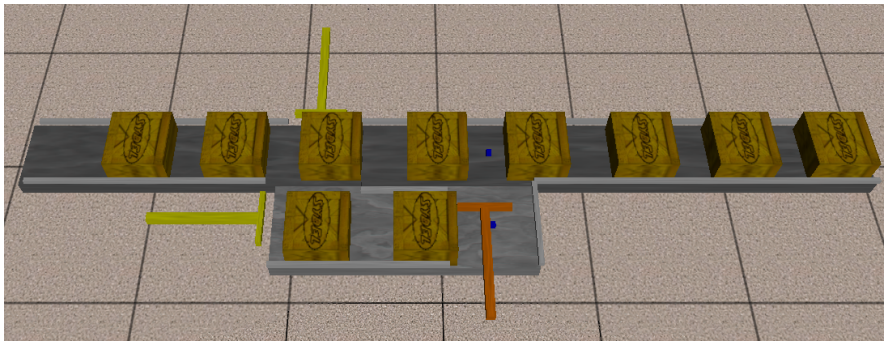


FIGURE 5.14 – Présentation des effets du scénario d'attaque

La méthode d'attaque est différente et ne fera pas appel aux mêmes interfaces cybers. C'est ce que l'on voulait démontrer par ces travaux, en effet il y a des méthodes de détection spécialisées ou généralistes pour les réseaux de la IT. Pour les réseaux de la OT on peut utiliser des méthodes basées sur l'observation des effets qui fournissent une protection plus large.
Les scénarios d'attaques présentés visent à tester les méthodes de détection et de réaction avec des exigences de sûreté. Cependant certaines attaques comme Stuxnet [Falliere et al., 2011] ne violerait pas suffisamment ces exigences pour pouvoir être détectée. Mais les méthodes avec des exigences opérationnelles ou d'intégrités le peuvent.

5.2.5 Autre cas d'étude

Ce cas d'étude a été principalement exploité pour sa complexité et son nombre de composants. Aussi il est composé de 4 postes similaires au premier cas d'étude et d'un tapis central. Il est présenté en figure 5.15, il a servi principalement pour valider la méthode de génération. Dans la section précédente, nous avons présenté un cas d'étude avec le point de vue de chaque outil, ainsi que les références qui configurent la passerelle par rapport à ce cas. Notre démonstrateur a aussi permis la mise en place de scénarios pour tester la configuration et les méthodes de détection et réaction.

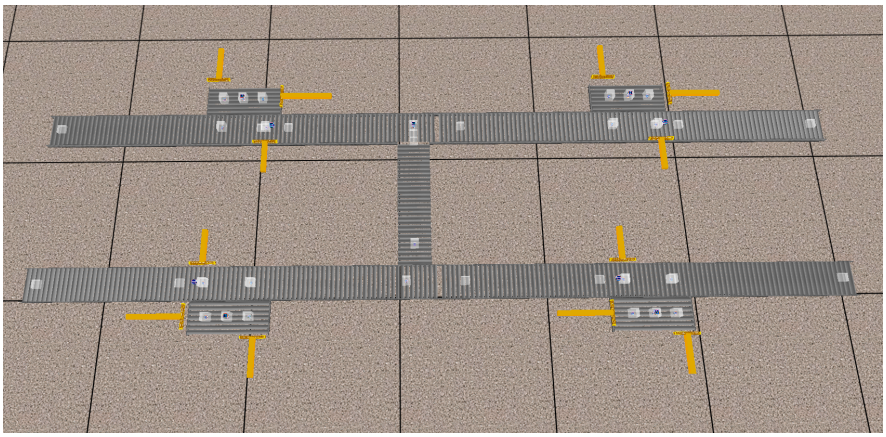


FIGURE 5.15 – Présentation du cas d'étude 4 postes

5.3 Mise en œuvre du cas d'étude

Le cas d'étude une fois implémenté, notre démonstrateur permet de tester le fonctionnement normal du système. Cependant notre objectif est le test de nos configurations pour nos méthodes de détection et de réaction. Nous allons présenter la mise en œuvre du cas d'étude dans chacun des environnements du démonstrateur.

5.3.1 PO dans l'environnement de simulation

La PO est simulée dans une machine virtuelle avec ses ressources cloisonnées des autres. Le simulateur est un client de notre passerelle par le canal de communication dédié. Il simule ainsi l'exécution des ordres qu'il reçoit virtuellement et remonte les informations. Une vision de l'interface est donnée en figure 5.16.

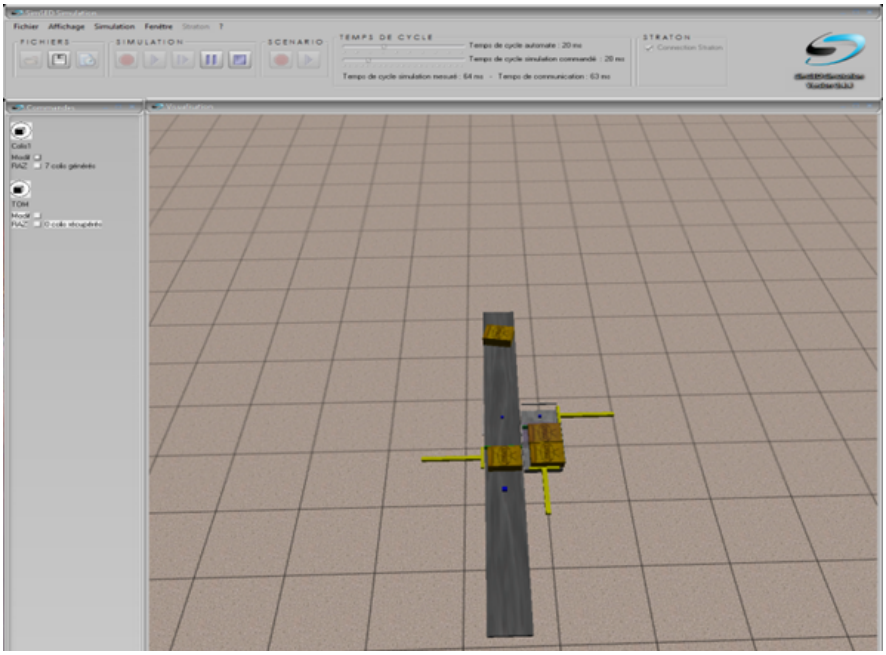


FIGURE 5.16 – Présentation du cas d'étude dans le démonstrateur

De cette interface l'opérateur peut voir les effets d'une attaque sur le niveau 0 du CIM. Il peut aussi commander la génération de produit afin de vérifier le comportement du système selon sa charge. SimSEDs contrôle le cycle automate par l'interface de communication, ce qui induit le respect du cycle automate quel que soit le temps d'exécution du simulateur dans le flot de simulation.

5.3.2 L'intelligence de processus dans l'environnement d'émulation

Cet environnement émule les niveaux 1 et 2 du CIM avec les entités dans une ou plusieurs machines virtuelles. En effet chacune des entités est présente dans l'environnement qui est duplicable. Il est laissé le choix au testeur de choisir l'architecture de ce niveau. Il peut isoler les entités les unes des autres avec des machines virtuelles différentes, ou les laisser toutes en une et utiliser les canaux locaux.

5.3.2.1 L'automate

C'est la première entité, elle émule un automate ainsi que son interface réseau. En effet comme montré en figure 5.17, celui-ci implémente un serveur où l'on peut se connecter en T5.

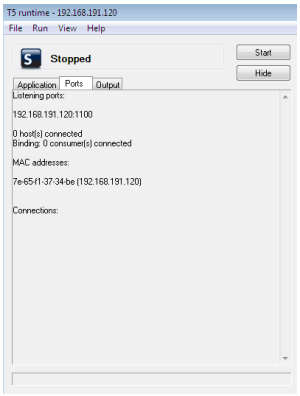


FIGURE 5.17 – Le Runtime

Il exécute le programme de commande, qui lui a été fourni. Il lit les entrées qu'il reçoit et calcule puis envoie les sorties, selon un fonctionnement cyclique.

5.3.2.2 La station d'ingénieur

Cette entité permet principalement de dialoguer avec l'automate pour lui charger des programmes, Elle affiche aussi l'état du programme parallèlement son exécution dans l'automate. ainsi que l'état des variables comme montré en figure 5.18

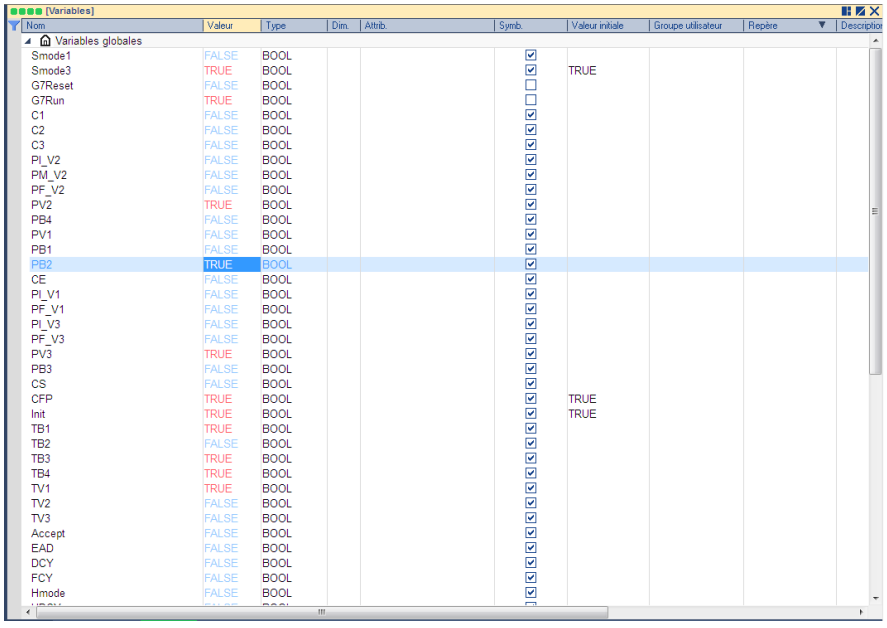


FIGURE 5.18 – L'affichage des variables dans l'IDE

Cette entité se connecte à l'automate en tant que client sur son interface.

5.3.3 Passerelle et résultat

Le socle des travaux présentés dans ce mémoire est la passerelle ASSecIN, dispositif de sécurisation des systèmes industriels à bas niveaux. Ce dernier a été défini comme un moyen de détection et de réaction face à des anomalies. Les méthodes mises en œuvre dans l'outil ont fourni des résultats probants lors des tests dans le démonstrateur.

5.3.3.1 La configuration

Elle est présentée dans la figure 5.19 après interrogation par son interface. Les informations sur

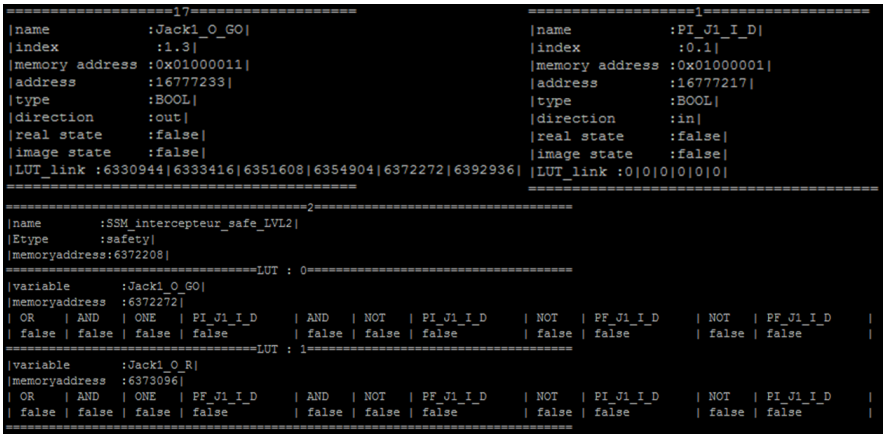


FIGURE 5.19 – Présentation du modèle interne de la passerelle par son interface

les variables sorties vérin 1 et position initiale vérin 1 y sont montrées, notamment l'état réel et sécurisé ainsi que le type. Les arbres binaires de niveau 2 pour le composant enrichi verni 1 sont également présentés.

