Convergence Sûreté-Sécurité des Systèmes de Contrôle Industriels : Appréciation du Risque de Cybersécurité pour la Sûreté des Systèmes

Mike Da Silva

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Convergence Sûreté-Sécurité des Systèmes de Contrôle Industriels : Appréciation du Risque de Cybersécurité pour la Sûreté des Systèmes

Safety-Security Convergence of Industrial Control Systems: Cybersecurity Risk Assessment for System Safety

Présentée par :

Mike DA SILVA

Direction de thèse :

Stéphane MOCANU MAITRE DE CONFERENCES HDR, Université Grenoble Alpes	Directeur de thèse
Maxime PUYS CEA	Co-encadrant de thèse
Pierre-Henri THEVENON CEA	Co-encadrant de thèse

Rapporteurs :

Nga NGUYEN
PROFESSEURE, ESILV - Ecole Supérieure d'Ingénieurs Léonard de Vinci
Vincent NICOMETTE
PROFESSEUR DES UNIVERSITES, INSA Toulouse

Thèse soutenue publiquement le **15 octobre 2024**, devant le jury composé de :

Stéphane MOCANU, MAITRE DE CONFERENCES, Université Grenoble Alpes	Directeur de thèse
Nga NGUYEN, PROFESSEURE, ESILV - Ecole Supérieure d'Ingénieurs Léonard de Vinci	Rapporteure
Vincent NICOMETTE, PROFESSEUR DES UNIVERSITES, INSA Toulouse	Rapporteur
Marie-Laure POTET, PROFESSEURE DES UNIVERSITES, Université Grenoble Alpes	Examinatrice
Pascal LAFOURCADE, PROFESSEUR DES UNIVERSITES, Université Clermont Auvergne	Examinateur

Invités :

Pierre-Henri THEVENON
INGENIEUR DOCTEUR, CEA
Maxime PUYS
MAITRE DE CONFERENCES, Université Clermont Auvergne

Résumé

Les Systèmes de Contrôle Industriels (SCI) sont conçus pour fournir un service, tel que la production d'électricité ou le traitement de l'eau, tout en protégeant les personnes, les biens et l'environnement contre les dangers liées aux procédés. Cependant, les SCI intègrent désormais des technologies de l'information (Information Technologies - IT) et sont interconnectés avec le monde extérieur, notamment à travers Internet, exposant ainsi leurs infrastructures à des cyberattaques. Les cyberattaques sont, donc, devenues de nouvelles menaces pour les opérations des systèmes industriels et, plus particulièrement, pour leur sûreté. Pour répondre à cette problématique, cette thèse présente une méthode d'appréciation complète du risque de cybersécurité pour la sûreté des SCI qui se différencie de l'état de l'art par son haut niveau d'automatisation et sa capacité à modéliser des SCI complexes. Cette méthode comprend trois étapes qui permettent respectivement : (1) d'Identifier les vulnérabilités des système de contrôle industriel, (2) d'Analyser les conséquences de ces vulnérabilités sur la sûreté du système et enfin (3) d'Évaluer l'impact du risque de cybersécurité sur la sûreté du système.

Dans la première étape de notre méthode, nous avons développé un processus automatisable de modélisation de la menace basé sur l'outil *Microsoft Threat Modeling Tool*. Les outils de modélisation de la menace, tel que *Microsoft Threat Modeling Tool*, sont couramment utilisées dans l'IT afin d'identifier les vulnérabilités des produits ou des systèmes, cependant ces outils manquent de représentativité lorsqu'il s'agit de modéliser des systèmes de contrôle industriels. Notre processus inclut donc une méthode qui permet de modéliser les composants des SCI dans l'outil *Microsoft Threat Modeling Tool*. Nous proposons aussi une méthode automatisable de création d'un modèle du système utilisable par *Microsoft Threat Modeling Tool* à partir du programme des API (automates programmables industriels). L'ensemble de ces deux contributions permettent d'automatiser complètement le flux de travail de l'outil *Microsoft Threat Modeling Tool* et de fournir un processus automatisable d'identification des vulnérabilités spécifiques aux SCI.

Dans la seconde étape de notre appréciation du risque, nous proposons une nouvelle méthode d'identification du risque de cybersécurité impactant la sûreté du système. Cette méthode met en relation les vulnérabilités identifiées dans la première étape avec leurs conséquences sur la sûreté du système. Cette relation est matérialisée au travers de scénarios d'attaque qui identifient les variables du système de contrôle qu'un attaquant peut manipuler pour compromettre la sûreté du système. Pour cela, nous avons développé une modélisation du comportement du système à partir des programmes de contrôle des API applicables à des systèmes complexes comprenant plus d'une vingtaine d'actionneurs et de capteurs tels que le procédé chimique de Tennessee-Eastman.

La dernière étape de notre méthode (étape 3) construit une matrice des risques qui classe les scénarios d'attaque générés dans la seconde étape en fonction de leur vraisemblance et de leur impact. La vraisemblance est déterminée selon un score d'exploitabilité des vulnérabilités qui composent le scénario d'attaque et l'impact est mesuré en fonction du danger engendré par les fonctions de sûreté compromises par le scénario d'attaque.

Enfin, cette matrice des risques est corrélée avec le risque tolérable par l'entreprise détentrice du SCI afin de déterminer les scénarios d'attaque qui nécessitent des contre-mesures complémentaires pour atteindre un niveau de risque tolérable.

Abstract

Industrial Control Systems (ICS) are designed to provide a service, such as power generation or water treatment, while protecting people, assets, and the environment against hazard. However, ICS now integrate Information Technology (IT) and are interconnected with the outside world such as the Internet, thereby exposing their infrastructures to cyberattacks. Cyberattacks have thus become new threats for industrial system operations and, more specifically, for their safety. To address this issue, this thesis presents a comprehensive cybersecurity risk assessment for the safety of ICS. This method differs from the state of the art in its high level of automation and its ability to model complex ICS. This method consists of three steps : (1) Identify industrial control system vulnerabilities, (2) Analyze the impact of these vulnerabilities on system safety, and (3) Assess the cybersecurity risk to system safety.

In the first step of our method, we developed an automatable threat modeling process based on the *Microsoft Threat Modeling Tool*. Threat modeling tools, such as the Microsoft Threat Modeling Tool, are commonly used in IT to identify vulnerabilities in products or systems, but they lack representativeness when it comes to modeling industrial control systems. Therefore, our process includes a method for modeling ICS components in the Microsoft Threat Modeling Tool. We also we propose an automatable method for creating a system model usable by the *Microsoft Threat Modeling Tool* from the PLC program. Together, these two contributions make it possible to fully automate the *Microsoft Threat Modeling Tool* workflow and provide an automatable process for identifying ICS-specific vulnerabilities.

In the second step of our risk assessment, we propose a new method for identifying the cybersecurity risk to system safety. This method relates the vulnerabilities identified in the first step to their consequences for system security. This relationship is materialized through attack scenarios that identify the variables in the control system that an attacker can manipulate to compromise system safety. To this end, we have developed a system behavior model based on PLC control programs that is applicable to complex systems comprising with more than twenty actuators and sensors, such as the Tennessee-Eastman chemical process.

The final step of our method (step 3) builds a risk matrix that ranks the attack scenarios generated in the second step of our method according to their likelihood and impact. The likelihood is determined based on an exploitability score for the vulnerabilities that make up the attack scenario, while the impact is measured in terms of the danger posed by the safety functions compromised by the attack scenario. Finally, this risk matrix is correlated with the risk tolerated by the ICS-owning organization, to determine which attack scenarios require additional countermeasures to achieve a tolerable level of risk.

Table des matières

Introduction

Dans ce chapitre introductif, nous proposons de présenter le contexte de cette thèse, les systèmes de contrôle industriels, la problématique qui a motivé ce travail et enfin nos contributions. Pour cela, dans une première section, nous présentons les systèmes industriels en commençant par introduire les contraintes spécifiques à ces systèmes. Pour répondre à ces contraintes, un système industriel comporte des équipements et des architectures spécifiques que nous présentons dans une seconde partie. Enfin, nous concluons cette présentation des systèmes de contrôle industriels en introduisant la dualité sûreté-(cyber)sécurité de ces systèmes. Ensuite, dans une seconde section, nous développons la problématique adressée dans cette thèse. Enfin, nous concluons cette introduction par une présentation des contributions de notre travail.

Les Systèmes de Contrôle Industriels

Les systèmes de contrôle industriel (Industrial Control Systems, ICS) sont à l'origine de la production de nos besoins quotidiens, tels que l'énergie électrique produite par les centrales électriques ou l'alimentaion produite par l'industrie agroalimentaire. L'ensemble des composants utilisés dans les systèmes industriels est désigné par le nom de "technologies opérationnelles" (Operational Technologies, OT). Cette expression, OT, a été, à l'origine, créée afin de mettre en évidence la différence entre les composants d'un système industriel et ceux des technologies de l'information (Informational Technologies, IT) utilisées au quotidien tel que les équipements réseaux qui structure l'Internet, les téléphones portables ou les ordinateurs personnels. Le NIST (National Institute of Standards and Technology), dans son guide pour la sécurité de l'OT [65], présente une vue d'ensemble de l'OT dont nous présentons la synthèse dans cette section afin d'introduire les systèmes de contrôle industriels.

Contraintes Spécifiques des Systèmes de Contrôle Industriel

Les systèmes de contrôle industriel ont pour objectif de fournir une production uniforme, économique et sûre en interagissant directement avec le procédé physique. Cette particularité des systèmes industriels implique des contraintes spécifiques à ces systèmes que nous proposons de présenter dans cette première partie.

La sûreté - Nous définissons la sûreté comme la capacité d'un système à protéger les biens, les personnes et l'environnement des dangers directement ou indirectement liés aux manipulations des équipements ou du procédé qu'il contrôle. Un système industriel doit être en mesure de détecter et de réduire les situations non sûres du système.

Les exigences de délais de contrôle - Le contrôle du procédé peut nécessiter que le système réduise la latence des communications afin d'être capable d'effectuer les actions de contrôle nécessaires en temps voulu. Par exemple, le protocole GOOSE défini dans la norme CEI 61850 [36] exige pour certaines de ces commandes un délai de transmission de bout-à-bout inférieur à 3 ms.

Répartition géographique - La répartition géographique des sites industriels, notamment pour les systèmes SCADA que nous détaillerons dans la prochaine partie, nécessite l'utilisation de réseaux informatiques étendus (Wide Area Network, WAN) tel que l'Internet pour centraliser l'information. Les communications critiques du système restent sur un réseau local (Local Area Network, LAN) au procédé contrôlé.

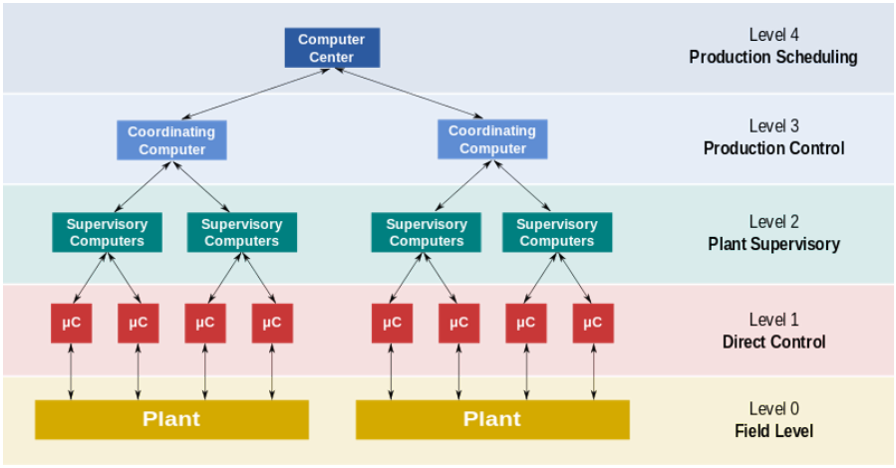


FIGURE 1 – Modèle de référence Purdue [58]

Contrôle centralisé - La supervision est utilisée pour centraliser et agréger les données provenant de plusieurs emplacements afin de fournir aux opérateurs une vue globale de l'ensemble du système. Cette vue d'ensemble aide les opérateurs à prendre des décisions de contrôle conformes à l'état du système. Le modèle de référence Purdue, présenté dans la Figure 1, représente l'architecture d'une entreprise en couche. Dans ce modèle en couches (ou niveaux), nous pouvons constater la centralisation de l'information qui remonte chacune des couches. À titre d'information, ce modèle est souvent utilisé pour montrer la frontière entre le système informatique de l'entreprise (IT, couches 3 et 4) et le système industriel (OT, couches 0 à 2). Dans le modèle Purdue, les systèmes industriels (systèmes OT) comprennent les niveaux 0 à 2 qui correspondent respectivement à :

- Niveau 0 (Niveau terrain) : l'ensemble des équipements physiques du système tels que les capteurs ou les actionneurs.
- Niveau 1 (Niveau contrôle) : l'ensemble des composants chargés du contrôle du procédé tels que les PLC, les RTU (Remote Terminal Unit) ou les IED (Intelligent Electonic Devices).

- Niveau 2 (Niveau supervision) : l'ensemble des composants chargés de centraliser et de mettre à la disposition des opérateurs les informations du système telles que les serveurs SCADA ou les serveurs d'historiques (Historian).

Les niveaux 3 et 4 font partie du système informatique de l'entreprise (IT) et ont un rôle de gestion et de planification de la production.

La disponibilité - Dans un système de contrôle industriel, la disponibilité est la propriété primordiale à assurer pour des raisons de sûreté. Ceci implique la nécessité d'une redondance plus marquée que dans les systèmes IT afin d'éviter les défaillances simultanées liées à un défaut de conception d'un équipement ou à une panne. Par exemple, la Figure 2 présente le fonctionnement du PRP (Parallel Redundancy Protocol) qui permet à l'émetteur d'une communication de dupliquer un même message vers un même expéditeur au travers de deux réseaux locaux différents. En cas de problème dans l'un des deux réseaux, le message arrive tout de même au destinataire par le second réseau.

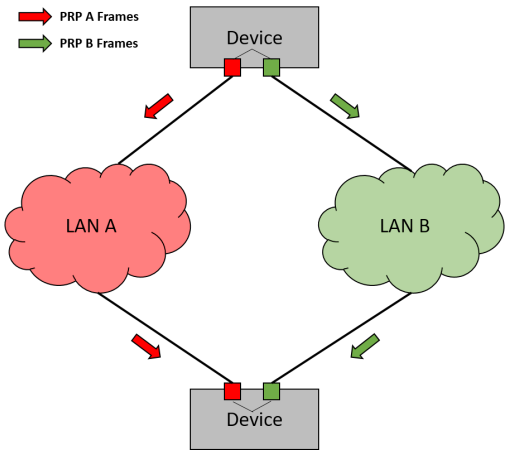


FIGURE 2 – Fonctionnement du PRP

La gestion des défaillances et des attaques - En lien avec la contrainte précédente, les systèmes industriels doivent être capables de poursuivre leurs opérations même en cas de défaillance ou d'attaque. Cela nécessite d'implémenter des mécanismes de redondance et de fonctionnement en état dégradé.

Nous venons de présenter les contraintes des systèmes de contrôle industriels. Afin de satisfaire ces contraintes, les ICS nécessitent des architectures et des composants spécifiques que nous présentons dans la partie suivante.

Architectures et Composants des Systèmes de Contrôle Industriel

Dans cette section, nous introduisons les SCADA (Supervisory Control and Data Acquisition) et les DCS (Distributed Control System) qui sont deux architectures de référence pour les systèmes de contrôle industriel.

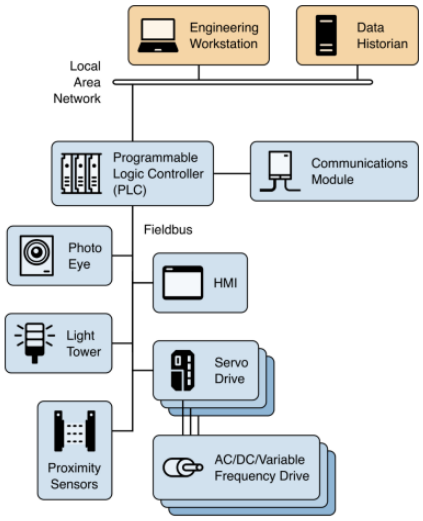


FIGURE 3 – Schéma d'une boucle de contrôle [65]

Dans ces deux architectures, la brique élémentaire est la boucle de contrôle telle que présentée dans la Figure 3. Du point de vue topologique, une boucle de contrôle est composée d'un contrôleur, un PLC dans notre cas, qui ajuste ses commandes de contrôle aux actionneurs en fonction des remontées d'information des capteurs. Ces informations (commandes d'actionneurs et mesures de capteurs) sont transmises au travers d'un réseau de communication, appelé bus de terrain, qui interconnecte les capteurs, les actionneurs et le PLC. Une boucle de contrôle comprend peu d'implication humaine et possède donc une IHM (Interface Homme-Machine) locale rudimentaire.

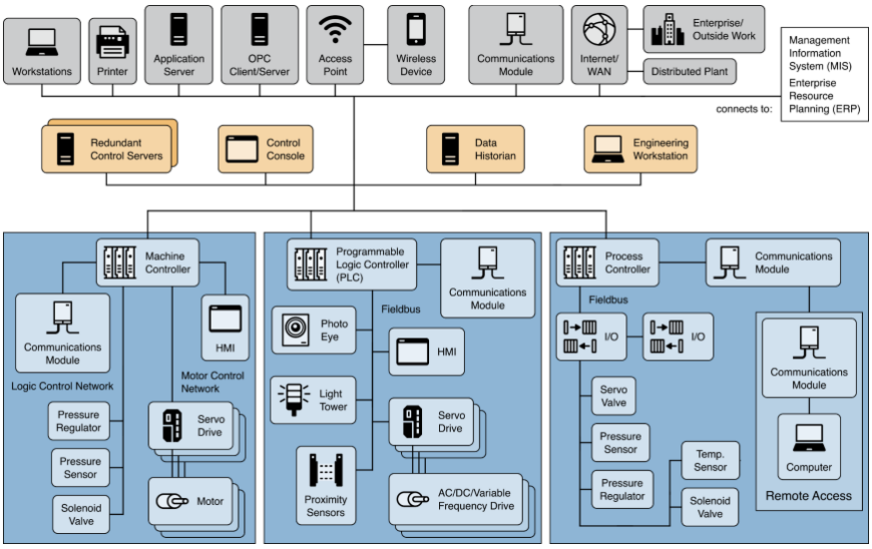


FIGURE 4 – Schéma d'une architecture DCS [65]

Distubuted Control System (DCS) - La première architecture de référence, présentée dans la Figure 4, est le DCS. L'objectif d'un DCS est d'assurer un contrôle distribué et une synchronisation des boucles de contrôle du système. Pour cela, les boucles de contrôle sont interconnéctés sur un réseau de contrôle. Ces architectures sont habituellement mises en œuvre à l'échelle d'une usine telle qu'une raffinerie de pétrole.

Supervisory Control and Data Acquisition (SCADA) - Enfin, les systèmes SCADA présentés dans la Figure 5 permettent une centralisation et une remontée des données en temps quasi réel. Ces infrastructures sont généralement déployées pour des systèmes comprenant des sites industriels géographiquement dispersés comme les réseaux ferroviaires, les réseaux de distribution d'eau ou les réseaux électriques. Dans un système SCADA, l'acquisition centralisée des données est tout aussi cruciale que le contrôle, ce qui implique l'existence d'une salle de contrôle équipée de communication longue distance comme l'Internet pour centraliser l'information des différents sites. Il est commun de trouver dans les systèmes SCADA des unités terminales distantes (Remote Terminal Unit, RTU) qui servent à la fois d'équipement de télémétrie et de contrôle distant du procédé.

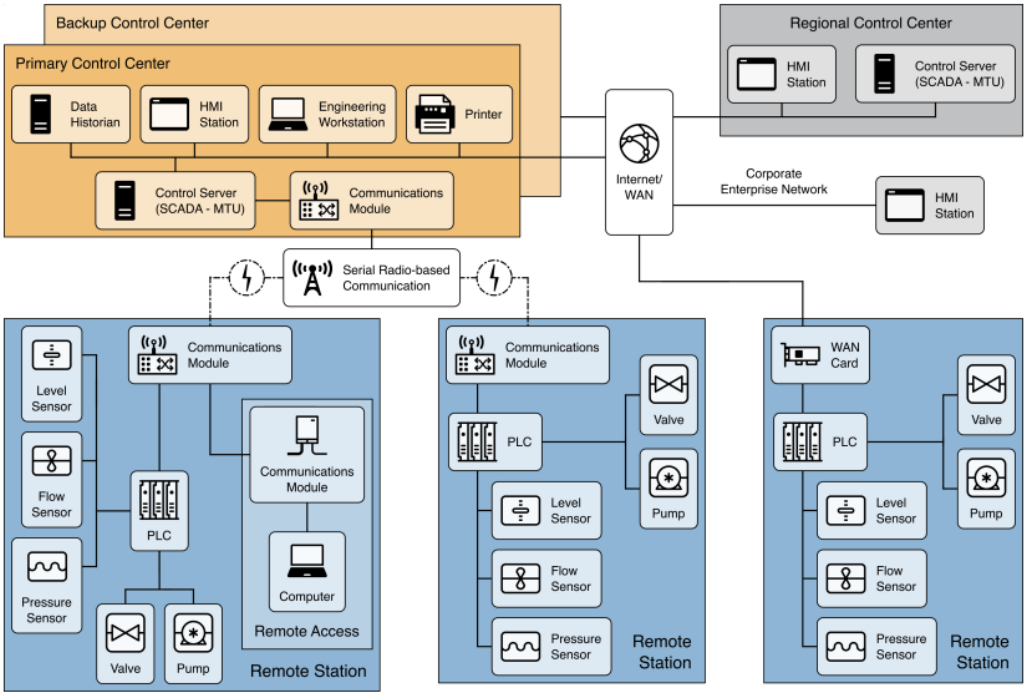


FIGURE 5 – Schéma d'une architecture SCADA [65]

À titre d'information, le terme SCADA est un terme relativement flou qui peut désigner un logiciel de supervision ou la taille d'un système industriel en fonction de la personne à qui nous nous adressons. Pour notre part, nous utilisons le terme SCADA ou système SCADA pour désigner l'architecture qui permet de centraliser et de remonter les données

du système en temps quasi réel. De plus, nous préférerons le terme serveur SCADA pour désigner le logiciel de supervision.