5.3.3.2 La détection et la réaction

Elles ont été testées avec le scénario d'attaque de l'automate. Il évalue la capacité de la passerelle à détecter les violations des exigences de sûreté et la réaction associée. Une description du flot de l'attaque est donnée en figure 5.20
Ce flot est décomposé en plusieurs séquences selon le mode de la passerelle et sa présence. La séquence avec réaction active est illustrée en figure 5.20 avec l'indice 4. On constate que la passerelle filtre les ordres , qui perturbent le bon fonctionnement selon l'état du système (croix oranges). Ce qui lui permet d'éviter un fonctionnement anormal aux conséquences désastreuses. Elle prévient ainsi la destruction du produit ou l'atteinte de l'opérateur. La passerelle permet ainsi le maintien en fonctionnement de l'IACS par cette filtration Elle alerte aussi le haut niveau du système sur l'état du système qui est attaqué.
La séquence avec réaction passive est illustrée en figure 5.20 avec l'indice 5. La passerelle ne filtre plus, mais informe toujours le haut niveau de l'occurrence d'une attaque. Cependant les effets ne sont pas évités.
Les résultats de la protection sont illustrés en figure 5.21
Cette figure montre une partie de la référence de détection. Puis la remontée d'information de la passerelle avec une vue de la simulation en fonctionnement nominale.
Elle présente ensuite la remontée d'information de la passerelle, qui est constituée des log de détection (la validation des différents arbres binaires), lorsqu'une attaque est détectée avec une vue de la simulation.
Enfin elle montre les différentes réactions que la passerelle déclenche. On y constate que le filtrage d'ordre de la passerelle pour un cas avec/sans produits est différent, mais pas pour l'alerte.

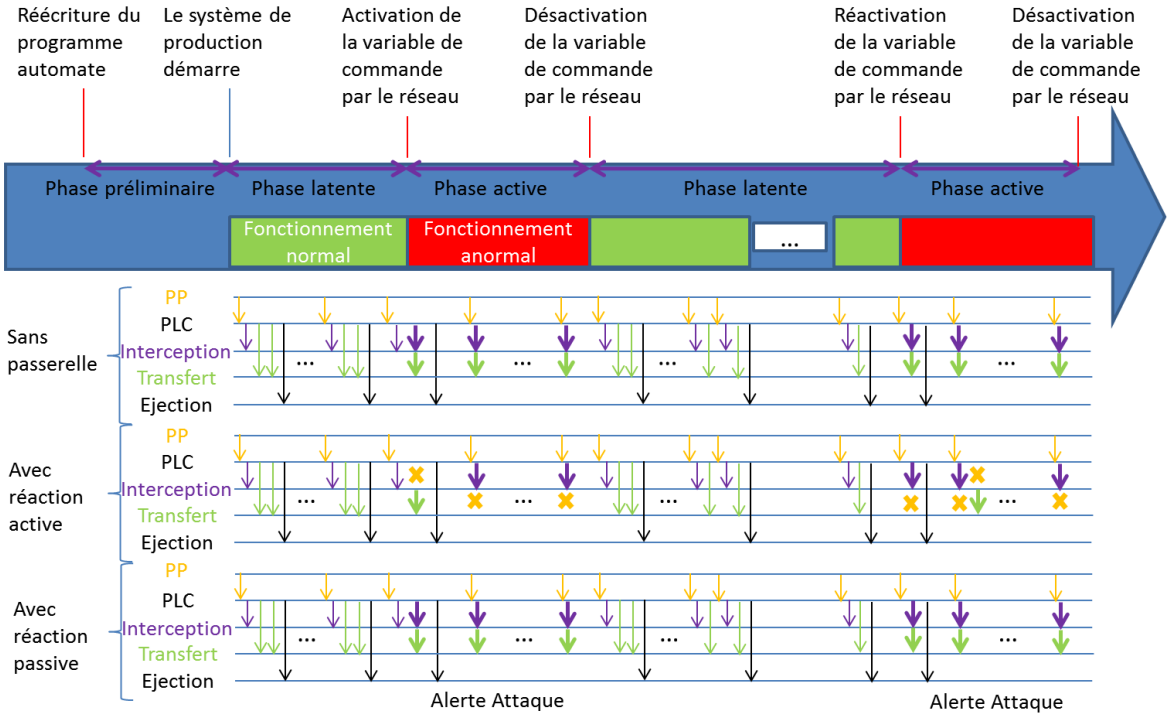


FIGURE 5.20 – Flot du scénario d'attaque corruption de l'automate

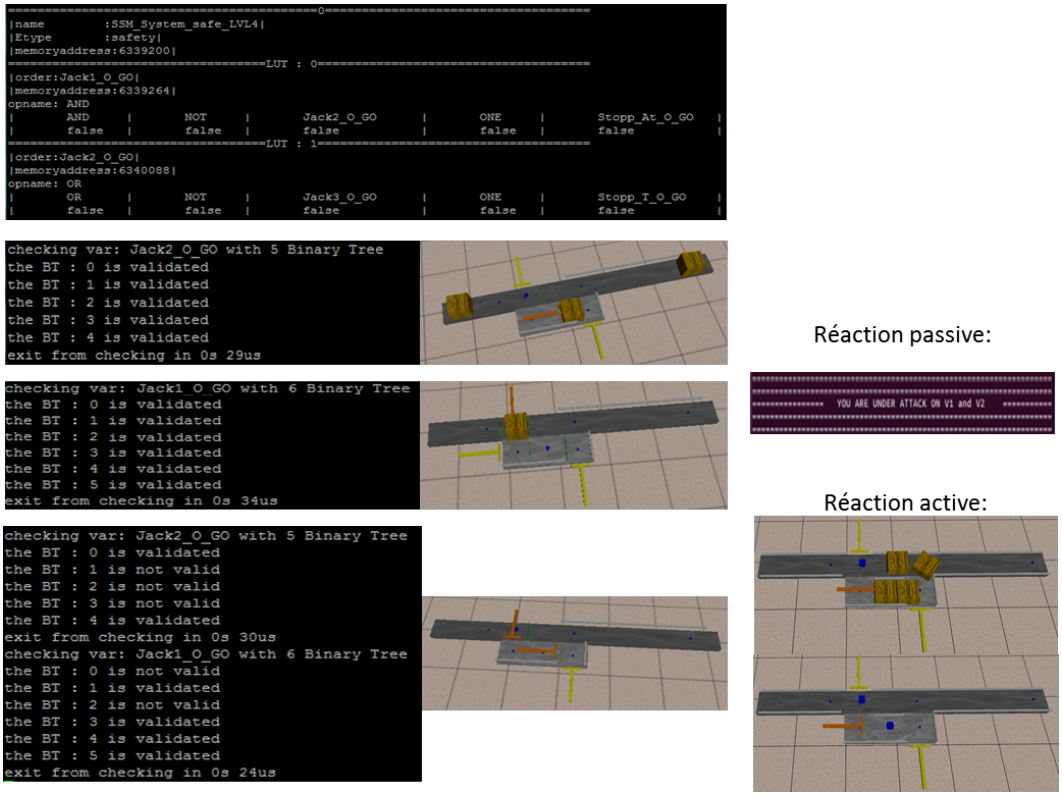


FIGURE 5.21 – Résultat sécurisation face au scénario d'attaque

La détection des attaques pour les scénarios 1 & 2 est bien effective. Les réactions permettent d'informer de manière passive, ou de filtrer afin de garantir la non-destruction ou le non-blocage de produit.

5.3.3.3 La latence

Les IACS ont des exigences de temps de réponse. Nous avons voulu vérifier l'influence de la passerelle sur le temps d'exécution Elle a été évaluée tout au long du développement de l'outil. Nous avons fait cette analyse à chaque étape du développement pour les valider. Les derniers résultats sont retranscrits dans le tableau 5.2. Il compare la mesure de la latence avec et sans la passerelle. Cela nous a permis d'évaluer notre solution de démonstration ainsi que l'impact de notre outil. Aussi une comparaison est réalisée sur notre cas d'étude simulé avec une charge légère de production et une lourde. La charge est relative au nombre de produits à traiter (fréquence d'envois). La lourde implique que certains produits ne seront pas traités du fait du temps d'attente sur le poste.

TABLEAU 5.2 – Résultat pour l'objectif de non-perturbation avec la latence pour métrique

charge du cas d'étude	***légère***	***lourde***	***moyenne***
sans la passerelle	32ms	42ms	37ms
avec la passerelle	37ms	43ms	40ms
latence induite	5ms	1ms	3ms

On y constate une diminution de la latence induite lorsque la charge du cas simulé augmente. Nous n'avons pas pu trouver d'explication à ce phénomène. Cependant nos résultats sont encourageants, car la latence correspond à 10
Nous avons présenté dans cette section la mise en œuvre du cas d'étude. Ainsi que les résultats concernant la passerelle et le démonstrateur. Cependant nos travaux se sont orientés vers la problématique d'intégration de la cybersécurité dans les systèmes.

5.4 La génération

Notre passerelle a besoin d'être configurée pour répondre aux objectifs de détection et réaction. La méthode élaborée permet de générer cette configuration pour un cas par l'IDM. Cette méthode a ensuite été améliorée par la mise en place de l'auto génération. Nous allons synthétiser la mise en œuvre des deux méthodes ainsi que leur bénéfice.

La génération des références est la méthode décrite en section 4.4. Elle a été outillée avec une transformation ATL de 282 lignes permettant la génération automatique de la configuration. Le tableau 5.3 chiffre les résultats de génération.

TABLEAU 5.3 – Résultat chiffré de la génération par l'outil

cas d'étude	***NB de lignes***	***NB d'arbres***
1 poste	**403**	**61**
4 postes	**1500**	**256**

Les références de la configuration générées sont données en annexe :

- 7 pour la référence physique
- pour la référence de détection
  - avec les arbres binaires de niveau 4 (système) 8
  - avec les chroniques de niveau 2 et 1 (transfert9-10)
- 10 pour la référence de réaction

La méthode a été ensuite comparée à une méthode entièrement manuelle issue de nos premiers essais. Le tableau 5.4 fait un bilan des temps de génération.

TABLEAU 5.4 – Comparaison entre une génération manuelle des références et une assistée

méthode	***définition des contraintes***	***traduction***	***total***
manuelle	**3**h	**1**h	**4**h
outillée	**2**h	**1**min	**2**h

Cependant, cette comparaison ne tient pas compte du temps de conception dans l'outil qui peut être très réduit. En effet dans les précédents travaux une bibliothèque de composants était réalisée. La possibilité d'agrégation de composants pour en former de nouveaux était possible avec la mémorisation de leurs caractéristiques. Cela permet donc par réutilisation de grandement diminuer le temps de définition des contraintes.
Nous avons réfléchi à une autre façon de diminuer, mais aussi simplifier la génération de contraintes. Nous allons vous présenter les résultats du complément à cette méthode.

L'auto génération de la vue surveillance est le complément à la méthode précédente. Des transformations élaborent automatiquement les contraintes par raffinement successif. Elles exploitent les vues graphiques et matérielles des composants du modèle. Elle les instancie ensuite dans le modèle adéquat "AutoSafety". Un exemple de l'auto génération est donnée en annexe 9. Notamment le volume de codage des transformations d'enrichissement et le nombre de règles conçues et mises en œuvre. Il y est aussi donné le volume d'arbre binaire (BT) généré par niveau et selon le cas d'étude avec le nombre de nœud (N) et de feuille (F).
Le tableau 5.5 chiffre les résultats de l'auto génération.

TABLEAU 5.5 – Valeur ajoutée des transformations d'instanciation automatique des arbres binaires

transformations	***taille du code***	***NB de règles***	**ajout pour 1 poste**	**ajout pour 4 postes**
raffinement CB	**253 lignes**	**4**	**18BT 1F**	**76 BT 1F**
raffinement EBC	**504 lignes**	**2**	**4BT 3N 4F, 2BT 8N 9F, 4BT 1F, 2BT 1N 2F**	**16BT 3N 4F, 8BT 8N 9F, 16BT 1F, 8BT 1N 2F**
raffinement ECC	**1500 lignes**	**9**	**10BT 1N 2F, 15BT 1F**	**40BT 1N 2F, 60BT 1F**
raffinement CSys	**254 lignes**	**3**	**2BT 3N 4F, 4BT 1N 2F**	**8BT 3N 4F, 24BT 1N 2F**
totaux	**2511 lignes**	**18**	**61BT 50N 111F**	**256BT 208N 464F**

Nous avons ensuite comparé les temps de réalisation avec la méthode manuelle et la méthode optimisée par raffinement dans le tableau 5.6.

TABLEAU 5.6 – Comparaison entre une génération manuelle des références et une avec auto génération

méthode et cas	***définition d'un site***	***définition des contraintes***	***traduction***	***total***
manuelle 1 poste	**1**h	**2**h	**1**h	**4**h
outillée 1 poste	**1**h	**10**min	**1**min	**1**h**.10**min
manuelle 4 postes	**4**h	**8**h	**4**h	**16**h
outillée 4 postes	**2**h	**40**min	**1**min	**2**h**.10**min

On constate que le temps de génération de la configuration pour 1 poste est quasiment divisé par 4. Le temps de définition des contraintes n'est pas totalement réduit, car le concepteur doit valider les contraintes générées.
Nous avons aussi pu constater que bien que le temps de définition augmente avec le nombre de contraintes, ce n'est en rien comparable avec l'augmentation induite par la méthode manuelle lors de la définition de cas complexe avec 4 postes. En effet le temps de génération est alors divisé par 8 grâces notamment à la réutilisation des composants.

5.5 Conclusion

Dans ce chapitre nous avons présenté un démonstrateur permettant de vérifier nos méthodes de détection et de réaction ainsi que leurs configurations. Nous avons introduit notre cas d'étude selon une démarche outillé en donnant les différentes implémentations dans les outils : La PO avec sa représentation générée par l'outil ComGEM pour le simulateur; La Partie Commande, *command part* (PC) avec ses opérations et les contraintes de commande aussi générée par ComGEM pour l'automate; La partie sécurité avec les références de sécurité générée par notre outil ComSecGeM pour la passerelle. Enfin des scénarios d'attaque nous ont permis de valider la sécurité offerte par notre approche. La sécurité étant réactive et passive selon l'effet induit par le scénario et réalisé en ligne, elle est donc assurée face à des anomalies d'origine externe ou interne.

6 Conclusion générale

« *La logique des passions renverse l'ordre traditionnel du raisonnement et place la conclusion avant les prémisses.* »

Albert Camus

Sommaire

Les travaux présentés dans cette thèse ont abouti à différentes contributions, pour répondre à la problématique de la sécurisation des SAP, notamment dans le cadre de la cybersécurité, appliqué aux systèmes transitique.
Après avoir introduit la quatrième révolution "numérique" que subissent nos systèmes de production, nous avons dressé un état de l'art sur ce sujet avec une vision sur la sécurité de ces systèmes. En nous focalisant sur les effets, nous avons pu regrouper attaque et défaillance sous la notion d'anomalie. Ce positionnement au niveau de l'effet d'une attaque cyber est intéressant, car il réduit l'éventail des possibilités à explorer afin d'empêcher la finalité de cette dernière. En effet, les techniques d'attaques évoluent sans cesse alors que la finalité (destruction ou détérioration d'une partie du système ou du produit) reste dénombrable. Nos travaux se positionnent donc en complément des travaux sur la détection d'intrusion et constituent le rempart ultime de protection du système physique.
Nous avons ensuite présenté nos contributions en relation avec les objectifs que nous avons posés.

6.1 Rappel des contributions

Les trois verrous que l'on a adressés sont :

— Comment sécuriser l'IACS d'un SAP ?

— Comment faciliter l'intégration de la cybersécurité ?

— Comment vérifier l'efficacité de la solution et de nos apports ?

Nous avons posé les objectifs de nos travaux en réponse à ces verrous :

— en concevant une solution pour détecter et réagir à des anomalies.
Un outil de sécurisation configurable pour les IACS a été développé. Il s'agit de la passerelle ASSecIN avec un placement innovant sur le réseau de terrain.

— en élaborant une méthode pour faciliter l'intégration de la cybersécurité dans les systèmes.
Cet objectif a été atteint par la proposition de deux méthodes génératives, s'inscrivant dans un flot de conception pour les systèmes de production. La première méthode s'établit par instanciation manuelle des contraintes avec un concepteur [Toublanc et al., 2018], alors que la deuxième auto-génère les contraintes par raffinement. Ces deux méthodes ont été implémentées en utilisant les concepts de l'IDM.

— en mettant en place un environnement démontrant le fonctionnement et vérifiant les caractéristiques de notre outil de sécurisation.
Ce dernier objectif a été atteint par l'élaboration d'un démonstrateur. La pertinence et l'efficacité de notre solution ont été confrontées à un cas d'étude concret, ainsi qu'à des scénarios d'anomalie [Toublanc et al., 2017].

Ces objectifs ont été atteints tant sur les aspects méthode que réalisation. Nous proposons un dispositif flexible pouvant sécuriser un système de production, ayant la capacité de détecter et réagir aux effets d'anomalies sur celui-ci ainsi qu'une méthode générant la configuration du dispositif. Nos travaux ouvrent ensuite différentes perspectives.

6.2 Perspectives

Les résultats obtenus dans nos travaux sont intéressants en matière de sécurité ainsi qu'en terme de méthode. Ils ouvrent des perspectives tout aussi intéressantes notamment sur de nouvelles méthodes de sécurisation, ou encore sur des aspects de modélisation de la sécurité et de l'environnement d'un SAP.

Point de vue du dispositif de sécurisation : permettre de sécuriser le fonctionnement d'un système de production en implémentant des méthodes de détection et de réaction face à des anomalies. Nous souhaitons continuer dans cet axe avec de nouvelles méthodes de détection relatives :

— à un historique d'information qui permettrait l'analyse du cycle de vie d'un composant par analyse statistique.

— à la violation de permission, avec le point de vue de la communication d'un équipement communicant sans permission avec un autre.

— au trafic réseau afin de remonter les métriques relatant la charge réseau pour ainsi identifier des attaques DDOS.

Nous pourrions aussi poursuivre les travaux sur de nouvelles méthodes de réaction relative :

— à un index de confiance pour l'équipement et ses variables.

— au "firewalling" permettant de protéger l'équipement face à des communications violant des permissions ou ayant un index de confiance bas.

— à l'isolation pour sécuriser les équipements autonomes (robot) ayant leurs propres intelligences de processus, ainsi que les systèmes massivement distribués, composés d'équipement issu de l'IOT.

Cela exigera un effort d'implémentation pour le code de la passerelle, ainsi que des travaux dans l'implémentation matérielle de certaines méthode ou partie de la passerelle [Lesjak et al., 2015] ou [Yin et al., 2015] compte tenu de l'objectif de non-perturbation de la passerelle par rapport au fonctionnement du système. Enfin un travail sur un moteur décisionnel synthétisant l'ensemble des détections qui ne couvrent pas des exigences de sûreté semble une piste prometteuse. En effet la mise en place d'une intelligence décisionnelle alliant la fusion d'information et les réseaux de neurones est une piste pouvant être explorée. Une autre piste intéressante est la distributivité de la sécurité avec un réseau de passerelles et une intelligence décisionnelle distribuée.

Point de vue de la méthode de configuration l'un des objectif serai la simplification du paramétrage de l'outil de sécurisation. La suite logique de nos travaux serait de continuer l'implémentation du flot afin de générer de nouvelles configurations pour de nouvelles méthodes. Pour cela il faut modéliser les ressources nécessaires à la nouvelle méthode de sécurisation. Sans être exhaustif, les travaux sur les méthodes de détection pourraient traiter de :

— l'analyse de l'information (un historique, des paramètres, un index de confiance [Franklin et al., 2014])

— la vérification des permissions (liste des permissions, un index de confiance, une probabilité d'attaque [Papp et al., 2015])

— l'analyse du réseau avec les méthodes de [Bhuyan et al., 2014] (des paramètres statistiques, une probabilité d'attaque)

Les travaux sur les méthodes de réaction pourraient s'intéresser à :

— un listing de confiance multi niveaux (les variables, les entités)

— le "firewalling" (les pare-feu matériels [Cotret et al., 2012], les pare-feu intelligents [Zou et al., 2002])

— l'isolation (une probabilité d'attaque)

Ce travail de modélisation des références de la passerelle doit être réalisé pour les nouvelles méthodes. Enfin des travaux sur un mécanisme "plug and play" par apprentissage pour l'auto configuration de l'outil serait intéressant.

Point de vue de la modélisation nous simplifierions la configuration de l'outil de sécurisation, Les travaux se sont focalisés sur une approche centralisée (une partie opérative pour un automate) Nous souhaitons par la suite nous orienter vers les architectures distribuées. Pour cela il sera nécessaire de modéliser la communication entre composants (l'architecture, les protocoles, les permissions) Il sera aussi nécessaire de modéliser la communication avec les autres entités du système de production :

— MES
— Gestionnaire de Maintenance Assisté par Ordinateur (GMAO)
— Gestionnaire de Production Assisté par Ordinateur (GPAO)
— SCADA
— serveur constructeur

Cela exigera un enrichissement du langage actuellement utilisé dans nos travaux, ainsi que des travaux de transformation de modèles notamment pour le raffinement automatique, mais aussi la génération des configurations pour les différentes passerelles. La finalité serait d'introduire la notion de mode pour le système afin d'y adapter les réactions de la passerelle en conséquence.
Enfin des travaux seront à articuler autour de la norme IEC-62264, qui est spécialement conçue pour la modélisation des systèmes de production, ou encore l'IEC61499 conçue pour les applications distribuées. Afin de générer un nouveau langage généraliste, qui permettrait la mise en place de nombreuse transformation pour générer des configurations de commande, de sécurité, ou même de jumeau numériques.

Point de vue du démonstrateur l'un des objectifs serait la virtualisation totale d'un site de production. Elle permettrait à la fois un entrainement pédagogique pour des opérateurs face à différentes anomalies, ainsi qu'un outil de test multi niveaux pour des solutions de cybersécurité. Les recherches sur la charge mentale ou les systèmes sociotechniques y trouveraient également un outil sur lequel s'appuyer pour leurs théories.

Bibliographie

Jean-Louis Lallican. *Proposition d'une approche composant pour la conception de la commande des systèmes transitiques*. PhD thesis, Université de Bretagne Sud, 2007.

Romain Bévan. *Approche composant pour la commande multi-version des système transitiques reconfigurables*. PhD thesis, Université de Bretagne Sud, 2013.

Edward A Lee. Cyber-physical systems-are computing foundations adequate. In *Position Paper for NSF Workshop On Cyber-Physical Systems : Research Motivation, Techniques and Roadmap*, volume 2, pages 1–9. Citeseer, 2006.

Joseph Harrington. *Computer integrated manufacturing*. RE Krieger Publishing Company, 1979.

Dennis L Brandl. Isa sp 95 : The factory-to-business link. *IND COMPUT*, 19(6) :16–17, 2000.

Éric Zamaï, Fabien Rigaud, Jean-François Pétin, Pascal Berruet, and Armand Toguyeni. Architectures de pilotage de procédés industriels. In *Techniques de l'Ingénieur. Génie industriel - Conception et Production - Conduite des systèmes industriels*, page AG3510. Editions Techniques de l'Ingénieur, July 2007. URL https://hal.archives-ouvertes.fr/hal-00156363.

Philippe Allot. SCADA et MES : les vérités qui dérangent. , L'usine nouvelle, 2014.

Jean-Sébastien Mouchard. *Proposition d'une approche méthodique pour la conception des systèmes automatisés de production : application aux systèmes transitiques*. PhD thesis, Universitée de Bretagne Sud, 2002. URL http://www.theses.fr/2002LORIS014.

Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

Florent Frizon De Lamotte. *Proposition d'une approche haut niveau pour la conception, l'analyse et l'implantation des systèmes reconfigurables*. PhD thesis, Université de Bretagne Sud, 2006.

ANSSI. *Guide de l'hygiene informatique*, https://www.ssi.gouv.fr/uploads/2017/01/guide_hygiene_informatique_anssi.pdf, , ANSSI, 2013.

Code pénal article 323-2. https://www.legifrance.gouv.fr/affichCodeArticle.do;jsessionid=C6636342B4926963DBF3E455E93672C8.tplgfr40s_1?idArticle=LEGIARTI000006418319&cidTexte=LEGITEXT000006070719&dateTexte=20120126, 2004.

Dorottya Papp, Zhendong Ma, and Levente Buttyan. Embedded systems security : Threats, vulnerabilities, and attack taxonomy. In *Privacy, Security and Trust (PST), 2015 13th Annual Conference on*, pages 145–152. IEEE, 2015.

Nicolas Falliere, Liam O. Murchu, and Eric Chien. W32. stuxnet dossier. , Symantec Corp, 2011.

Tim Conway Robert M. Lee, Michael J. Assante. German steel mill cyber attack. , SANS Industrial Control Systems, 2017.

CERT. Wannacry ransomware campaign exploiting smb vulnerability. , Computer Emergency Response Team, 2017.

CLUSIF. Site web du club de la sécurité de l'information. https://clusif.fr, 1992.

Habib Hadj Mabrouk. Introduction à la sécurité et à l'analyse des risques technologiques et humains. In *3ème Symposium International sur la Maintenance et la Maîtrise des Risques*, page 16p, 2010.

Jean-Claude Laprie, Jean Arlat, JP Blanquart, A Costes, Y Crouzet, Y Deswarte, JC Fabre, H Guillermain, M Kaâniche, K Kanoun, et al. Guide de la sûreté de fonctionnement. *Cépaduès éditions*, 1995.

Edmond M. Clarke and Jeannette M. Wing. Formal Methods : State of the Art and Future Directions. *ACM Computing Surveys*, 28(4) :626–643, 1995. URL https://hal.archives-ouvertes.fr/hal-00444076.

Soraya Mesli Kesraoui. *Integration of formal verification techniques into a control-command system design approach : application to SCADA architectures*. These, Université de Bretagne Sud, May 2017. URL https://tel.archives-ouvertes.fr/tel-01738049.

Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*, volume 2. McGraw-Hill New York, 1991.

Sophie Prat. *Integration of simulation-based checking into an automated design approach of control-monitoring system*. These, Université de Bretagne Sud, December 2017. URL https://tel.archives-ouvertes.fr/tel-01691344.

Olivier Cardin. *Contribution of online simulation to production activity control decision support – Application to a flexible manufacturing system*. These, Université de Nantes, October 2007. URL https://tel.archives-ouvertes.fr/tel-00338761.

Modèle OSI. https://www.frameip.com/osi/, 2017.

Modèle TCP et IP. https://www.frameip.com/tcpip/#1-8211-introduction-au-modele-tcpip, 2017.

ISA/IEC 62443-3-1. Industrial communication network - network and system security : Security technologies for industrial automation and control systems. , International Society of Automation, 2009.

Keith Stouffer and Joe Falco. *Guide to supervisory control and data acquisition (SCADA) and industrial control systems security*. National institute of standards and technology, 2006.

ANSSI-GT-SI. Méthode de classification et principales mesures. , ANSSI Groupe de Travail Sécurité Industrielle, 2014.

EBIOS. Expression des Besoins et Identification des Objectifs de Sécurité. https://www.ssi.gouv.fr/uploads/2011/10/EBIOS-1-GuideMethodologique-2010-01-25.pdf , ANSSI, 2010.

Rotronic. Site web marchand. https://www.touslescables.com/cle-verrou-prise-rj45-A8AL-1092.html, 2014.

WALLIX. Wallix solution de bastion numérique. https://www.wallix.com/wp-content/uploads/2018/02/WALLIX_BASTION_FR_2018.pdf, 2018.

Sentryo. Site WEB. https://www.sentryo.net/fr/, 2000. Solution for cyber security for industrial Internet.

Stormshield. *Guide pour création d'un réseau local virtuel*, 2019.

Réseau privé virtuel VPN. https://www.frameip.com/vpn/, 2017.

Tatu Ylonen and Chris Lonvick. The secure shell (SSH) protocol architecture. , RFC 4251, 2005.

Protocole SSL et TLS. https://www.frameip.com/ssl-tls/#1-8211generalites-du-protocole-ssl-et-tls, 2017.

Tobias Heer and Samu Varjonen. Host identity protocol certificates. , RFC 8002, 2016.

Ronald L Rivest et al. The md5 message-digest algorithm. , RFC 1321, 1992.

Young H Cho and William H Mangione-Smith. Deep packet filter with dedicated logic and read only memories. In *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, pages 125–134. IEEE, 2004. doi : 10.1109/FCCM.2004.25.

Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, and John Lockwood. Deep packet inspection using parallel bloom filters. In *High performance interconnects, 2003. proceedings. 11th symposium on*, pages 44–51. IEEE, 2003.

Fang Yu, Zhifeng Chen, Yanlei Diao, TV Lakshman, and Randy H Katz. Fast and memory-efficient regular expression matching for deep packet inspection. In *Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, pages 93–102. ACM, 2006.

Daniel Smallwood and Andrew Vance. Intrusion analysis with deep packet inspection : increasing efficiency of packet based investigations. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 342–347. IEEE, 2011. doi : 10.1109/CSC.2011.6138545.

Sandeep Bhatt, Pratyusa K Manadhata, and Loai Zomlot. The operational role of security information and event management systems. *IEEE security & Privacy*, 12(5) :35–41, 2014.

Manuel Cheminod, Luca Durante, and Adriano Valenzano. Review of security issues in industrial network. *IEEE TII*, pages 277–293, February 2013.

Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8) :805 – 822, 1999. ISSN 1389-1286. doi : https://doi.org/10.1016/S1389-1286(98)00017-6. URL http://www.sciencedirect.com/science/article/pii/S1389128698000176.

J. P. ANDERSON. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980. URL https://ci.nii.ac.jp/naid/10014688737/en/.

D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13 (2) :222–232, Feb 1987. ISSN 0098-5589. doi : 10.1109/TSE.1987.232894.

T. F. Lunt, R. Jagannathan, R. Lee, A. Whitehurst, and S. Listgarten. Knowledge-based intrusion detection. In *[1989] Proceedings. The Annual AI Systems in Government Conference*, pages 102–107, March 1989. doi : 10.1109/AISIG.1989.47311.

Lee Wilmoth Lerner. *Trustworthy embedded computing for cyber-physical control*. PhD thesis, Virginia Polytechnic institute and state university, 2015.

Sandeep KumarandEugene Spaord. Apattern matching model for misuse intrusion detection. In *17th National Computer Security Conference*, 1994.

Ramla Saddem and Alexandre Philippot. Causal temporal signature from diagnoser model for online diagnosis of discrete event systems. In *Control, Decision and Information Technologies (CoDIT)*, pages 551–556, Nov 2014.

D. Dzung, M. Naedele, T. P. Von Hoff, and M. Crevatin. Security for industrial communication systems. *Proceedings of the IEEE*, 93(6) :1152–1177, June 2005. ISSN 0018-9219. doi : 10.1109/JPROC.2005.849714.

Sarwarul Chowdhury and Martin Maier. Security issues in integrated epon and next-generation wlan networks. In *2010 7th IEEE Consumer Communications and Networking Conference*, pages 1–2. IEEE, 2010.

Kevin G Lyn, Lee W Lerner, Christopher J McCarty, and Cameron D Patterson. The trustworthy autonomic interface guardian architecture for cyber-physical systems. In *2015 IEEE International Conference on Computer and Information Technology ; Ubiquitous Computing and Communications ; Dependable, Autonomic and Secure Computing ; Pervasive Intelligence and Computing*, pages 1803–1810. IEEE, 2015.

Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985.

Pascal Lhoste. Contribution au Génie Automatique : Concepts, Modèles, Méthodes et Outils, February 1994. URL https://hal.archives-ouvertes.fr/hal-00137246. Habilitation à Diriger des Thèses.

Alexandre Philippot, Pascale Marangé, François Gellot, Jean-François Pétin, and Bernard Riera. Fault tolerant control for manufacturing discrete systems by filter and diagnoser interactions. In *Annual Conference of the Prognostics and Health Management Society, PHM Conference 2014*, Dallas, United States, 2014. URL https://hal.archives-ouvertes.fr/hal-01094956.

A. Terai, S. Abe, S. Kojima, Y. Takano, and I. Koshijima. Cyber-attack detection for industrial control system monitoring with support vector machine based on communication profile. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pages 132–138, April 2017. doi : 10.1109/EuroSPW.2017.62.

J. Wang, F. Yang, T. Chen, and S. L. Shah. An overview of industrial alarm systems : Main causes for alarm overloading, research status, and open problems. *IEEE Transactions on Automation Science and Engineering*, 13(2) :1045–1061, April 2016. ISSN 1545-5955. doi : 10.1109/TASE.2015.2464234.

Franck Sicard, Eric Zamaï, and Jean-Marie Flaus. Distance Concept Based Filter Approach for Detection of Cyberattacks on Industrial Control Systems. In *20th World Congress of the International Federation of Automatic Control (IFAC 2017)*, Preprints IFAC WC 2017 Toulouse., Toulouse, France, July 2017. IFAC. URL https://hal.archives-ouvertes.fr/hal-01562593. GdR MACS Young PhD Researchers - Open Invited Track of Extended Abstract.

Amer Atta Yaseen and Mireille Bayart. Attack-tolerant networked control system : an approach for detection the controller stealthy hijacking attack. In *Journal of Physics : Conference Series*, volume 783, page 012022. IOP Publishing, 2017.

Romain Pichard, Alexandre Philippot, Ramla Saddem, and Bernard Riera. Safety of manufacturing systems controllers by logical constraints with safety filter. *IEEE Transactions on Control Systems Technology*, 2018.

William Jardine, Sylvain Frey, Benjamin Green, and Awais Rashid. Senami : Selective non-invasive active monitoring for ics intrusion detection. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 23–34. ACM, 2016.

Bernard Riera, Romain Pichard, Alexandre Philippot, Ramla Saddem, François Gellot, David Annebicque, and Fabien Emprin. Home i/o et factory i/o : 2 logiciels innovants de simulation de po pour la formation à l'automatique. In *12e Colloque consacré à l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes*, 2017.

BK Penney and AA Baghdadi. Survey of computer communications loop networks : Part 1. *Computer Communications*, 2(4) :165 – 180, 1979. ISSN 0140-3664. doi : https://doi.org/10.1016/0140-3664(79)90091-4. URL http://www.sciencedirect.com/science/article/pii/0140366479900914.

Eric Niel and Etienne Craye. Maîtrise des risques et sûreté de fonctionnement des systèmes de production. *Productique : information, commande, communication. Lavoisier*, 2002.

Abdoul Karim Armand Toguyéni, Pascal Berruet, and Etienne Craye. Models and algorithms for failure diagnosis and recovery in fmss. *International Journal of Flexible Manufacturing Systems*, 15(1) :57–85, Jan 2003. ISSN 1572-9370. doi : 10.1023/A:1023905023772. URL https://doi.org/10.1023/A:1023905023772.

Alain Bignon. *Joint generation of controls and user interfaces for reconfigurable sociotechnical systems*. These, Université de Bretagne Sud, July 2012. URL https://tel.archives-ouvertes.fr/tel-00735869.

DL Kuhn. Selecting and effectively using a computer aided software engineering tool. In *Annual Westinghouse computer symposium*, Pitsburg, USA, November, 1989 DOE Project, 1989

ID Landau and A Besançon-Voda. Identification des systèmes. traité ic2–section systèmes automatisés. *Hermès, Paris*, 2001.

Joaquin Miller, Jishnu Mukerji, et al. Model driven architecture (mda). *Object Management Group, Draft Specification ormsc/2001-07-01*, page 17, 2001.

Joaquin Miller, Jishnu Mukerji, et al. Mda guide version 1.0. 1, 2003. *Object Management Group : Needham*, 2003.

Franck Fleurey. *Langage et méthode pour une ingénierie des modèles fiable*. These, Université Rennes 1, October 2006. URL https://tel.archives-ouvertes.fr/tel-00538288.

OMG MOF. Omg's metaobject facility. *Object Management Group,,. En ligne.< http ://www. omg. org/mof>. Consulté le*, 30(09) :2008, 2006.

Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF : eclipse modeling framework*. Pearson Education, 2008.

Euriell Le Corronc, Alexandre Sahuguède, and Yannick Pencolé. Détection et localisation de fautes temporelles dans les systèmes (max,+)-linéaires. In *Modélisation des Systèmes Réactifs (MSR 2017)*, page 14p., Marseille, France, November 2017. URL https://hal.laas.fr/hal-01710331.

Romain Pichard, Alesandre Philippot, and Bernard Riera. Consistency checking of safety constraints for manufacturing systems with graph analysis. *IFAC-PapersOnLine*, 50(1) :1193–1198, 2017.

Hamzat Olanrewaju Aliyu. *An Integrative Framework for Model-Driven Systems Engineering : Towards the Co-Evolution of Simulation, Formal Analysis and Enactment Methodologies for Discrete Event Systems*. These, Université Blaise Pascal - Clermont-Ferrand II, December 2016. URL https://tel.archives-ouvertes.fr/tel-01539439.

Willy Allègre. *Model-driven flow for assistive home automation systems control and supervision*. These, Université de Bretagne Sud, December 2012. URL https://tel.archives-ouvertes.fr/tel-00803402.

Ramla Saddem, Toguyeni Armand, and Tagina Moncef. Algorithme d'interprétation d'une base de signatures temporelles causales pour le diagnostic en ligne des systèmes à événements discrets. In *9th International Conference on Modeling, Optimization & SIMulation*, 2012.

Wassim EL OSTA. *Surveillabilité structurelle et platitude pour le diagnostic des modèles Bond Graph couplés*. These, Ecole Centrale Lille ; Université des Sciences et Technologies de Lille , December 2005. URL https://hal.archives-ouvertes.fr/tel-01737497.

Ganesha Nur Laksmana, Prianggada Indra Tanaya, and Yuki Indrayadi. Benchmarking and configuration of opensource manufacturing execution system (mes) application. *CommIT (Communication and Information Technology) Journal*, 7(1) :1–6, 2013.

Site web de l'outil wireshark. https://www.wireshark.org/, 1990. Analyseur de paquets libre et gratuit.

Site web de l'outil vivado. https://www.xilinx.com/products/design-tools/vivado.html, 1990. Outil de dévellopement et synthèse sur FPGA.

ATMEL. page web de la carte électronique sama5d3. https://www.microchip.com/design-centers/32-bit-mpus/microprocessors/sama5/sama5d3-series, 2014.

YP. Site web de l'os yocto. http://www.yoctoproject.org/, 2011.

Russell Smith et al. Open dynamics engine, 2005.

Thomas Toublanc, Romain Bévan, Florent de Lamotte, and Pascal Berruet. Assisting the configuration of intelligent safety gateway. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 5875–5880. IEEE, 2018.