Sûreté et (Cyber)Sécurité

Les ICS ont toujours été soumis aux contraintes de sûreté, mais depuis récemment, les ICS sont de plus en plus interconnectés, informatisés et ouverts sur Internet, ce qui les expose aux cyberattaques. Cette nouvelle menace ouvre un nouveau paradigme où la sûreté des ICS est dépendante de sa cybersécurité, comme en témoignent ces cyberattaques [16] :

Attaque d'une centrale nucléaire Iranienne (2010) - Stuxnet [26, 48] est la première cyberattaque étatique connue ciblant un système de contrôle industriel. Ce vers informatique, découvert en 2010, comprenait 15 exploits et 4 0-day (vulnérabilité inédite) qui ciblait spécifiquement les centrales d'enrichissement d'uranium Iraniennes de Bushehr et de Natanz. L'attaque, supposée Américaine en partenariat avec Israël, a été réalisé en introduisant le vers informatique (Stuxnet) via une clé USB. Cette attaque a eu pour impact un retard de plusieurs mois à un an du programme nucléaire iraniens et plusieurs millions d'euros de dommage matériel.

Attaque du réseau électrique ukrainien (2015) - BlackEnergy [12, 66] est un logiciel malveillant dont la version la plus récente (BlackEnergy 3 - 2015) a servi à créer une panne d'électricité généralisée en Ukraine pendant 6 heures en décembre 2015. À la suite d'une campagne d'hameçonnage contre les distributeurs d'électricité, les attaquants ont pu charger le logiciel malveillant BlackEnergy 3 et mettre hors service 11 sous-stations électriques 110 kV et 23 sous-stations 35 kV, plongeant 225 000 foyers ukrainiens dans le noir en plein hiver. La provenance de certains éléments du logiciel tels que les botnets semble indiquer que l'attaque aurait été perpétrée par un groupe affilié aux renseignements russes dans un contexte d'annexion de la Crimée par la Russie.

Attaque du réseau d'assainissements des eaux usées (2000) - Un ancien employé vindicatif de la centrale des eaux usées de la Maroochy Shire [2] a profité de sa connaissance du système SCADA pour déverser 800 m^3 d'eaux usées dans des rivières et des parcs. À partir d'un équipement radio volé à son ancien employeur, l'attaquant a pu piloter à distance le système de contrôle industriel afin de déverser l'eau usée.

Ces attaques montrent la nécessité de concevoir des systèmes industriels qui sont à la fois sûrs et sécurisés. Pour rappel, nous utilisons le terme sûreté pour définir la capacité d'un système à protéger les biens, les personnes et l'environnement des dangers directement ou indirectement liés aux manipulations des équipements ou du procédé qu'il contrôle. Le terme sécurité définit la capacité d'un système à prévenir les interférences d'une menace informatique avec le fonctionnement correct du système de contrôle industriel [37]. Dans la suite de ce manuscrit, nous utiliserons les termes sécurité et cybersécurité de manière équivalente.

Dans cette première section, nous avons présenté les systèmes de contrôle industriel et plus particulièrement leurs contraintes, leurs architectures et leurs composants spéci-

fiques. Nous avons aussi introduit la dualité sûreté-sécurité de ces systèmes en présentant des exemples d'attaques qui exploitent les vulnérabilités informatiques d'un système de contrôle industriel pour impacter sa sûreté. Dans la prochaine section, nous approfondissons cette dualité sûreté-sécurité afin de présenter les problématiques que nous adressons dans cette thèse.

Problématique

Durant cette thèse, nous nous sommes intéressés à la notion de convergence sûreté-sécurité. La convergence est, selon le CISA (Cybersecurity and Infrastructure Security Agency), une collaboration formelle entre des fonctions de sécurité auparavant disjointes [15], ici la sûreté (sécurité physique) et la cybersécurité (sécurité informatique). Pour faire converger la sûreté et la sécurité, nous devons donc développer des méthodes et outils capables d'intégrer à la fois la sûreté et la sécurité, c'est-à-dire de prendre en compte leurs interdépendances dans le système [54] afin de concevoir des systèmes à la fois sûrs et sécurisés :

- **Dépendance conditionnelle** : le système peut comprendre une dépendance conditionnelle entre la sûreté et la cybersécurité, où le respect d'exigences de cybersécurité conditionne le niveau de sûreté et inversement. Par exemple, l'intégrité des commandes d'un contrôleur (exigence de cybersécurité) conditionne la mise en œuvre de ses fonctions de sûreté ou inversement la fiabilité d'un système de ventilation d'un centre de données conditionne la disponibilité de ses données.
- **Renforcement** : La sûreté et la sécurité peuvent avoir des relations de complémentarité tel que la redondance qui permet à la fois de résister à des cyberattaques par déni de service avec des mécanismes d'équilibrage de charge, mais assure aussi la continuité opérationnelle en cas de défaillance matérielle.
- **Antagoniste** : Le système peut aussi être confronté à une incompatibilité entre des contre-mesures de sûreté et de sécurité. Par exemple, pour protéger les personnes en cas d'urgence dans un bâtiment, la sûreté ouvrira toutes les portes pour laisser évacuer les personnes, là où la sécurité fermera toutes les portes pour éviter les intrusions [61].
- **Indépendance** : La sûreté et la sécurité peuvent aussi ne pas avoir d'influence l'une sur l'autre.

Ces différentes interdépendances entre la sûreté et la sécurité montrent qu'en traitant séparément la sûreté et la sécurité, on risque d'ignorer des interactions qui peuvent être bénéfiques ou critiques pour le système. Dans cette thèse, nous nous sommes interrogés sur ces interactions entre sûreté-sécurité et plus précisément sur la dépendance conditionnelle où la sûreté du système dépend de sa cybersécurité. Nos travaux de recherche se sont axés sur les trois questions suivantes :

- Comment **identifier** les menaces et vulnérabilités de cybersécurité spécifiques aux systèmes de contrôle industriel ?
- Comment **analyser** leurs conséquences sur la sûreté du système ?

- Comment **évaluer** le risque de cybersécurité pour la sûreté du système ?

Ces trois tâches : identifier, analyser et évaluer constituent les trois étapes d'une méthode d'appréciation du risque. Nous avons donc développé durant cette thèse une méthode d'appréciation du risque de cybersécurité pour la sûreté des systèmes de contrôle industriel.

Appréciation du Risque

Une appréciation du risque est un processus global d'identification, d'analyse et d'évaluation du risque [40] pour laquelle deux grandes approches se dégagent dans la littérature. La première propose une appréciation *unifiée* [46] du risque. Les méthodes unifiées [1, 47, 7, 45] vise à apprécier le risque de sûreté et de sécurité en utilisant un seul modèle tel que les arbres d'attaques et de défaillance qui représentent l'unification d'un arbre d'attaque (cybersécurité) et d'un arbre de défaillance (sûreté). Dans le cas des méthodes unifiées, nous pouvons distinguer plusieurs stratégies selon le modèle d'unification. La première stratégie unifie la sûreté et la sécurité dans un modèle historiquement utilisé par la sûreté. La deuxième stratégie est d'unifier la sûreté et la sécurité dans un modèle historiquement utilisé par la cybersécurité. Enfin, la troisième stratégie est d'unifier la sûreté et la sécurité dans un modèle ad-hoc. À l'inverse, les méthodes *intégrées* [46] séparent les appréciations initiales du risque de sûreté et de sécurité puis définissent, dans un second temps, les relations de cause à effet entre la sécurité et la sûreté.

Dans cette thèse, nous utilisons une méthode *intégrée* car ces méthodes permettent de placer la sûreté en tant que propriété prioritaire à assurer et donc la sécurité en tant que propriété au service de la sûreté. Selon nous, le couplage des modèles de sûreté et de sécurité dans les méthodes unifiées permet, certes, de mettre en évidence les conséquences et les effets en cascade sur le système, mais il accroît également la complexité de la modélisation (par exemple, en augmentant les possibilités combinatoires, le temps de traitement des résultats, etc.). De plus, alors que les appréciations du risque de sûreté sont généralement probabilistes en raison de la prévisibilité des défaillances, les appréciations du risque de cybersécurité utilisent la notion de vraisemblance par manque de données quantitatives pour réaliser des analyses probabilistes, ce qui ajoute une difficulté supplémentaire pour unifier les modèles de sécurité et de sûreté.

Contributions

Dans cette thèse, nous proposons une méthode automatisable du processus d'appréciation du risque ISO 3100 [40] (2018) appliquée aux risques de cybersécurité pour la sûreté des systèmes industriels. Cette méthode que nous proposons est complète, comprend un haut niveau d'automatisation et est applicable à des systèmes complexes (systèmes comprenant plus d'une dizaine de capteurs et d'actionneurs). Ces trois caractéristiques de notre méthode ont été évaluées selon les critères suivants :

Automatisation : Dans ce travail, nous présentons une vue d'ensemble de notre processus

d'appréciation du risque ainsi qu'une vue détaillée depuis laquelle nous présentons le niveau d'automatisation de notre méthode selon 3 niveaux d'automatisation :

1. *Manuel :* l'étape du processus nécessite une intervention externe
2. *Automatisable :* nous avons apporté les informations nécessaires pour qu'une personne initiée à la programmation puisse créer un code d'automatisation.
3. *Automatisé :* un code d'automatisation a été développé dans le cadre de cette thèse.

Nous verrons dans le Chapitre 1 que notre méthode est majoritairement composée d'étapes automatisées ou automatisables.

Couverture : Nous évaluons une méthode comme complète si elle intègre les trois étapes du processus d'appréciation du risque de l'ISO 31000, à savoir l'identification, l'analyse et l'évaluation du risque. Selon ce critère, la méthode que nous avons conçue est complète, contrairement à une majorité des articles de la littérature scientifique qui proposent des appréciations du risque sûreté-sécurité partielles.