Thomas Toublanc, Sébastien Guillet, Florent de Lamotte, Pascal Berruet, and Vianney Lapotre. Using a virtual plant to support the development of intelligent gateway for sensors/actuators security. *IFAC-PapersOnLine*, 50(1) :5837–5842, 2017.

Christian Lesjak, Daniel Hein, and Johannes Winter. Hardware-security technologies for industrial iot : Trustzone and security controller. In *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*, pages 002589–002595. IEEE, 2015.

Rong Yin, Feng Zhang, Min Liu, Feng Gui, Fei Li, and Weiming Shen. Communication model of embedded multi-protocol gateway for mro online monitoring system. In *Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on*, pages 97–102. IEEE, 2015.

Zane R. Franklin, Cameron D. Patterson, Lee Wilmoth Lerner, and Ron J. Prado. Isolating trust in an industrial control system-on-chip architecture. In *Resilient Control Systems (ISRCS)*, pages 1–6, Aug 2014.

Monowar Hussain Bhuyan, Dhruba K. Bhattacharyya, and Jugal K. Kalista. Network anomaly detection : method, system and tools. *IEEE CST*, pages 303–336, "January"/"February"/"March" 2014.

Pascal Cotret, Jérémie Crenne, Guy Gogniat, and Jean-Philippe Diguet. Bus-based mpsoc security through communication protection : A latency-efficient alternative. In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pages 200–207. IEEE, 2012.

Jun Zou, Kaining Lu, and Zhigang Jin. Architecture and fuzzy adaptive security algorithm in intelligent firewall. In *MILCOM 2002. Proceedings*, volume 2, pages 1145–1149. IEEE, 2002.

Annexes

Tables des annexes

Règle : 1 – transfert avec vérin sur axe support avec butées & capteur

```
rule binaryTreeT4T7{
    from
        s : MMComponent!Transfer(s.srcZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
            Sensor))
                                and  s.srcZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
                                    DAStopper))
                                and not s.dstZone.comp -> exists(c | c.oclIsTypeOf(
                                    MMComponent!DAStopper))
                                and  if(s.srcZone.comp -> exists(c1 | c1.oclIsTypeOf(
                                    MMComponent!DAJack)))
                                    then
                                        if(s.srcZone.comp -> select(e | e.oclIsTypeOf(
                                            MMComponent!DAJack))->first().views -> select
                                            (e | e.oclIsTypeOf(MMComponent!MaterialView))
                                            ->first().materialParameters.lenghtOfRace =
                                        s.srcZone.comp -> select(e | e.oclIsTypeOf(
                                            MMComponent!Conveyor))->first().views ->
                                            select(e | e.oclIsTypeOf(MMComponent!
                                            MaterialView))->first().materialParameters.width
                                            )
                                        then false
                                        else true
                                        endif
                                    else false
                                    endif
                                )
    using{ sensSRC: MMComponent!Component = s.srcZone.comp -> select(e | e.oclIsTypeOf(
        MMComponent!Sensor))->first();
            sensDST: MMComponent!Component = s.dstZone.comp -> select(e | e.oclIsTypeOf(
                MMComponent!Sensor))->first();
            stoppSRC: MMComponent!Component = s.srcZone.comp -> select(e | e.oclIsTypeOf
                (MMComponent!DAStopper))->first();
            compOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!EnrichedBaseComponent))->first().childs ->
                select(e | e.oclIsTypeOf(MMComponent!DAJack))->first();
            posIOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!EnrichedBaseComponent))->first().childs ->
                select(e | e.oclIsTypeOf(MMComponent!Sensor))->first();
            posFOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!EnrichedBaseComponent))->first().childs ->
                select(e | e.oclIsTypeOf(MMComponent!Sensor))->last();
    }
    to
        oc : MMComponent!Transfer (component <- thisModule.resolveTemp(s.component, '
            ecc'),
            name <- s.name,
            globalVariable <- s.globalVariable,
            controlConstraint <- s.controlConstraint,
            srcZone <- s.srcZone,
            dstZone <- s.dstZone,
```

```
            distance <- s.distance),
        bt1 : MMComponent!BinaryTree(monitoringSafetyModel <- thisModule.resolveTemp(s.
            component, 'assm'),
                name <- 'Orders-Transfert-ECC-'+compOP.name+'O_GO',
                var2Sec <- compOP.basicOperations->select(o | o.oclIsKindOf(MMComponent!
                    GoOut))->first().oCommand),
        L1_bt1: MMComponent!Leaf (unaryOperator <- 'ONE',
            binaryTree <- bt1,
            logicEnvVar <- stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Return)) -> first().oCommand,
            name <- 'ONE-' + stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Return)) -> first().oCommand.name),
--  bt2 : MMComponent!BinaryTree(
--   monitoringSafetyModel <- thisModule.resolveTemp(s.component, 'assm'),
--   name <- 'Orders-Transfert-ECC-'+compOP.name,
--   var2Sec <- compOP.basicOperations->select(o | o.oclIsKindOf(MMComponent!Return))
    ->first().oCommand
--  ),
--  L1_bt2: MMComponent!Leaf (
--   unaryOperator <- 'One',
--   binaryTree <- bt2,
--   logicEnvVar <- stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(MMComponent!
    GoOut)) -> first().oCommand,
--   name <- 'One-' + stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(MMComponent!
    GoOut)) -> first().oCommand.name
--  ),
        bt3 : MMComponent!BinaryTree(monitoringSafetyModel <- thisModule.resolveTemp(s.
            component, 'assm'),
        name <- 'Enviro-Transfert-ECC-'+compOP.name+'O_GO',
        var2Sec <- compOP.basicOperations->select(o | o.oclIsKindOf(MMComponent!GoOut))
            ->first().oCommand),
        N1_bt3: MMComponent!Node (binaryTree <- bt3,
            binaryOperator <- 'OR',
            name <- 'LEAF1' + '-' + 'OR' + '-' + 'LEAF2'),
        L1_bt3: MMComponent!Leaf (unaryOperator <- 'ONE',
            node <- N1_bt3,
            logicEnvVar <- sensSRC.basicOperations -> select(e | e.oclIsTypeOf(MMComponent
                !Detect)) -> first().iCommand,
            name <- 'ONE-' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Detect)) -> first().iCommand.name),
        L2_bt3: MMComponent!Leaf (unaryOperator <- 'NOT',
            node <- N1_bt3,
            logicEnvVar <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(MMComponent!
                Detect)) -> first().iCommand,
            name <- 'NOT-' + posIOP.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Detect)) -> first().iCommand.name),
        bt4 : MMComponent!BinaryTree(monitoringSafetyModel <- thisModule.resolveTemp(s.
            component, 'assm'),
                name <- 'Enviro-Transfert-ECC-'+compOP.name+'O_R',
                var2Sec <- compOP.basicOperations->select(o | o.oclIsKindOf(MMComponent!
                    Return))->first().oCommand),
        N1_bt4: MMComponent!Node (binaryTree <- bt4,
```

```
        binaryOperator <- 'OR',
        name <- 'LEAF1' + '-' + 'OR' + '-' + 'LEAF2'),
    L1_bt4: MMComponent!Leaf (unaryOperator <- 'NOT',
        node <- N1_bt4,
        logicEnvVar <- sensSRC.basicOperations -> select(e | e.oclIsTypeOf(MMComponent
            !Detect)) -> first().iCommand,
        name <- 'NOT-' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand.name),
    L2_bt4: MMComponent!Leaf (unaryOperator <- 'NOT',
        node <- N1_bt4,
        logicEnvVar <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(MMComponent!
            Detect)) -> first().iCommand,
        name <- 'NOT-' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand.name)
}
```

Règle : 2 – requêtes de position d'un vérin 2 positions

```
rule vueMonitoringEBC2 {
    from
        s: MMComponent!EnrichedBaseComponent (if (s.childs -> select(e | e.oclIsTypeOf(
            MMComponent!Sensor)) -> size() = 2)
                then
                true
            else
                false
            endif
        )
    using{ compOP : MMComponent!Component = s.component.childs -> select(e | e.
        oclIsTypeOf(MMComponent!DAJack))->first();
            posIOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!Sensor))->first();
            posFOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!Sensor))->last();
            speedOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!DAJack))->first().views->select(e | e->
                oclIsKindOf(Component!MaterialView))->first().materialParameters.speed;
            lenghtOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!DAJack))->first().views->select(e | e->
                oclIsKindOf(Component!MaterialView))->first().materialParameters.
                lenghtOfRace;}
    to
        ebc: MMComponent!EnrichedBaseComponent (name <- s.name,
            father <- s.father,
            io <- s.io,
            views <- s.views,
            childs <- s.childs),
        sv: MMComponent!MonitoringView (name <- '-' + s.name,
            component <- s),
        aomm: MMComponent!AutoOperationalMonitoringModel (name <- 'AOMM-LVL1-' +
            s.name,
            view <- sv),
        cts1: MMComponent!CausalTemporalSignature (monitoringOperationalModel <- aomm
            ,
            unobservableEvent <- 'Sensor PI faillure'),
        O1_cts1: MMComponent!Operator (cTSignature <- cts1,
            binaryOperator <- 'AND',
            name <- 'T1' + '-' + 'AND' + '-' + 'T2'),
        T1_cts1: MMComponent!Triplet (unaryOperator <- 'NOT',
            cTSignature <- cts1,
            referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!GoOut)) -> first().oCommand,
            frontEvent <- 'Falling',
            monitoredEvent <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Detect)) -> first().iCommand,
            contrainteTemporelle <- 0,
            name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!GoOut)) -> first().oCommand.name
```

```
                + ', R' + posIOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ', 0 )'),
    T2_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
        cTSignature <- cts1,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand,
        frontEvent <- 'Rizing',
        monitoredEvent <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- speedOP."div"(lenghtOP),
        name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand.name
                + ', R' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ',' + speedOP + '/' + lenghtOP + ' )'),
    cts2: MMComponent!CausalTemporalSignature (monitoringOperationalModel <- aomm
        ,
        unobservableEvent <- 'ED1'),
    O1_cts2: MMComponent!Operator (cTSignature <- cts2,
        binaryOperator <- 'AND',
        name <- 'T1' + '-' + 'AND' + '-' + 'T2'),
    T1_cts2: MMComponent!Triplet (unaryOperator <- 'ONE',
        cTSignature <- cts2,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand,
        frontEvent <- 'Falling',
        monitoredEvent <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- 0,
        name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand.name
                + ', F' + posIOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ', 0 )'),
    T2_cts2: MMComponent!Triplet (unaryOperator <- 'NOT',
        cTSignature <- cts2,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand,
        frontEvent <- 'Rizing',
        monitoredEvent <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- speedOP."div"(lenghtOP),
        name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand.name
                + ', R' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ',' + speedOP + '/' + lenghtOP + ' )'),
    cts3: MMComponent!CausalTemporalSignature (monitoringOperationalModel <- aomm
        ,
        unobservableEvent <- 'ED2'),
    O1_cts3: MMComponent!Operator (cTSignature <- cts3,
```

```
    binaryOperator <- 'AND',
    name <- 'T1' + '-' + 'AND' + '-' + 'T2'),
T1_cts3: MMComponent!Triplet (unaryOperator <- 'NOT',
    cTSignature <- cts3,
    referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand,
    frontEvent <- 'Falling',
    monitoredEvent <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Detect)) -> first().iCommand,
    contrainteTemporelle <- 0,
    name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand.name
                + ', F' + posIOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ', 0 )'),
T2_cts3: MMComponent!Triplet (unaryOperator <- 'NOT',
    cTSignature <- cts3,
    referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand,
    frontEvent <- 'Rizing',
    monitoredEvent <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Detect)) -> first().iCommand,
    contrainteTemporelle <- speedOP."div"(lenghtOP),
    name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand.name
                + ', R' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ',' + speedOP + '/' + lenghtOP + ' )'),
cts4: MMComponent!CausalTemporalSignature (monitoringOperationalModel <- aomm
    ,
    unobservableEvent <- 'Sensor PF faillure'),
O1_cts4: MMComponent!Operator (cTSignature <- cts4,
    binaryOperator <- 'AND',
    name <- 'T1' + '-' + 'AND' + '-' + 'T2'),
T1_cts4: MMComponent!Triplet (unaryOperator <- 'NOT',
    cTSignature <- cts4,
    referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Return)) -> first().oCommand,
    frontEvent <- 'Falling',
    monitoredEvent <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Detect)) -> first().iCommand,
    contrainteTemporelle <- 0,
    name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Return)) -> first().oCommand.name
                + ', F' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ', 0 )'),
T2_cts4: MMComponent!Triplet (unaryOperator <- 'ONE',
    cTSignature <- cts4,
    referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Return)) -> first().oCommand,
    frontEvent <- 'Rizing',
```

```
        monitoredEvent <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- speedOP."div"(lenghtOP),
        name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand.name
                    + ', R' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!Detect)) -> first().iCommand.name
                    + ',' + speedOP + '/' + lenghtOP + ' )'),
    cts5: MMComponent!CausalTemporalSignature (monitoringOperationalModel <- aomm
        ,
        unobservableEvent <- 'ED3'),
    O1_cts5: MMComponent!Operator (cTSignature <- cts5,
        binaryOperator <- 'AND',
        name <- 'T1' + '-' + 'AND' + '-' + 'T2'),
    T1_cts5: MMComponent!Triplet (unaryOperator <- 'ONE',
        cTSignature <- cts5,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand,
        frontEvent <- 'Falling',
        monitoredEvent <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- 0,
        name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand.name
                    + ', F' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!Detect)) -> first().iCommand.name
                    + ', 0 )'),
    T2_cts5: MMComponent!Triplet (unaryOperator <- 'NOT',
        cTSignature <- cts5,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand,
        frontEvent <- 'Rizing',
        monitoredEvent <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- speedOP."div"(lenghtOP),
        name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand.name
                    + ', R' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!Detect)) -> first().iCommand.name
                    + ',' + speedOP + '/' + lenghtOP + ' )'),
    cts6: MMComponent!CausalTemporalSignature (monitoringOperationalModel <- aomm
        ,
        unobservableEvent <- 'ED4'),
    O1_cts6: MMComponent!Operator (cTSignature <- cts6,
        binaryOperator <- 'AND',
        name <- 'T1' + '-' + 'AND' + '-' + 'T2'),
    T1_cts6: MMComponent!Triplet (unaryOperator <- 'NOT',
        cTSignature <- cts6,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand,
        frontEvent <- 'Falling',
        monitoredEvent <- posFOP.basicOperations -> select(e | e.oclIsTypeOf(
```

```
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- 0,
        name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand.name
                    + ', F' + posFOP.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!Detect)) -> first().iCommand.name
                    + ', 0 )'),
    T2_cts6: MMComponent!Triplet (unaryOperator <- 'NOT',
        cTSignature <- cts6,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand,
        frontEvent <- 'Rizing',
        monitoredEvent <- posIOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- speedOP."div"(lenghtOP),
        name <- 'NOT-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Return)) -> first().oCommand.name
                    + ', R' + posIOP.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!Detect)) -> first().iCommand.name
                    + ',' + speedOP + '/' + lenghtOP + ' 0 )')
}
```

Règle : 3 – transfert par vérin avec capteur et butée en zone src et dst

```
rule RuleTransfert24 {
    from
        s : MMComponent!Transfer(s.srcZone.comp->one(c1 |s.dstZone.comp->one(c2| c1 = c2)
            )
                                and (s.srcZone.eCCTopologicalModel.view.component.childs->
                                    one(c3 | c3.oclIsTypeOf(MMComponent!Conveyor)))
                                and (s.srcZone.eCCTopologicalModel.view.component.childs->
                                    one(c3 | c3.oclIsTypeOf(MMComponent!DAJack)))
                                and (s.srcZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
                                    Sensor)))
                                and (s.dstZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
                                    Sensor)))
                                and not (s.srcZone.comp -> exists(c | c.oclIsTypeOf(
                                    MMComponent!BCReader)))
                                and not (s.dstZone.comp -> exists(c | c.oclIsTypeOf(
                                    MMComponent!BCReader)))
                                and (s.srcZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
                                    DAStopper)))
                                and (s.dstZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
                                    DAStopper)))
                                and not (s.srcZone.comp -> exists(c | c.oclIsTypeOf(
                                    MMComponent!DAJack)))
                                and (s.dstZone.comp -> exists(c | c.oclIsTypeOf(MMComponent!
                                    DAJack))))
    using{ compOP : MMComponent!Component = s.component.childs -> select(e | e.
        oclIsTypeOf(MMComponent!Conveyor))->first();
            speedOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!DAJack))->first().views->select(e | e->
                oclIsKindOf(Component!MaterialView))->first().materialParameters.speed;
            lenghtOP : MMComponent!Component = s.component.childs -> select(e | e.
                oclIsTypeOf(MMComponent!DAJack))->first().views->select(e | e->
                oclIsKindOf(Component!MaterialView))->first().materialParameters.
                lenghtOfRace;
            compEBCCTS : MMComponent!Component = s.childs -> select(e | e.oclIsTypeOf(
                MMComponent!EnrichedBaseComponent))->first().views->select(e | e->
                oclIsKindOf(Component!MonitoringView))->first().monitoringModel->select(e
                | e->oclIsKindOf(Component!AutoOperationalMonitoringModel))->first();
            sensSRC: MMComponent!Component = s.srcZone.comp -> select(e | e.oclIsTypeOf(
                MMComponent!Sensor))->first();
            sensDST: MMComponent!Component = s.dstZone.comp -> select(e | e.oclIsTypeOf(
                MMComponent!Sensor))->first();
            stoppSRC: MMComponent!Component = s.srcZone.comp -> select(e | e.oclIsTypeOf
                (MMComponent!DAStopper))->first();
--  stoppdst: MMComponent!Component = s.dstZone.comp -> select(e | e.oclIsTypeOf(
    MMComponent!DAStopper))->first();}
    to
        oc : MMComponent!Transfer (
            component <- thisModule.resolveTemp(s.component, 'ecc'),
            name <- s.name,
            globalVariable <- s.globalVariable,
```

```
        controlConstraint <- s.controlConstraint,
        srcZone <- s.srcZone,
        dstZone <- s.dstZone,
        distance <- s.distance
    ),
    cts1 : MMComponent!CausalTemporalSignature(monitoringOperationalModel <-
         thisModule.resolveTemp(s.component, 'aomm'),
        unobservableEvent <- 'Sensor PF faillure'),
    O1_cts1: MMComponent!Operator (cTSignature <- cts1,
        binaryOperator <- 'AND',
        name <- 'O2' + '-' + 'AND' + '-' + 'O3'),
    O2_cts1: MMComponent!Operator (cTSignature <- cts1,
        binaryOperator <- 'AND',
        name <- 'T1' + '-' + 'AND' + '-' + 'T1'),
    T1_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
        cTSignature <- cts1,
        referenceEvent <- stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand,
        frontEvent <- 'Rizing',
        monitoredEvent <- sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        contrainteTemporelle <- 0,
        name <- 'NOT-' + '(' + stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand.name
                    + ', R' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!Detect)) -> first().iCommand.name
                    + ', 0 )'),
    T2_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
        cTSignature <- cts1,
        referenceEvent <- sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand,
        frontEvent <- 'Rizing',
        monitoredEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand,
        contrainteTemporelle <- 40, -- 2 x 20ms (Automation cycle time) to be sure
        name <- 'ONE-' + '(' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!Detect)) -> first().iCommand.name
                    + ', R' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
                        MMComponent!GoOut)) -> first().oCommand.name
                    + ',' + '40' + ' )'),
    O3_cts1: MMComponent!Operator (cTSignature <- cts1,
        binaryOperator <- 'AND',
        name <- 'O4' + '-' + 'AND' + '-' + 'O5'),
    O4_cts1: MMComponent!Operator (cTSignature <- cts1,
        binaryOperator <- 'AND',
        name <- 'T3' + '-' + 'AND' + '-' + 'T4'),
    T3_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
        cTSignature <- cts1,
        referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
            MMComponent!GoOut)) -> first().oCommand,
        frontEvent <- 'Falling',
        monitoredEvent <- sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
```

```
        MMComponent!Detect)) -> first().iCommand,
    contrainteTemporelle <- 40, -- 2 x 20ms (Automation cycle time) to be sure
    name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand.name
                + ', F' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ',' + '40' + ' )'),
T4_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
    cTSignature <- cts1,
    referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand,
    frontEvent <- 'Rising',
    monitoredEvent <- sensDST.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Detect)) -> first().iCommand,
    contrainteTemporelle <- (speedOP."div"(lenghtOP))*1000,
    name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand.name
                + ', R' + sensDST.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!Detect)) -> first().iCommand.name
                + ',' + speedOP + '/' + lenghtOP + ' )'),
O5_cts1: MMComponent!Operator (cTSignature <- cts1,
    binaryOperator <- 'AND',
    name <- 'T5' + '-' + 'AND' + '-' + 'O6'),
T5_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
    cTSignature <- cts1,
    referenceEvent <- compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand,
    frontEvent <- 'Rising',
    monitoredEvent <- compEBCCTS.causalTemporalSignature -> at(2),
    contrainteTemporelle <- (speedOP."div"(lenghtOP))*1000,
    name <- 'ONE-' + '(' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!GoOut)) -> first().oCommand.name
                + ', R' + compEBCCTS.causalTemporalSignature -> at(2).
                    unobservableEvent
                + ',' + speedOP + '/' + lenghtOP + ' )'),
O6_cts1: MMComponent!Operator (cTSignature <- cts1,
    binaryOperator <- 'AND',
    name <- 'T6' + '-' + 'AND' + '-' + 'T7'),
T6_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
    cTSignature <- cts1,
    referenceEvent <- sensDST.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Detect)) -> first().iCommand,
    frontEvent <- 'Rising',
    monitoredEvent <- compEBCCTS.causalTemporalSignature -> at(4),
    contrainteTemporelle <- (speedOP."div"(lenghtOP))*1000,
    name <- 'ONE-' + '(' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
        MMComponent!Detect)) -> first().iCommand.name
                + ', R' + compOP.basicOperations -> select(e | e.oclIsTypeOf(
                    MMComponent!GoOut)) -> first().oCommand.name
                + ',' + speedOP + '/' + lenghtOP + ' )'),
T7_cts1: MMComponent!Triplet (unaryOperator <- 'ONE',
    cTSignature <- cts1,
```

```
            referenceEvent <- sensDST.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Detect)) -> first().iCommand,
            frontEvent <- 'Rising',
            monitoredEvent <- stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Return)) -> first().oCommand,
            contrainteTemporelle <- (speedOP."div"(lenghtOP))*1000,
            name <- 'ONE-' + '(' + sensSRC.basicOperations -> select(e | e.oclIsTypeOf(
                MMComponent!Detect)) -> first().iCommand.name
                        + ', R' + stoppSRC.basicOperations -> select(e | e.oclIsTypeOf(
                            MMComponent!Return)) -> first().oCommand
                        + ',' + speedOP + '/' + lenghtOP + ' )'),
        cts2 : MMComponent!CausalTemporalSignature(monitoringOperationalModel <-
            thisModule.resolveTemp(s.component, 'aomm'),
            unobservableEvent <- 'Jack faill during operation at Go_Out phase'),
-- not detailled
        cts3 : MMComponent!CausalTemporalSignature(monitoringOperationalModel <-
            thisModule.resolveTemp(s.component, 'aomm'),
            unobservableEvent <- 'Jack faill during operation at return phase'),
-- not detailled
        cts4 : MMComponent!CausalTemporalSignature(monitoringOperationalModel <-
            thisModule.resolveTemp(s.component, 'aomm'),
            unobservableEvent <- 'Jack faill to start operation'),
-- not detailled
        cts5 : MMComponent!CausalTemporalSignature(monitoringOperationalModel <-
            thisModule.resolveTemp(s.component, 'aomm'),
            unobservableEvent <- 'Sensor PI \ PF faillure')
-- not detailled
        cts6 : MMComponent!CausalTemporalSignature(monitoringOperationalModel <-
            thisModule.resolveTemp(s.component, 'aomm'),
            unobservableEvent <- 'Jack block during operation'),
-- not detailled
}
```