Passage à l'échelle : Nous avons développé un modèle sûreté-sécurité pour l'identification du risque de cybersécurité pour la sûreté du système à partir de la logique des PLC. Ce modèle comporte une complexité exponentielle pour laquelle nous avons appliqué plusieurs optimisations afin qu'elle soit est applicable dans un temps raisonnable à des systèmes complexes tels que le procédé chimique Tennessee-Eastman qui comporte plus d'une vingtaine d'actionneurs et de capteurs. À ce jour, notre méthode est capable de modéliser des systèmes avec des sous-procédés comprenant jusqu'à 12 960 états, au-delà, nous atteignons les limites de mémoire vive de notre machine (16Go).

Contributions Spécifiques

Le processus d'appréciation que nous avons conçu comprend 3 étapes qui permettent de relever plusieurs challenges plus spécifiques à chacune des étapes de notre méthode.

Modélisation de la menace des systèmes industriels - Dans la première étape de notre méthode, présenté dans le Chapitre 2 de ce manuscrit, nous avons rendu possible la modélisation des systèmes de contrôle industriels dans un outil de modélisation de la menace. Ces outils de modélisation de la menace aident les experts en cybersécurité à identifier les vulnérabilités de leur système, mais manquent de représentativité lorsqu'il s'agit de modéliser des systèmes de contrôle industriels. Pour pallier ce manque, nous avons fourni un processus de modélisation afin de représenter les composants des ICS et d'intégrer leurs vulnérabilités connues (Common Vulnerabilities and Exposures, CVE) dans l'outil de modélisation de la menace. Nous avons, aussi, élaboré une méthode automatisable de construction d'un modèle du système utilisable par l'outil de modélisation de la menace dans le but d'automatiser l'intégralité du flux de travail de cet outil. Ces différentes contributions permettent de construire un processus automatisable d'identification des vulnérabilités des systèmes de contrôle industriels.

Identification des conséquences pour la sûreté - Dans la seconde étape de la méthode, nous avons conçu une nouvelle méthode d'identification des risques de cybersécurité pour la sûreté du système basée sur les programmes des PLC. Cette méthode, que nous décrivons dans le Chapitre 3, comprend un modèle sûreté-sécurité capable de passer à l'échelle sur des systèmes complexes. Dans ce travail, nous proposons une analyse détaillée de notre processus de modélisation afin de présenter les limites de notre méthode en temps et en espace. À ce jour, la taille maximale du système que nous pouvons modéliser est plafonnée à 12 960 états par sous-procédé du système, au-delà, nous atteignons les limites d'espace de stockage de la mémoire vive de notre machine (16 Go). Deuxièmement, nous proposons un modèle de menace sûreté-sécurité capable de mettre en relation les vulnérabilités identifiées dans la première partie de notre méthode avec leurs impacts sur la sûreté sous forme de scénarios d'attaque. Ces scénarios d'attaque identifient les vulnérabilités à exploiter et déterminent les valeurs de variables à fixer par un attaquant pour compromettre la sûreté du système.

Évaluation du risque de cybersécurité pour la sûreté - Dans la dernière étape (étape 3), détaillé dans la section 1.1.3 de ce manuscrit, nous construisons une matrice des risques qui classe chaque scénario d'attaque élaboré en étape 2 selon la vraisemblance d'exploitation des vulnérabilités qui composent le scénario d'attaque et de la sévérité du danger lié à la compromission de la sûreté. Cette matrice des risques est ensuite corrélée avec le risque tolérable par l'organisation détentrice de l'actif afin de déterminer les risques de cybersécurité (scénarios d'attaque) qui nécessite des remédiations complémentaires pour atteindre un niveau de risque conforme à la politique de l'organisation.

Publications

Ce travail a donné lieu à la publication de deux articles dans une conférence et à un brevet déposé en France et aux États-Unis :

Mike Da Silva, Maxime Puys, Pierre-Henri Thevenon, and Stephane Mocanu. 2023. PLC Logic-Based Cybersecurity Risks Identification for ICS. *In Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES '23). Association for Computing Machinery, New York, NY, USA, Article 117, 1–10. https ://doi.org/10.1145/3600160.3605067*

Mike Da Silva, Maxime Puys, Pierre-Henri Thevenon, Stephane Mocanu, and Nelson Nkawa. 2023. Automated ICS template for STRIDE Microsoft Threat Modeling Tool. *In Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES '23). Association for Computing Machinery, New York, NY, USA, Article 118, 1–7. https ://doi.org/10.1145/3600160.3605068*

Mike Da Silva, Maxime Puys, Pierre-Henri Thevenon, Stephane Mocanu. 2024. Brevet : FR3144328 / US20240211607A1 - Procédé et dispositif d'identification des risques de cyberattaques / Method and device for identifying risks of cyberattacks.

Soumis au Journal COSE (en révision) : **Mike Da Silva, Stephane Mocanu, Maxime Puys, and Pierre-Henri Thevenon. 2024. Safety-Security Convergence : Automation of IEC 62443-3-2**

1 Automatisation de l'Appréciation du Risque de Cybersécurité pour la Sûreté des Systèmes de Contrôle Industriels

Les appréciations du risque de cybersécurité des systèmes industriels sont généralement réalisées à partir de processus génériques définis par des normes telles que la CEI 62443-3-2 [38]. Ces processus sont fréquemment appliqués de manière ad hoc selon une succession de tâches manuelles, sujettes aux erreurs humaines, dont la qualité dépend essentiellement de l'expertise de l'équipe. Afin de réduire les efforts manuels nécessaires à l'application des processus définis dans les normes de cybersécurité, nous avons développé une méthode qui automatise une appréciation du risque ISO 31000 [40] appliquée à la cybersécurité des ICS. La norme ISO 31000 [40] fournit une appréciation du risque générique applicable à toute organisation, indépendamment de son secteur, de son industrie ou de la forme de risque à laquelle elle est confrontée. Ainsi, nous proposons une méthode suffisamment générique pour s'intégrer et partiellement automatiser des processus d'appréciations du risque comme celui de la norme CEI 62443-3-2.

La méthode que nous avons conçue relève trois challenges. Le premier challenge est de proposer une méthode d'appréciation du risque sûreté-sécurité qui passe à l'échelle sur des systèmes larges. Ce challenge est traité dans une section dédiée (section 3.1.3) dans le Chapitre 3. Le second challenge a été d'automatiser un processus de modélisation de la menace capable d'identifier les vulnérabilités spécifiques aux systèmes industriels. Enfin, le troisième challenge a été de concevoir une appréciation du risque globale et automatisée. La globalité de notre méthode est discutée comparativement à l'état de l'art dans la section 3.4 du Chapitre 3 et l'automatisation est présentée dans ce chapitre.

Aperçu - L'automatisation est une contribution transverse à l'ensemble du développement de notre méthode d'appréciation du risque sûreté-sécurité. Nous proposons donc une présentation générale de la méthode que nous avons conçue (section 1.1) afin de mettre en avant son automatisation. Puis, dans la section 1.2, nous montrons aussi que notre méthode est applicable au processus d'évaluation des risques de la norme CEI 62443-3-2 et permet de l'automatiser partiellement, ce qui réduit les efforts manuels nécessaires à son application et augmente la cohérence des résultats. Enfin, nous concluons ce chapitre dans la section 1.3 en rappelant les contributions liées à notre méthode avec une attention particulière sur l'automatisation de notre méthode.

1.1 Vue d'ensemble de la méthode

Notre méthode automatise une appréciation du risque définie par la norme ISO 31000 [40] appliquée aux risques de cybersécurité pour la sûreté des systèmes industriels. Le processus d'appréciation du risque ISO 31000 [40] comprend trois étapes :

1. **Identification du risque :** L'identification du risque a pour objectif de rechercher, reconnaître et décrire le risque. Dans notre méthode, nous identifions le risque de cybersécurité qui a un impact sur la sûreté du système au travers de scénarios d'attaque.
2. **Analyse du risque :** L'analyse fournit les informations nécessaires afin d'évaluer le risque. Les scénarios que nous identifions sont classés dans une matrice de risques en fonction de leur vraisemblance et de la sévérité de leur impact sur la sûreté.
3. **Évaluation du risque :** L'évaluation du risque compare les résultats de l'analyse du risque avec le niveau de risque tolérable pour l'organisation afin de déterminer si le risque doit être traité ou non. En fonction du risque tolérable par l'entreprise, nous déterminons les scénarios d'attaque qui nécessitent un traitement complémentaire.

Notre méthode, dont une vue d'ensemble est présentée dans la Figure 1.1, vise à construire des scénarios de cyberattaque qui mettent en lien des vulnérabilités de cybersécurité avec leurs conséquences sur la sûreté du système. Nous générons et évaluons ces scénarios d'attaque au travers d'une méthode en trois parties.

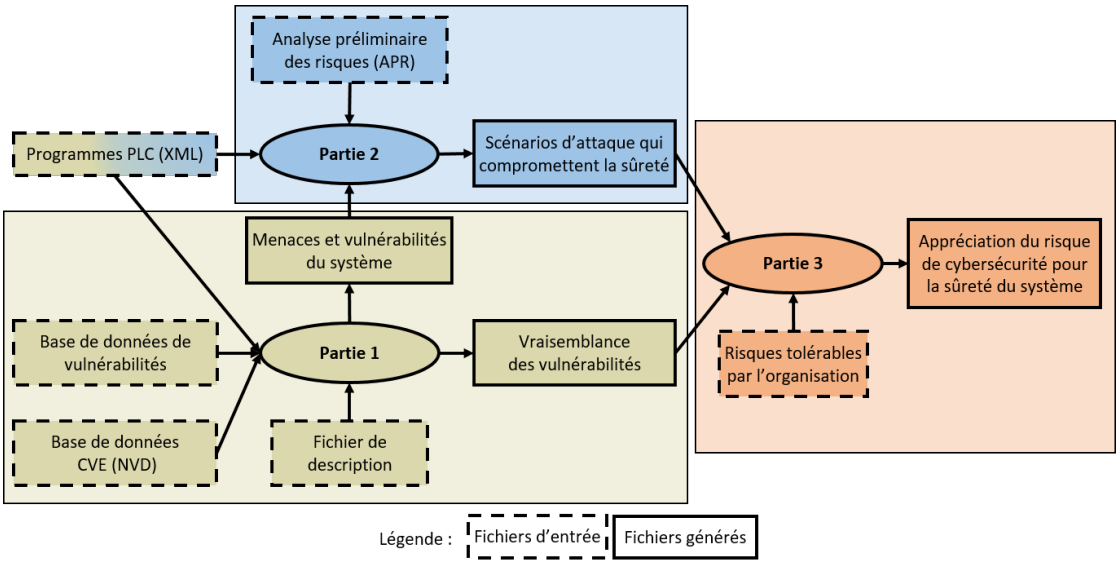


FIGURE 1.1 – Vue d'ensemble de la méthode

Partie 1 – Modélisation de la menace et attribution de la vraisemblance : Cette première partie (Partie 1 de la Figure 1.1) vise à identifier les vulnérabilités de cybersécurité du système à l'aide de l'outil de modélisation de la menace *Microsoft Threat Modeling Tool* (MTMT). MTMT implémente une énumération automatique des vulnérabilités du système à partir d'un modèle du système et d'un template que nous construisons auto-

matiquement dans notre méthode :

- Nous construisons le modèle du système à partir des programmes des automates programmables au format XML (cf. *Programmes PLC (XML)* Figure 1.1) et d'un fichier de description des composants à modéliser que nous détaillerons plus tard ;
- Un *template* est une bibliothèque de composants et de vulnérabilités que nous construisons à partir d'une base de données de vulnérabilité alimentée par l'organisation en charge de l'appréciation du risque et d'une base de données de vulnérabilités publiques (CVE - Common Vulnerabilities and Exposures) des composants du système. Dans ce travail, nous utilisons la base de données de CVE *NVD* (National Vulnerability Database) du NIST.

Les vulnérabilités énumérées par MTMT sont classées par menaces pour lesquels nous attribuons un score de vraisemblance. Nous déterminons, donc, les menaces et vulnérabilités du système qui sont utilisées dans la Partie 2 de notre méthode (introduite dans ce chapitre puis détaillée dans le Chapitre 3) ainsi que la vraisemblance des vulnérabilités qui sera utilisée dans la troisième partie de notre méthode (Partie 3 de la Figure 1.1).

Partie 2 – Identification des conséquences sur la sûreté : La seconde partie de notre méthode (Partie 2 de la Figure 1.1) construit des scénarios d'attaque qui compromettent la sûreté du système. Un scénario d'attaque met en relation les vulnérabilités du système, identifiées dans la Partie 1 de notre méthode, avec leurs conséquences sur la sûreté que nous modélisons sous la forme de fonctions implémentées dans le programme des PLC (cf. *Programmes PLC (XML)* Figure 1.1). Nous identifions les fonctions de sûreté assurées par le programme des PLC à partir d'une Analyse Préliminaire des Risques (APR) réalisée préalablement par un expert en sûreté. Cette APR permet aussi de déterminer la sévérité du danger liée à la compromission d'une fonction de sûreté par un scénario d'attaque.

Partie 3 - Évaluation du risque : Cette dernière partie (Partie 3 de la Figure 1.1) classe les scénarios d'attaque dans une matrice de risques et évalue selon les risques tolérables par l'organisation les scénarios d'attaque qui nécessitent ou non un traitement complémentaire. Nous construisons la matrice de risques selon la vraisemblance des vulnérabilités (cf. Partie 1 de la Figure 1.1) à exploiter pour réaliser un scénario d'attaque et la sévérité du danger liée à la compromission de la fonction de sûreté par le scénario d'attaque (cf. Partie 2 de la Figure 1.1).

Cette description macroscopique permet de présenter pour chacune des parties les informations d'entrées et de sorties, leurs objectifs ainsi que leur articulation avec les autres parties de la méthode. Dans la suite de cette section, nous détaillons le processus de chaque partie afin de présenter le niveau d'automatisation de notre méthode. Les détails techniques d'implémentation de la Partie 1 et de la Partie 2 seront traités dans leur chapitre respectif, à savoir le Chapitre 2 pour la Partie 1 et le Chapitre 3 pour la Partie 2.

Dans ce manuscrit, nous détaillons le niveau d'automatisation de chacun des processus de notre méthode et définissons 3 niveaux d'automatisation : manuel, automatisable et automatisé. Nous distinguons **automatisé** et **automatisable** selon si nous avons développé un code d'automatisation (automatisé) ou non (automatisable).

1.1.1 Partie 1 : Modélisation de la menace et attribution de la vraisemblance

La modélisation de la menace est une méthode qui analyse une représentation d'un système, d'un logiciel, d'une application, etc. afin de mettre en évidence les problématiques de cybersécurité [9]. Cela permet de comprendre les menaces et d'identifier les vulnérabilités qui pèsent sur le système [20]. Dans cette première partie de notre méthode (Partie 1), nous étendons le processus d'identification des vulnérabilités fourni par l'outil de modélisation de la menace *Microsoft Threat Modeling Tool* (MTMT) pour :

1. Modéliser les systèmes industriels et identifier leurs vulnérabilités spécifiques dans MTMT ; et
2. Automatiser l'intégralité du processus d'identification des vulnérabilités de MTMT

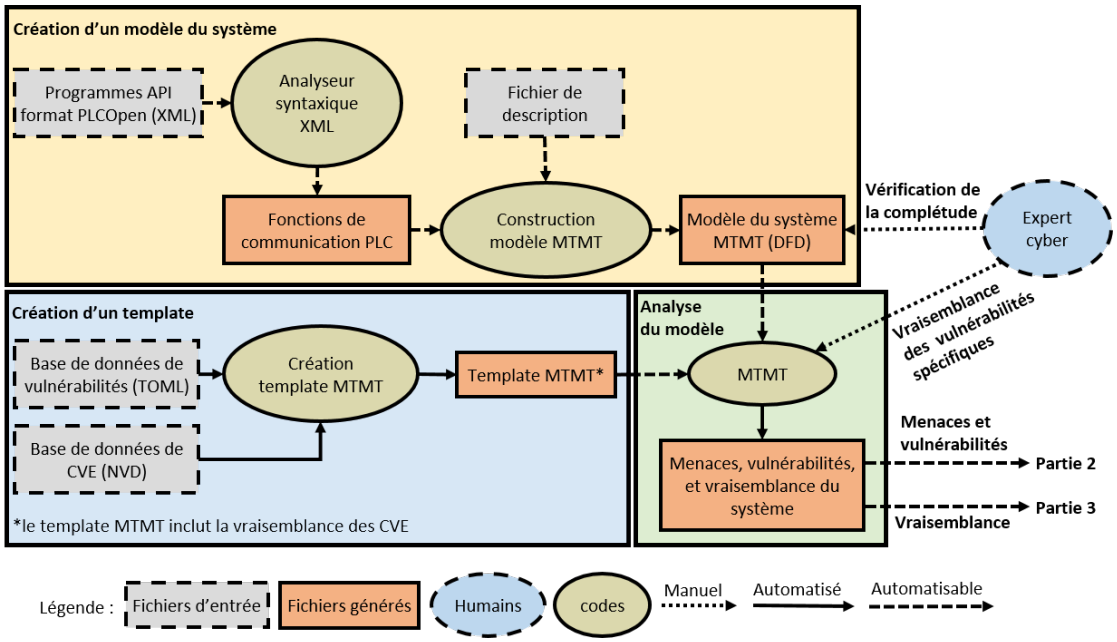


FIGURE 1.2 – Processus de la Partie 1

L'identification des vulnérabilités d'un système dans *Microsoft Threat Modeling Tool* (MTMT) se fait en trois étapes : (1) Création d'un template (bibliothèque de composants et de vulnérabilités), (2) Création d'un modèle du système et (3) Analyse du modèle. Dans MTMT, l'analyse du modèle, i.e. une énumération des vulnérabilités du template qui sont applicables au système modélisé, est automatisée. Pour automatiser le processus d'identification des vulnérabilités de MTMT, nous devons donc créer automatiquement un template et un modèle du système. La Figure 1.2 présente le processus détaillé de la première partie de notre méthode. Cette figure est découpée en trois afin de représenter chacune des étapes du processus d'identification des vulnérabilités de MTMT, à savoir la création du template en bas à gauche de la figure, la création du modèle du système en haut et enfin l'analyse du modèle (automatisée dans MTMT) en bas à droite de la Figure 1.2.

Création d'un template - Par défaut, MTMT fournit plusieurs templates et propose une interface pour améliorer ou créer son propre template. Cependant, il n'existe pas de template pour analyser les systèmes industriels. Nous avons donc développé un programme permettant de générer des templates spécifiques aux ICS à partir d'une base de données de vulnérabilités personnalisées et de la base de données de CVE NVD [1] (National Vulnerability Database) fournie par le NIST. Les CVE sont des descriptions publiques relatives à une vulnérabilité de cybersécurité d'une application, d'un équipement ou d'un système d'exploitation spécifique.

Création d'un modèle du système - La représentation du système dans MTMT est réalisée selon un diagramme de flux de données (DFD) qui modélise graphiquement les composants et les flux échangés. Notre méthode propose d'automatiser la création de ce DFD pour les systèmes industriels à partir du programme des PLC. Pour automatiser la construction du modèle, nous utilisons un fichier XML [17] normalisé par la CEI 61131 [33], connu sous le nom de PLCopen [55], qui décrit le contenu du programme d'un PLC. Ce programme contient, entre autres, une description des fonctions de communication de l'automate à partir de laquelle nous sommes en mesure de construire le DFD du système (*Modèle du système MTMT* - Figure 1.2). Le fichier de description, détaillé plus tard dans la section 2.3.2 du Chapitre 2, permet notamment de faire le lien entre les composants du système et leur représentation dans MTMT. Un exemple d'un modèle du système (DFD) dans MTMT est fournit dans la Figure 1.3.

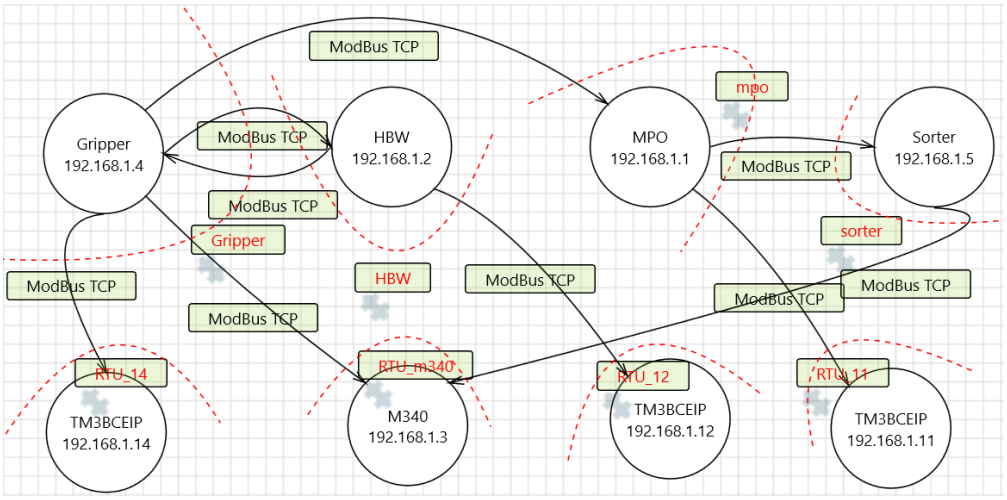


FIGURE 1.3 – Exemple d'un modèle du système (DFD) dans MTMT

Enfin, nous assignons automatiquement dans MTMT une vraisemblance à chaque CVE identifiée. Cette vraisemblance est déterminée à partir du score d'exploitabilité défini par le système de notation CVSS v3.1 (Common Vulnerability Scoring System). Dans la NVD (base de données de CVE), chaque CVE a son score d'exploitabilité déjà calculé. Pour les vulnérabilités personnalisées, pour lesquelles il n'est pas possible d'automatiser

1. https ://nvd.nist.gov/vuln

l'attribution de la vraisemblance, un expert en cybersécurité assigne une vraisemblance aux vulnérabilités en calculant leur score d'exploitabilité CVSS v3.1. L'utilisation de ce score d'exploitabilité permet d'harmoniser les notations entre les CVE et les vulnérabilités personnalisées.