Règle : 4 – transformation modèle ComGEM en Configuration pour ASSecIN

```
rule Sytem2GatewayRefModel{
    from
        c  : MMComponent!System
    to
        s  : ASSecIN!GatewayRefModel(version <- '1.1',
            path <- 'TransformationsATL/models/ASSecIN.assecin'),
        phys  : ASSecIN!PhysicalReference(gatewayRefModel <- s),
        variable: ASSecIN!Variables(physicalRef <- phys),
        vg1  : ASSecIN!VarGroup(name <- '%IX0',
            kind <- 'IO',
            variables <- variable),
        vg2  : ASSecIN!VarGroup(name <- '%QX1',
            kind <- 'IO',
            variables <- variable),
        vg3  : ASSecIN!VarGroup(name <- '(Global)',
            kind <- 'GLOBAL',
            variables <- variable),
        DR  : ASSecIN!DetectionReference(gatewayRefModel <- s)
        RR  : ASSecIN!ReactionReference(gatewayRefModel <- s)
}
rule CommandInput2AVar{
    from
        c : MMComponent!CommandInput(not c.component.oclIsTypeOf(MMComponent!
            BCReader) and not c.component.oclIsTypeOf(MMComponent!RotaryEncoder))
    using{compSys : MMComponent!System = MMComponent!System.allInstances()->
        asSequence()->first();}
    to
        v : ASSecIN!Var(name <- if (MMComponent!CommandInput.allInstances().size()>1)
                                then c.name
                            else '%IX0.'+thisModule.numbI(c).toString() endif,
            type <- 'BOOL',
--  addresse <- '0x01'+'0000'+thisModule.numbI(c).toString(),
            indexH <- '0',
            indexL <- thisModule.numbI(c).toString(),
            dir <- #INPUT,
            groupName <- if (MMComponent!CommandInput.allInstances().size()>1)
                                    then '(Global)'
                        else '%IX0'
                        endif,
            varGroup <- if (MMComponent!CommandInput.allInstances().size()>1)
                          then thisModule.resolveTemp(compSys, 'vg3')
                         else thisModule.resolveTemp(compSys, 'vg1')
                         endif)
}
rule CommandOutput2Var{
    from
        c : MMComponent!CommandOutput(not c.component.oclIsTypeOf(MMComponent!
            BCReader) and not c.component.oclIsTypeOf(MMComponent!RotaryEncoder))
    using{ compSys : MMComponent!System = MMComponent!System.allInstances()->
        asSequence()->first();
```

```
            ci: MMComponent!CommandInput = MMComponent!CommandInput.allInstances()
                ->select(e| not e.component.oclIsTypeOf(MMComponent!BCReader))->
                asSequence()->last();
    }
    to
        v : ASSecIN!Var(name <- if (MMComponent!CommandOutput.allInstances().size()>1)
                                then c.name
                            else '%QX1.'+thisModule.numbO(c).toString() endif,
            type <- 'BOOL',
--  addresse <- '0x01'+'0000'+(thisModule.numbI(ci)+thisModule.numbO(c)).toString(),
            indexH <- '1',
            indexL <- thisModule.numbO(c).toString(),
            dir <- #OUTPUT,
            groupName <- if (MMComponent!CommandOutput.allInstances().size()>1)
                                        then '(Global)'
                        else '%QX1'
                        endif,
            varGroup <- if (MMComponent!CommandOutput.allInstances().size()>1)
                          then thisModule.resolveTemp(compSys, 'vg3')
                        else thisModule.resolveTemp(compSys, 'vg2')
                        endif)
}
```

Règle : 5 – Règle faisant correspondre le SMM à la DR

```
rule SafetySurveyModel2SafetyDetectionReference{
    from
        c  : MMComponent!SafetySurveyModel
    using{compSys : MMComponent!System = MMComponent!System.allInstances()->
         asSequence()->first();}
    to
        safe  : ASSecIN!SafetyDetectionReference(name <- c.name,
            detectionRef <- thisModule.resolveTemp(compSys, 'DR'))
}
rule BinaryTree2BT{
    from
        c : MMComponent!BinaryTree
    to
        BT : ASSecIN!BinaryTree(safetyDetectionReference <- c.safetySurveyModel,
            var2Sec <- c.var2Sec)}
rule Node2Node{
    from
        c : MMComponent!Node
    using{compByT : MMComponent!BinaryTree = MMComponent!BinaryTree.allInstances()->
         asSequence()->first();}
    to
        N : ASSecIN!Node(binaryTree <- c.binaryTree,
            name <- c.name,
            bOperator <- c.binaryOperator,
            node <- c.node)
}
rule Leaf2Leaf{
    from
        c : MMComponent!Leaf
    using{compByT : MMComponent!BinaryTree = MMComponent!BinaryTree.allInstances()->
         asSequence()->first();}
    to
        L : ASSecIN!Leaf(binaryTree <- c.binaryTree,
            name <- c.name,
            uOperator <- c.unaryOperator,
            logicEnv <- c.logicEnvVar,
            node <- c.node)
}
```

Règle : 6 – Règle faisant correspondre le OMM à la DR

```
rule OperationalMonitoringModel2OperationalDetectionReference{
    from
        c  : MMComponent!OperationalMonitoringModel
    using{compSys : MMComponent!System = MMComponent!System.allInstances()->
         asSequence()->first();}
    to
        CH : ASSecIN!Chronicle(operationalDetectionReference <- ope,
            name <- 'Chronicle'+c.view.component.name),
        ET : ASSecIN!StartPlace(chronicle <- CH,
            name <- 'Ptrigerring',
            trigerringVar <- c.causalTemporalSignature->first().cTSElements->select(e | e.
                oclIsTypeOf(MMComponent!Triplet))->first().referenceEvent),
        ED : ASSecIN!Place(chronicle <- CH,
            name <- 'Pfaillure',
            front <- 'Rizing',
            deffaillanceEvent <- c.causalTemporalSignature->first().unobservableEvent)
}
rule Triplet2subsubgraphe{
    from
        c : MMComponent!Triplet(
    using{compCTS : MMComponent!CausalTemporalSignature = MMComponent!
         CausalTemporalSignature.allInstances()->asSequence()->first();}
    to
        P1 : ASSecIN!Place(chronicle <- thisModule.resolveTemp(compCTS, 'CH'),
            name <- 'P-reference'+c.name,
            front <- OclUndefined,
            eventVar <- c.referenceEvent),
        A : ASSecIN!Arc(chronicle <- thisModule.resolveTemp(compCTS, 'CH'),
            name <- 'ARC'+c.name,
            nodeChronicle <- P1 + P2,
            minimalTime <- c.contrainteTemporelle-10,
            maximalTime <- c.contrainteTemporelle+10),
        P2 : ASSecIN!Place(chronicle <- thisModule.resolveTemp(compCTS, 'CH'),
            name <- 'P-monitored'+c.name,
            front <- c.frontEvent,
            eventVar <- c.monitoredEvent)
}
```

Résultat : 7 – génération de la références physique

```
<physicalReference>
  <variables>
    <varGroup name="%IX0" kind="IO"/>
    <varGroup name="%QX1" kind="IO"/>
    <varGroup name="(Global)" kind="GLOBAL">
      <vars name="PI_J1_I_D"  indexH="0"indexL="1"dir="INPUT"leaf="x"/>
      <vars name="PF_J1_I_D"  indexH="0"indexL="2"dir="INPUT"leaf="x"/>
      <vars name="CI_I_D"   indexH="0"indexL="3"dir="INPUT"leaf="x"/>
      <vars name="CE_I_D"   indexH="0"indexL="4"dir="INPUT"leaf="x"/>
      <vars name="PI_J2_I_D"  indexH="0"indexL="5"dir="INPUT"leaf="x"/>
      <vars name="PM_J2_I_D"  indexH="0"indexL="6"dir="INPUT"leaf="x"/>
      <vars name="PF_J2_I_D"  indexH="0"indexL="7"dir="INPUT"leaf="x"/>
      <vars name="C1_I_D"   indexH="0"indexL="8"dir="INPUT"leaf="x"/>
      <vars name="C2_I_D"   indexH="0"indexL="9"dir="INPUT"leaf="x"/>
      <vars name="C3_I_D"   indexH="0"indexL="10"dir="INPUT"leaf="x"/>
      <vars name="PI_J3_I_D"  indexH="0"indexL="11"dir="INPUT"leaf="x"/>
      <vars name="PF_J3_I_D"  indexH="0"indexL="12"dir="INPUT"leaf="x"/>
      <vars name="CEj_I_D"  indexH="0"indexL="13"dir="INPUT"leaf="x"/>
      <vars name="CS_I_D"   indexH="0"indexL="14"dir="INPUT"/>
      <vars name="Conv_line_1_O_M"indexH="1"indexL="1"dir="OUTPUT"/>
      <vars name="Conv_line_1_O_R"indexH="1"indexL="2"dir="OUTPUT"/>
      <vars name="Jack1_O_GO"  indexH="1"indexL="3"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Jack1_O_R"  indexH="1"indexL="4"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_E_O_GO" indexH="1"indexL="5"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_E_R"  indexH="1"indexL="6"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_Ai_O_GO" indexH="1"indexL="7"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_Ai_O_R" indexH="1"indexL="8"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Jack2_O_GO"  indexH="1"indexL="9"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Jack2_O_R"  indexH="1"indexL="10"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_A_O_GO" indexH="1"indexL="11"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_A_O_R" indexH="1"indexL="12"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_T_O_GO" indexH="1"indexL="13"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Stopp_T_O_R" indexH="1"indexL="14"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Conv_line_2_O_M"indexH="1"indexL="15"dir="OUTPUT"/>
      <vars name="Jack3_O_GO"  indexH="1"indexL="16"dir="OUTPUT"bTree="x"leaf="x"/>
      <vars name="Jack3_O_R"  indexH="1"indexL="17"dir="OUTPUT"bTree="x"leaf="x"/>
    </varGroup>
  </variables>
</physicalReference>
```

Résultat : 8 – génération de la références de détection sûreté

```
<detectionReference>
  <safetyDetectionRef name="SDR_System_safe_LVL4">
    <binarytree var2Sec="1.3">
      <Node binaryOperator="AND">
        <Leaf unaryOperator="NOT" logicEnv="1.9"/>
        <Leaf unaryOperator="ONE" logicEnv="1.11"/>
      </Node>
    </binarytree>
    <binarytree var2Sec="1.9">
      <Node binaryOperator="OR">
        <Leaf unaryOperator="NOT" logicEnv="1.16"/>
        <Leaf unaryOperator="ONE" logicEnv="1.13"/>
      </Node>
    </binarytree>
    <binarytree var2Sec="1.16">
      <Node binaryOperator="AND">
        <Leaf unaryOperator="ONE" logicEnv="1.7"/>
        <Leaf unaryOperator="ONE" logicEnv="1.13"/>
      </Node>
    </binarytree>
    <binarytree var2Sec="1.3">
      <Leaf unaryOperator="NOT" logicEnv="0.8"/>
    </binarytree>
    <binarytree var2Sec="1.16">
      <Leaf unaryOperator="ONE" logicEnv="0.10"/>
    </binarytree>
    <binarytree var2Sec="1.11">
      <Leaf unaryOperator="ONE" logicEnv="0.3"/>
    </binarytree>
  </safetyDetectionReference>
  <safetyDetectionRef name="SDR_intercept_safe_LVL3">
  <safetyDetectionRef name="SDR_intercepteur_safe_LVL2">
  <safetyDetectionRef name="SDR_jack1_safe_LVL1">
  <safetyDetectionRef name="SDR_Stopp_entry_safe_LVL1">
  <safetyDetectionRef name="SDR_Stopp_aigu_safe_LVL1">
  <safetyDetectionRef name="SDR_transfert_safe_LVL3">
  <safetyDetectionRef name="SDR_transferreur_safe_LVL2">
  <safetyDetectionRef name="SDR_jack2_safe_LVL1">
  <safetyDetectionRef name="SDR_stoppA_safe_LVL1">
  <safetyDetectionRef name="SDR_stoppT_safe_LVL1">
  <safetyDetectionRef name="SDR_eject_safe_LVL3">
  <safetyDetectionRef name="SDR_ejecteur_safe_LVL2">
  <safetyDetectionRef name="SDR_jack3_safe_LVL1">
```