Cette première partie de notre méthode met à disposition un fichier qui énumère les vulnérabilités du système au format CSV (choix imposé par MTMT) ainsi que les menaces qu'elles réalisent et leur vraisemblance d'exploitation. Ce fichier sera utilisé à la fois par la seconde partie de la méthode (Partie 2) pour identifier les conséquences de l'exploitation des vulnérabilités sur la sûreté du système (indépendamment de leur vraisemblance) ; et par la troisième partie (Partie 3) pour évaluer le risque de cybersécurité pour la sûreté du système.

1.1.2 Partie 2 : Identification des conséquences sur la sûreté

La deuxième partie de notre méthode (Partie 2, détaillé dans le Chapitre 3) comprend un processus d'identification du risque de cybersécurité pour la sûreté du système. Ce processus, présenté dans la Figure 1.4, permet de générer des scénarios de cyberattaque qui, par l'intermédiaire de manipulation des relevés de capteurs et des commandes d'actionneurs, compromettent les fonctions de sûreté du système. Les scénarios d'attaque ainsi générés sont mis en correspondance avec les vulnérabilités du système, identifiées dans la Partie 1 (cf. section 1.2). Cette correspondance permet de conserver uniquement les scénarios d'attaque réalisables et de mettre en évidence les vulnérabilités de cybersécurité ayant un impact sur la sûreté. Cette approche comprend trois étapes séquentielles que nous allons présenter.

Étape 1 - Dans la première étape, le système est modélisé en convertissant les programmes de contrôle des PLC au format PLCopen (XML) en automates finis (*Modèle du système (Automate)* - Figure 1.4). Habituellement, les états d'un automate sont représentés par un cercle et les transitions par des flèches reliant deux états. Dans notre modélisation, une transition comporte deux informations : (1) l'événement du système qui déclenche la transition, par exemple une cuve qui devient pleine, et (2) les sorties émises, dans notre cas les commandes émises par le PLC lors du changement d'état (transition). Ce modèle mathématique a l'avantage d'expliciter le comportement des programmes des PLC en représentant tous les états et tous les événements du système exprimés directement ou indirectement par les programmes.

Afin d'éviter la confusion entre **automate programmable** et **automate fini**, nous parlerons dans le reste de ce chapitre de PLC pour l'automate programmable (l'équipement physique) et d'automate pour le modèle mathématique.

Étape 2 - Nous déterminons les fonctions de sûreté que le système doit garantir en les dérivant des dangers identifiés par une analyse préliminaire des risques (APR). Nous considérons l'APR comme une entrée de la méthode et qu'elle doit être réalisée par un expert en sûreté. Nous modélisons les fonctions de sûreté par une implication logique entre un état dangereux du système et la commande permettant de sortir de l'état dangereux. Les fonctions de sûreté sont ensuite modélisées dans le modèle du système (*Modèle du système (Automate)* – Figure 1.4).

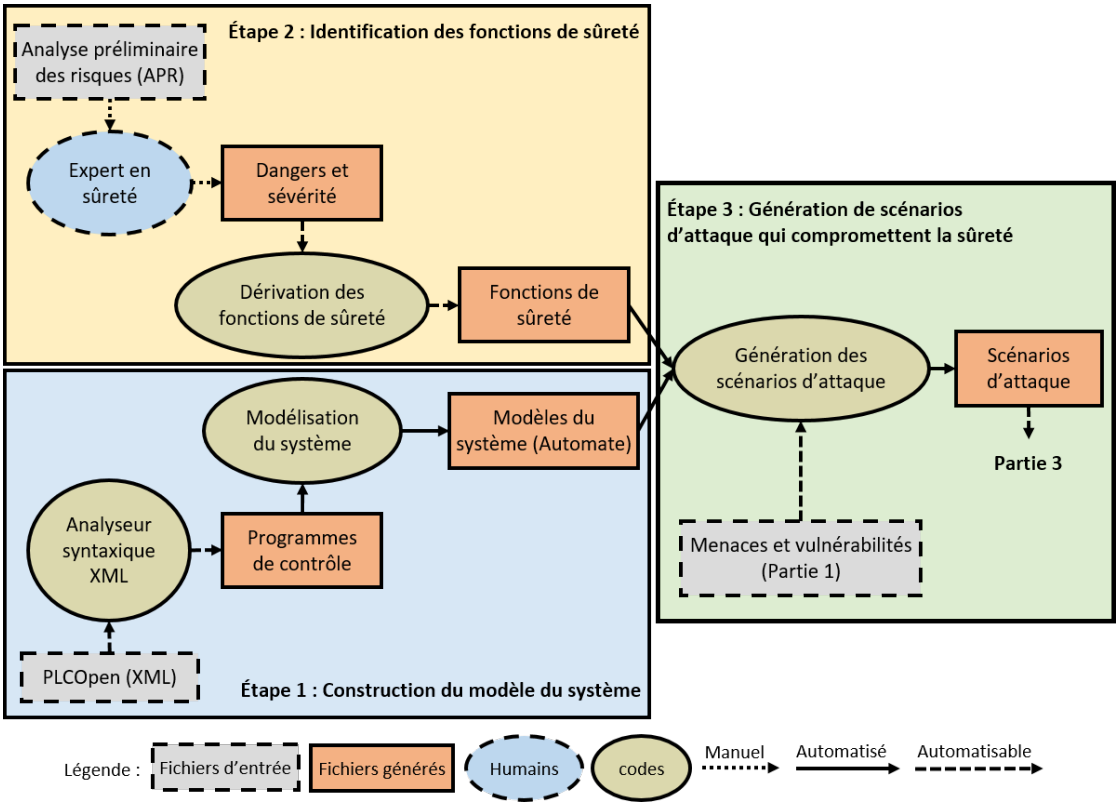


FIGURE 1.4 – Processus de la Partie 2

Étape 3 - Enfin, nous générons des scénarios d'attaque qui compromettent les fonctions de sûreté. Un scénario d'attaque est une manipulation de données de capteurs ou d'actionneurs qui permet soit de :

- Bloquer un événement (la cuve devient pleine) ou une commande (fermeture des vannes de remplissage) afin d'empêcher le système de réagir à la situation de danger ; soit de
- Forcer un événement ou une commande, à savoir de fournir une information erronée au PLC afin de forcer une réaction inadaptée à la situation ou, simplement, d'envoyer une commande illégitime aux actionneurs.

Par exemple, dans un scénario d'attaque par blocage, l'attaquant peut empêcher la détection d'un événement *La cuve devient pleine* afin d'inhiber la réaction de fermeture des vannes de remplissage et ainsi faire déborder la cuve ; sinon, l'attaquant peut bloquer la commande de fermeture des vannes de remplissage. Dans un scénario par forçage, un attaquant peut, lorsque la cuve est pleine, forcer un ensemble de valeurs de variables dans le but de fausser l'état du système ce qui entraîne une réouverture des vannes de remplissage et fait déborder la cuve ; ou envoyer directement une commande d'ouverture aux vannes de remplissage.

Cette méthode s'appuie sur trois fichiers d'entrées comprenant respectivement : les programmes PLC pour construire le modèle du système (Étape 1), l'analyse préliminaire des risques qui détermine les dangers du système et que nous déclinons en fonctions de sûreté garanties par le système (Étape 2), et les menaces et vulnérabilités du système pour générer les scénarios d'attaque (Étape 3).

1.1.3 Partie 3 : Évaluation du risque

La dernière étape de notre méthode (Partie 3 – Figure 1.1) propose une évaluation du risque de cybersécurité pour la sûreté du système. Une évaluation du risque détermine si le niveau du risque est tolérable par l'organisation. Ainsi, afin d'évaluer le risque, nous devons déterminer le niveau du risque ainsi que le risque tolérable par l'organisation, comme présenté dans la Figure 1.5.

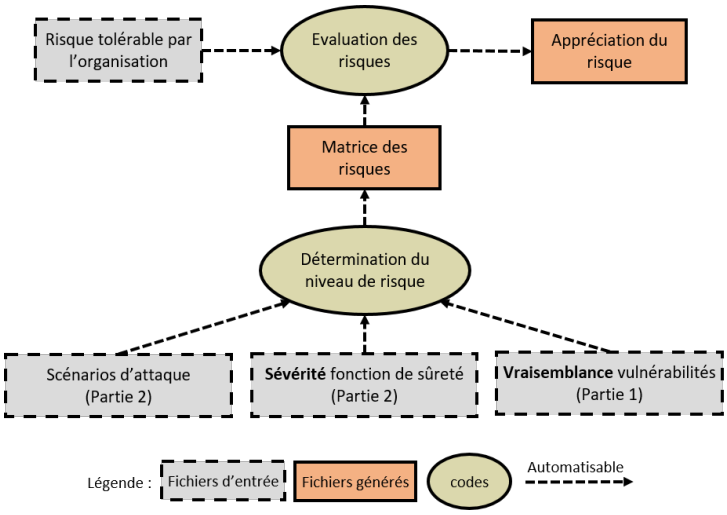


FIGURE 1.5 – Processus de la Partie 3

Généralement, le niveau de risque est déterminé grâce à une matrice de risques qui classe le risque en fonction de sa vraisemblance et de sa sévérité. Dans notre méthode, le risque est représenté par des scénarios d'attaque qui compromettent la sûreté du système. Nous définissons la vraisemblance d'un scénario d'attaque en fonction de la vraisemblance d'exploitabilité des vulnérabilités qui composent le scénario. Pour rappel, la vraisemblance d'exploitabilité des vulnérabilités a été déterminé dans la Partie 1 de la méthode (cf. section 1.1.1). Plus précisément, nous établissons l'échelle d'évaluation de la vraisemblance en fonction du score d'exploitabilité CVSS v3.1 (Common Vulnerability Scoring System). Le score d'exploitabilité CVSS v3.1, allant de 0,1 à 3,9, est une partie du score globale qui lui s'étend de 0,1 à 10,0. Dans ce travail, nous nous intéressons à l'assignation d'une vraisemblance aux vulnérabilités alors que le score global CVSS v3.1 cherche à quantifier la gravité d'une vulnérabilité (vraisemblance et sévérité). C'est pour cette raison que nous nous préoccupons uniquement du score d'exploitabilité des vulnérabilités (vraisemblance). Le score global CVSS v3.1 propose une échelle d'éva-

luation qualitative de la gravité des vulnérabilités : $Aucun = 0$, $Faible = 0,1 - 3,9$, $Moyen = 4,0 - 6,9$, $Haut = 7,0 - 8,9$, $Critique = 9,0 - 10,0$. Nous appliquons cette échelle d'évaluation au score d'exploitabilité [2] proportionnellement à sa plage de valeurs grâce à un produit en croix tel que présenté dans l'équation 1.1.

$$Score = \frac{\text{exploitabilité} \times 10}{3,9} \tag{1.1}$$

Exemple :

$$Score = \frac{2 \times 10}{3,9} = 5,13 \implies \text{Vraisemblance Moyenne} \tag{1.2}$$

Ensuite, nous déterminons la sévérité d'un scénario d'attaque selon la gravité du danger, identifiée par l'analyse préliminaire du risque (APR), liée à la fonction de sûreté compromise par le scénario. Ainsi, à partir de la vraisemblance et de la sévérité des scénarios d'attaque, nous pouvons automatiquement construire une matrice de risques qui classe les scénarios d'attaque en fonction de leur niveau de risque (sévérité et vraisemblance).

Afin d'évaluer le risque, l'organisation doit quantifier le niveau du risque qu'elle peut tolérer (couple vraisemblance/sévérité). Le niveau de risque de chaque cellule d'une matrice de risques est couramment évalué qualitativement afin que l'organisation puisse définir quel risque elle tolère ou non. Par exemple, la matrice de risques présentée dans la Figure 1.6 quantifie le niveau de risque de Faible à Extrême (Faible, Modéré, Élevé et Extrême). Si l'organisation définie que, par exemple, le risque tolérable sont les scénarios d'attaque comprenant un niveau de risque faible ou modéré, alors les scénarios d'attaque qui dépassent le risque tolérable (risques élevé et extrême) devront faire l'objet d'une remédiation (en dehors du périmètre de cette thèse) afin de réduire le risque à un niveau tolérable par l'organisation.

		Sévérité				
		A	B	C	D	E
Vraisemblance	Critique (9.0 - 10.0)	Modéré	Élevé	Élevé	Extrême	Extrême
	Haut (7.0 - 8.9)	Faible	Modéré	Élevé	Extrême	Extrême
	Moyen (4.0 - 6.9)	Faible	Faible	Modéré	Élevé	Extrême
	Faible (0.1 - 3.9)	Faible	Faible	Modéré	Élevé	Élevé

FIGURE 1.6 – Exemple d'une matrice de risques

2. Nous n'incluons pas l'évaluation *Aucune* (score égal à zéro) car le score d'exploitabilité est strictement supérieur à zéro selon son équation

Dans cette section, nous avons présenté une vue d'ensemble du processus de notre méthode d'appréciation du risque sûreté-sécurité. Dans chacune des trois parties de notre méthode, nous avons détaillé leur processus afin de montrer le niveau d'automatisation de notre méthode basée sur la norme ISO 31000. Dans la prochaine section, nous présentons de quelle manière cette méthode peut être appliquée à la norme d'appréciation du risque de cybersécurité CEI 62443-3-2 afin d'automatiser partiellement ce processus. Cette automatisation améliore la cohérence des résultats, réduit les efforts manuels et limite les erreurs humaines.

1.2 Automatisation de la CEI 62443-3-2

Dans cette section, nous introduisons la norme CEI 62443-3-2 et nous présentons son processus d'évaluation des risques de cybersécurité pour les systèmes d'automatisation et de commande industriels (industrial automation and control system(s) — IACS). Ensuite, nous présentons l'applicabilité de notre méthode à ce processus d'évaluation. Enfin, nous comparons notre travail avec l'état de l'art en matière d'automatisation de l'évaluation des risques CEI 62443-3-2.

Le terme évaluation du(es) risque(s) n'a pas le même sens dans la CEI 62443-3-2 et l'ISO 31000. Dans la CEI 62443-3-2 l'**évaluation des risques** fait référence au **processus global** de la méthode ; et dans l'ISO 31000 l'**évaluation du risque** fait référence à la troisième étape de l'appréciation du risque qui consiste à comparer les résultats de l'analyse de risque (sévérité et vraisemblance du risque) avec le risque tolérable par l'organisation [40].

1.2.1 Présentation de la norme CEI 62443-3-2

La CEI 62443 est une série de normes sur la *Sécurité des systèmes d'automatisation et de commande industriels* (industrial automation and control system(s) — IACS). Les normes appartenant à cette série sont réparties en quatre groupes : (1) Généralités, (2) Politiques et procédures, (3) Système, et (4) Composant. La CEI 62443-3-2, appartenant donc aux normes sur les systèmes (groupe 3), propose un processus récent (2020) d'évaluation des risques de sécurité pour la conception de systèmes IACS largement utilisé dans le domaine de la cybersécurité industrielle. Cette évaluation, présentée dans la Figure 1.7, est constituée de 7 étapes dénommées exigences de zone et conduit (zone and conduit requirements – ZCR) dans la norme.

La première exigence propose d'identifier le système à l'étude (ZCR 1) au travers d'une architecture du système et d'un inventaire des actifs (logiciels et matériels). Ensuite, une appréciation initiale du risque de cybersécurité (ZCR 2) est accomplie pour identifier le risque de cybersécurité non atténué le plus défavorable. Le système est ensuite partitionné en zones et conduits (ZCR 3), c'est-à-dire en groupes d'actifs physiques (zones) et de canaux de communication qui relient les différentes zones (conduits). Ces zones et conduits permettent de regrouper les actifs du système qui partagent des exigences de cybersécurité communes. Si le risque initial, déterminé par la ZCR 2, excède le risque tolérable par l'entreprise (ZCR 4), alors une appréciation détaillée du risque est accomplie

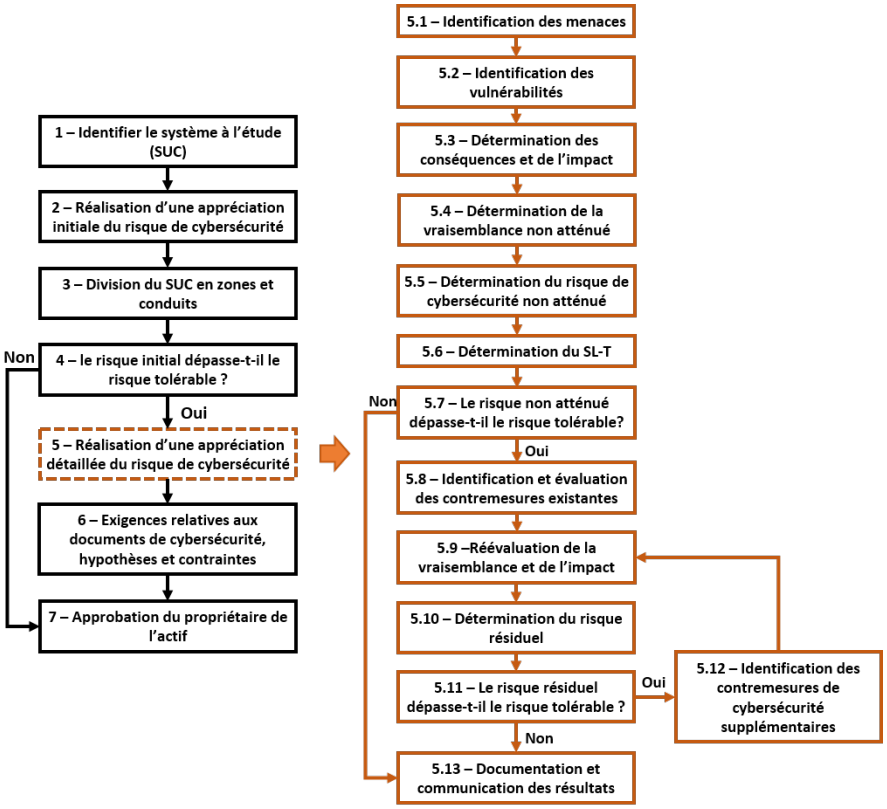


FIGURE 1.7 – Processus d'évaluation des risques CEI 62443-3-2

pour chaque zone et conduit (ZCR 5). La spécification des exigences de cybersécurité est ensuite rédigée (ZCR 6) et, enfin, l'évaluation du risque doit être approuvée par le propriétaire du système (ZCR 7).

Chaque exigence (ZCR) du processus que nous venons de décrire est déclinée en une ou plusieurs sous-exigences. Chaque sous-exigence comporte un énoncé de la tâche à satisfaire et un paragraphe incluant une justification et des recommandations supplémentaires afin d'aider les utilisateurs à mettre en œuvre la tâche. Par exemple, la ZCR 1 *Identifier le système à l'étude* comporte une seule sous-exigence (ZCR 1.1) *Identifier le périmètre du système à l'étude et les points d'accès.*

1.2.2 Applicabilité de notre méthode à la CEI 62443-3-2

Dans ce travail, nous avons développé une méthode d'appréciation du risque sûreté-sécurité automatisée. Cette partie présente l'intégration de notre méthode dans le processus d'évaluation des risques de la norme CEI 62443-3-2. Pour cela, nous présentons les exigences de la norme automatisables par notre méthode.

ZCR 1 - La ZCR 1 *Identifier le système à l'étude* comprend une seule exigence qui vise à identifier clairement le système à l'étude en délimitant le périmètre de l'évaluation de

cybersécurité et en identifiant tous les points d'accès au système étudié (System Under Consideration - SUC). L'identification doit comprendre l'ensemble des actifs nécessaires au fonctionnement de l'IACS et peut être réalisé par un diagramme d'inventaire, d'architecture système, de réseau, ou de flux de données (DFD). Dans notre méthode, nous avons automatisé la construction d'un DFD du système de contrôle pour MTMT (Partie 1 - section 1.1.1). Cette modélisation peut servir de base à une identification du système à l'étude et de ses points d'accès.

ZCR 2 et 5 - Le processus d'évaluation des risques CEI 62443-3-2 comprend à la fois une appréciation initiale (ZCR 2) et une appréciation détaillée (ZCR 5) du risque. La norme définit clairement les exigences à satisfaire pour réaliser une appréciation détaillée du risque (ZCR 5), ce que n'est pas le cas pour l'appréciation initiale du risque (ZCR 2). Toutefois, la norme précise que l'appréciation initiale et l'appréciation détaillée du risque doivent être dérivées de la même source (norme, cadre, etc.) afin de produire des résultats cohérents et homogènes. De plus, l'appréciation détaillée du risque et l'appréciation initiale du risque visent toutes les deux à déterminer le risque de cybersécurité non atténué. Compte tenu de ces observations, nous considérons que l'appréciation initiale du risque est conforme aux exigences ZCR 5.1 à ZCR 5.5 de l'appréciation détaillée du risque tel que présenté dans la Figure 1.8.