Résultat : 9 – auto-génération des contraintes pour l'intercepteur

```
<childs xsi:type="cpn:EffectiveContextualComponent" name="ECC1">
  <views xsi:type="cpn:GraphicView" name="GraphicViewECC1">
  <views xsi:type="cpn:ConstraintView" name="ConstraintViewECC1">
  <views xsi:type="cpn:MonitoringView" name="Monitoring View-ECC1">
    <monitoringModel xsi:type="cpn:AutoSafetyMonitoringModel" name="ASSM-LVL3-ECC1">
    <binaryTrees var2Sec="O" name="Orders-Transfert-ECC-jack1_O_GO"/>
    <binaryTrees var2Sec="O" name="Enviro-Transfert-ECC-jack1_O_GO"/>
    <binaryTrees var2Sec="O" name="Enviro-Transfert-ECC-jack1_O_R"/>
    <binaryTrees var2Sec="O" name="Orders-AS-ECC-Stopper_aiguilleurO_R"/>
    <binaryTrees var2Sec="O" name="Enviro-AS-ECC-Stopper_aiguilleurO_GO"/>
    <binaryTrees var2Sec="O" name="Enviro-AS-ECC-Stopper_aiguilleurO_R"/>
  <childs xsi:type="cpn:Conveyor" name="Conv1" zone="x" baseComp="x" motor="true">
    <views xsi:type="cpn:MaterialView" name="MaterialViewConv1">
  </childs>
  <childs xsi:type="cpn:DAStopper" name="stopper1" zone="x" supportComp="x">
    <views xsi:type="cpn:MaterialView" name="MaterialViewstopper1">
    <views xsi:type="cpn:MonitoringView" name="Monitoring View-stopper1">
      <monitoringModel xsi:type="cpn:AutoSafetyMonitoringModel" name="ASSM-LVL1-stopper1">
  </childs>
  <childs xsi:type="cpn:EnrichedBaseComponent" name="EBC1">
    <views xsi:type="cpn:GraphicView" name="GraphicViewEBC1">
    <views xsi:type="cpn:MonitoringView" name="Monitoring ViewEBC1">
      <monitoringModel xsi:type="cpn:AutoSafetyMonitoringModel" name="ASSM-LVL2-EBC1">
        <binaryTrees var2Sec="O" name="BinaryTree EBC for 2 positions jack-Jack1_O_GO">
        <binaryTrees var2Sec="O" name="BinaryTree EBC for 2 positions jack-Jack1_O_R">
        <binaryTrees var2Sec="I" name="BinaryTree EBC for 2 positions jack-Sensor12_I_D">
        <binaryTrees var2Sec="I" name="BinaryTree EBC for 2 positions jack-Sensor11_I_D">
      </monitoringModel>
    <childs xsi:type="cpn:DAJack" name="Jack1" position="x" supportComp="x">
      <views xsi:type="cpn:MaterialView" name="MaterialViewJack1">
      <views xsi:type="cpn:MonitoringView" name="SV-Jack1">
        <monitoringModel xsi:type="cpn:AutoSafetyMonitoringModel" name="ASSM-LVL1-Jack1">
          <binaryTrees var2Sec="O" name="BinaryTree-Jack1_O_GO">
          <binaryTrees var2Sec="O" name="BinaryTree-Jack1_O_R">
        </monitoringModel>
    </childs>
    <childs xsi:type="cpn:Sensor" name="Sensor12" position="x" supportComp="x">
      <views xsi:type="cpn:MaterialView" name="MaterialViewSensor12">
    </childs>
    <childs xsi:type="cpn:Sensor" name="Sensor11" position="x" supportComp="x">
      <views xsi:type="cpn:MaterialView" name="MaterialViewSensor11">
    </childs>
  </childs>
  <childs xsi:type="cpn:Sensor" name="Sensor1" zone="x" supportComp="x">
    <views xsi:type="cpn:MaterialView" name="MaterialViewSensor1">
  </childs>
  <childs xsi:type="cpn:DAStopper" name="stopper2" zone="x" supportComp="x">
    <views xsi:type="cpn:MaterialView" name="MaterialViewstopper2">
    <views xsi:type="cpn:MonitoringView" name="Monitoring View-stopper2">
      <monitoringModel xsi:type="cpn:AutoSafetyMonitoringModel" name="ASSM-LVL1-stopper2">
  </childs>
  <childs xsi:type="cpn:BCReader" name="BCReader1" zone="x" supportComp="x">
    <views xsi:type="cpn:MaterialView" name="MVBCReader1">
  </childs>
  <childs xsi:type="cpn:Sensor" name="Sensor5" zone="x" supportComp="x">
    <views xsi:type="cpn:MaterialView" name="MaterialViewSensor5">
  </childs>
</childs>
```

Résultat : 10 – génération de la références supervision

```
<reactionReference>
  <safetyReaction name="SRM_system"log="YOU ARE UNDER ATTACK ON V1 AND V2">
  <safetyReaction name="SRM_jack1">
  <safetyReaction name="SRM_jack2">
  <operationalReaction name="ORM_jack3"
      <operationOutGoing operationalLog="Operation Ejection of component Ejecteur failure
          without movement" var4inform="1.16"/>
</reactionReference>
```

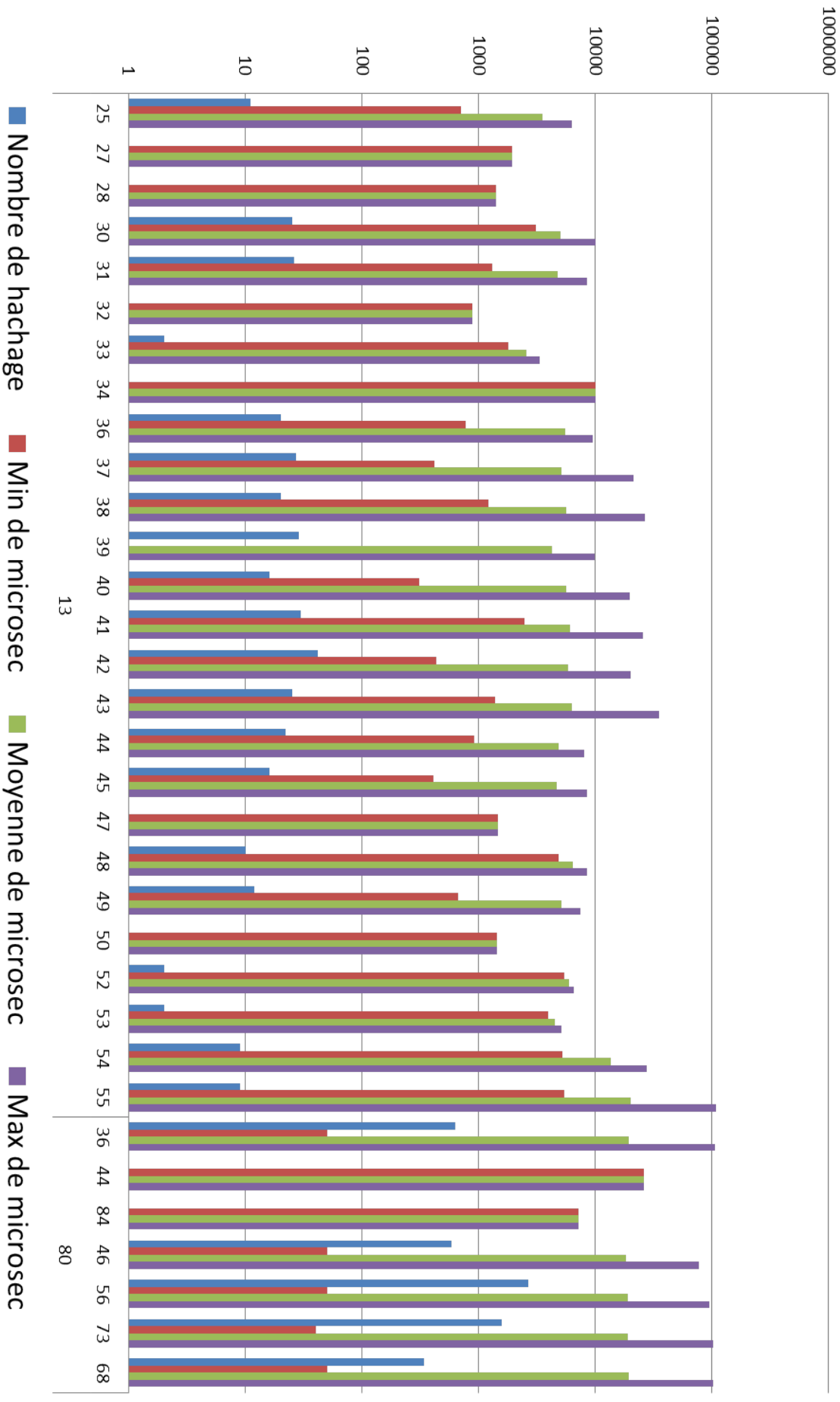


figure 11 – Analyse détaillée des trames qui nous intéressent

Acronymes

AGV véhicule autoguidé, *Automated Guided Vehicule*.

ANSSI Agence Nationale pour la Sécurité des Systèmes d'Information.

API interface de programmation applicative, *Application Programming Interface*.

ARED Allocation de REcherche Doctorale.

ATL *ATLAS Transformation Language*.

CB/BC Composant de Base, *Basic Component*.

CBE/EBC Composant de Base Enrichi, *Enriched Base Component*.

CCE/ECC Composant Contextuel Effectif, *Effective Contextual Component*.

CIM *Computer-Integrated Manufacturing*.

CLUSIF CLUb de la Sécurité de l'InFormation.

CM/HC Composant de Manutention, *Handling Components*.

CSu/SuC Composant Support, *Support Component*.

CSys/SysC Composant Système, *System Component*.

DMZ *DeMilitarized Zone*.

DPI inspection profonde de paquet, *Deep Packet Inspection*.

DSL langage de spécification métier, *Domain Specific Language*.

DZ Détection de Zone, *Detect Zone*.

DZO/DOZ Détection de Zone Occupée, *Detect Ocupied Zone*.

DZOP/DPOZ Détection de Zone Occupée partiellement, *Detect Parcel Ocupied Zone*.

EMF *Eclipse Modeling Framework*.

ERP planificateur de ressource de l'entreprise, *Entreprise Resource Planning*.

FB réseau de terrain, *Field Bus*.

FBD *Functional Block Diagram*.

FPGA réseau de portes programmables, *Field-Programmable Gate Array*.

FTP *File Transfert Protocole*.

GMAO Gestionnaire de Maintenance Assisté par Ordinateur.

GPAO Gestionnaire de Production Assisté par Ordinateur.

IACS système de contrôle automatisé industriel, *Industrial Automated Control System*.

IDE environnement de développement intégré, *Integrated Development Environment*.

IDM Ingénierie Dirigée par les Modèles, *model driven engineering*.

IDS system de détection d'intrusion, *Intrusion Detection System*.

IHM Interface Homme-Machine.

IIoT internet des objets industriels, *Industrial Internet of Thing*.

IPS system de prévention d'intrusion,*Intrusion Prevention System*.

ISO *International Standards Organisation*.

IT technologie de l'information, *Information Technology*.

MDA architecture dirigée par les modèles, *Model Driven Architecture*.

MES *Manufacturing Execution System*.

MOF *Meta Object Facility*.

NCS système contrôlé par le réseau, *Network Control System*.

NIST *National Institute of Standards and Technology*.

NTIC Nouvelle Technologie de l'Information et de la Communication.

OB/BO Opération Basique, *Basic Operation*.

OC/CO Opération Contextuelle, *Contextual Operation*.

OCE/ECO Opération Contextuelle Effective, *Effective Contextual Operation*.

ODE *Open Dynamic Engine*.

OF Ordre de Fabrication.

OIV Opérateur d'Importance Vitale.

OMG *Object Modeling Groupe*.

OS système d'exploitation, *Operating System*.

OSI *Open Systems Interconnection*.

OSu/SuO Opération Support, *Support Operation*.

OSys/SysO Opération du Système, *System Operation*.

OT technologie des opérations, *Operation Technology*.

PC Partie Commande, *command part*.

PLC automate programmable industriel, *Programmable Logic Controler*.

PME Petite ou Moyenne Entreprises, *small medium entreprises*.

PO Partie Opérative, *operational part*.

PWM modulation de largeur d'impulsions, *Pulse Width Modulation*.

QoS qualité de service, *Quality of Services*.

RFID radio identification, *Radio Frequency IDentification*.

RLI Réseau Locale Industriel, *industrial local area network*.

RTU *Remote Terminal Unit*.

SA/AS Stockage Actif, *Active Storage*.

SAN réseau de stockage, *Storage Area Network*.

SAP Système Automatisé de Production.

SCADA système d'acquisition et de contrôle de données, *Supervisory Control And Data Acquisition*.

SCP/CPS Système Cyber-Physique, *Cyber Physical System*.

SdF Sûreté de Fonctionnement.

SED Système à Événement Discret.

SFC *Sequential Functioning Chart*.

SI Système d'Information, *information system*.

SIEM gestionnaire d'événement et d'information de sécurité, *Security Information and Event Manager*.

SSH *Secure SHell*.

SSL *Secure Sockets Layer*.

STC Signature Temporelle Causale, *causale temporal signature*.

TCP-IP *Open Transmission Control Protocol/Internet Protocol*.

TLS *Transport Layer Security*.

UML *Unified Modeling language*.

VLAN réseau local virtuel, *Virtual Local Area Network*.

VPN réseau privé virtuel, *Virtual Private Network*.

Glossaire

0-day qualificatif décrivant la nouveauté pour une faille de sécurité ou une attaque, elle n'a jamais été rencontrer auparavant donc la détection et la remédiation est ardue.

Center outil de management et de configuration ainsi que l'interface utilisateur de la solution Sentryo.

ComGEM outil de conception de PO et de génération pour la PC d'un système (principalement transitique).

ComSecGeM outil de conception de partie opérative et de génération pour la partie commande et pour la partie sécurité d'un système.

ENSIBS École National Supérieur d'Ingénieur de Bretagne Sud.

I4.0 acronyme utilisé par des constructeurs d'équipement pour dénommer le nouveau paradigme de l'industrie du futur.

Lab-STICC Laboratoire des Sciences et Techniques de l'Information de la Communication et de la Connaissance).

RAMI acronyme utilisé pour le modèle de référence pour l'architecture de l'industrie du futur.

SimSED outil de conception et de simulation pour la partie opérative d'un système (principalement transitique).

UBL Université Bretagne Loire -COMUE : Université région Bretagne et Université région Pays de Loire.

Wireshark outil de capture et d'analyse de communication réseau.