La Figure 1.8 présente le processus d'appréciation détaillé du risque de cybersécurité (ZCR 5) IEC 63443-3-2 pour lequel nous avons précisé les exigences automatisables par notre méthode (cadre de l'exigence en trait plein) et celles qui ne l'étaient pas (cadre de l'exigence en trait pointillé). Pour les exigences automatisables, nous avons mentionné à droite du cadre la partie de notre méthode qui permet leur automatisation. Dans la suite, nous présentons comment notre méthode permet d'automatiser chacune de ces exigences.

- **ZCR 5.2 :** La réalisation de l'exigence ZCR 5.2 nécessite une identification et une documentation des vulnérabilités connues. La première partie de notre méthode (Partie 1 – section 1.1.1) automatise le processus d'identification des vulnérabilités *Microsoft Threat Modeling Tool*. Ce processus inclut la récupération des CVE (vulnérabilités connues) des composants du système. Enfin, *Microsoft Threat Modeling Tool* comporte une fonction d'exportation de la liste des vulnérabilités identifiées (CVE comprises) au format CVS afin de produire la documentation requise par l'exigence ZCR 5.2.
- **ZCR 5.3 :** L'exigence ZCR 5.3 requiert de documenter les conséquences des scénarios de menaces selon l'impact sur la sécurité du personnel, les pertes financières, etc. L'exigence précise que, pour déterminer les conséquences et l'impact, il convient d'utiliser l'analyse préliminaire des risques (APR). Il est suggéré d'utiliser l'échelle de conséquence de l'organisation pour mesurer l'impact des scénarios de menace. La Partie 2 de notre méthode (cf. section 1.1.2) automatise la génération de scénarios d'attaque qui compromettent des fonctions de sûreté que nous avons dérivées d'une APR. Nous quantifions la gravité de l'impact conformément à l'échelle de gravité du danger utilisée dans l'APR. Cette échelle peut être qualitative ou quantitative et mesurer la gravité selon les exigences de l'organisation, qu'elles soient financières, basées sur la sécurité du personnel ou autres.

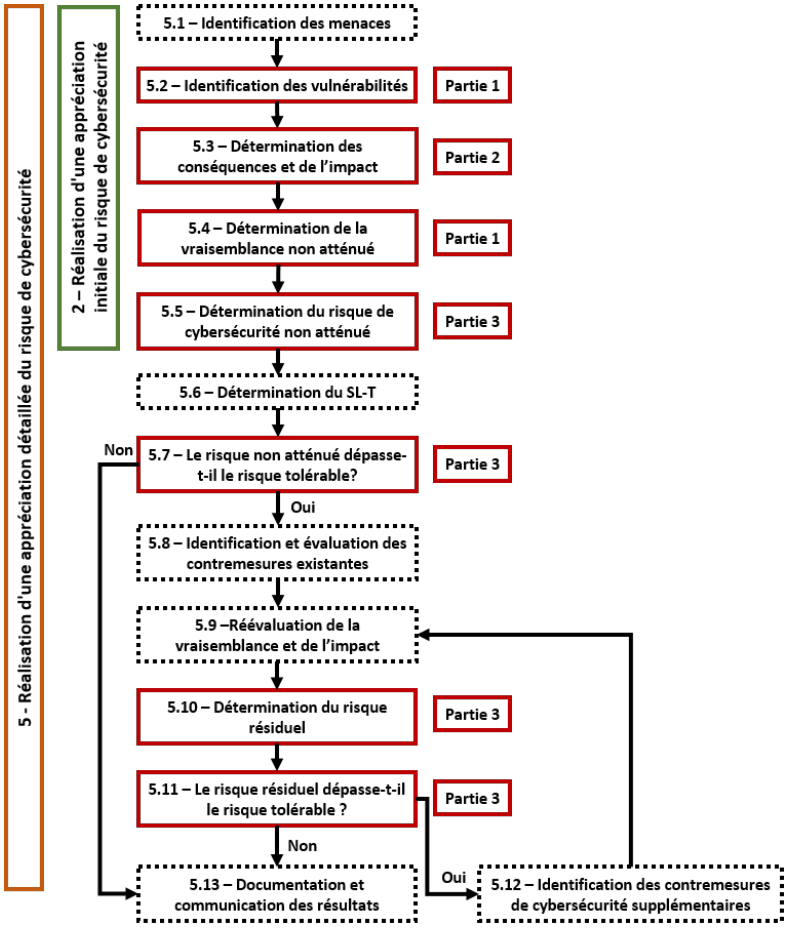


FIGURE 1.8 – Processus et automatisation des appréciations du risque CEI 62443-3-2

- **ZCR 5.4 :** La ZCR 5.4 suggère de déterminer la vraisemblance non atténuée de réalisation d'une menace selon une échelle semi-quantitative définie par l'organisation. Dans notre méthode, nous proposons de déterminer la vraisemblance d'un scénario d'attaque selon un score d'exploitabilité des vulnérabilités qui composent le scénario d'attaque (Partie 3 - section 1.1.3). Le score d'exploitabilité est extrait du système de notation semi-quantitatif [3] CVSS v3.1.
- **ZCR 5.5 & 5.10 :** Les exigences ZCR 5.5 et 5.10 déterminent le risque de cybersécurité en combinant la mesure de vraisemblance et de gravité des menaces à l'aide d'une matrice de risques. La seule différence entre la ZCR 5.5 et 5.10 est que la ZCR 5.5 détermine le risque non atténué à partir de la gravité déterminée par la ZCR 5.3 et la vraisemblance déterminée par la ZCR 5.4 alors que la ZCR 5.10 détermine le risque résiduel après une réévaluation de la vraisemblance et de l'impact (ZCR 5.9) dû à la prise en compte des contremesures existantes (ZCR 5.8) ou

3. Calcul d'un score d'exploitation basé sur des métriques qualitatives

supplémentaires (ZCR 5.12). Dans la Partie 3 de notre méthode (cf. section 1.1.3), nous proposons un processus automatique de classement des scénarios d'attaque dans une matrice de risques selon leur vraisemblance et la sévérité de leur impact.

- **ZCR 5.7 & 5.11 :** Les exigences 5.7 et 5.11 comparent respectivement le risque non atténué et résiduel avec le risque tolérable pour l'organisation. Si le risque n'est pas tolérable, des actions complémentaires sont exigées. Dans notre méthode, nous automatisons la prise en compte du risque tolérable par l'organisation dans la matrice de risques (Partie 3 - section 1.1.3). Cela nous permet d'identifier les scénarios d'attaque ayant un niveau de risque non tolérable par l'organisation et qui donc nécessitent un traitement complémentaire pour diminuer leur niveau de risque (vraisemblance et/ou sévérité).

1.2.3 Travaux connexes à l'automatisation de l'appréciation du risque CEI 62443-3-2

À notre connaissance, il n'existe pas de méthode d'automatisation du processus d'appréciation du risque de la CEI 62443-3-2. Cependant, Ehrlich et al. [24] (2023) présentent un prototype de mise en œuvre d'un processus d'appréciation du risque basé sur six exigences de la CEI 62443-3-2 (Zones & Conduit Requirements - ZCR). La méthodologie présentée comporte quatre étapes séquentielles : (1) Segmentation du réseau (ZCR 2 et ZCR 3.1), (2) Exigences & Garanties (ZCR 5.6), (3) Évaluation du risque (ZCR 5.1 et 5.2), et (4) Attestation (ZCR 7.1). Dans un second article [23] (2023), les auteurs présentent plus en détail l'étape *Exigences & Garanties* de leur méthode basée sur la ZCR 5.6. L'exigence ZCR 5.6 requiert l'établissement du niveau de sécurité souhaité (SL-T) pour l'IACS, le conduit ou la zone à l'étude. Les auteurs répondent à cette exigence en déterminant le niveau de sécurité cible (Security Level - Target SL-T) à partir du cadre MITRE ATT&CK[4] et de la bibliothèque Intel Threat Agent [13]. Nous n'avons pas trouvé d'informations sur l'automatisation des autres exigences. Dans leur travail [24], les auteurs construisent un processus d'appréciation du risque basé sur six exigences de la 62443-3-2. Dans notre travail, nous avons conçu un processus d'appréciation du risque basé sur la norme ISO 31000 que nous appliquons à l'évaluation des risques de la CEI 62443-3-2 pour partiellement automatiser son processus. De plus, comme présenté dans le Tableau 1.1, notre travail et celui de Ehrlich et al. [24] ne se focalisent pas sur les mêmes exigences et peuvent même être complémentaires.

ZCR	2	3	5.1	5.2	5.3	5.4	5.5	5.6	5.7	5.10	5.11	7.1
Ehrlich et al. [24]	✓	✓	✓	✓				✓				✓
Notre travail	✓			✓	✓	✓	✓		✓	✓	✓	

Table 1.1 – Comparaison des exigences traitées

Fockel et al. [29] (2019) présentent une méthodologie en quatre étapes pour effectuer une analyse des menaces conforme à la norme CEI 62443. Comme nous, les auteurs utilisent l'outil de modélisation des menaces de Microsoft (Microsoft Threat Modeling Tool

4. https ://attack.mitre.org/

– MTMT) pour identifier les menaces qui pèsent sur le système. Cependant, ils ne fournissent aucune information sur leur base de connaissances ICS pour MTMT (template). Ils précisent simplement que pour la conformité avec la CEI 62443-3-2, le template a été adapté au domaine de l'automatisation industrielle pour tenir compte de la taxonomie, des technologies et des menaces spécifiques au domaine [29]. Dans notre travail, nous avons développé une méthode pour modéliser les systèmes industriels dans MTMT. De plus, nous avons fourni un programme pour intégrer facilement de nouvelles connaissances dans MTMT et un processus automatisable pour construire le modèle du système dans MTMT. Enfin, nous avons présenté dans la Section 1.2.2 comment l'automatisation du flux de travail de MTMT permettait de répondre à certaines exigences de la CEI 62443-3-2.

1.3 Conclusion du chapitre

Dans ce chapitre, nous avons présenté le processus d'appréciation du risque sûreté-sécurité de notre méthode et de son applicabilité à l'évaluation des risques de la CEI 62443-3-2. Notre méthode, composée de trois parties, relève plusieurs challenges. Le premier challenge a été de proposer une méthode d'appréciation du risque sûreté-sécurité qui passe à l'échelle sur des systèmes larges. En effet, représenter les interactions entre la sûreté et la cybersécurité induit un espace d'état du modèle sûreté-sécurité ingérable pour des systèmes industriels de grande taille. Ces problématiques de passage à l'échelle sont traités dans la section 3.1.3 du Chapitre 3 qui aborde aussi, dans la section 3.4, les aspects de complétude de la méthode comparativement à l'état de l'art.

Le deuxième challenge a été de proposer un processus d'automatisation d'un outil de modélisation de la menace afin d'identifier les vulnérabilités des systèmes industriels. En effet, les méthodes de modélisation de la menace sont des approches couramment utilisées dans le domaine de l'IT (Information Technology), mais manquent de représentativité lorsqu'il s'agit de représenter les systèmes industriels. Le prochain chapitre (Chapitre 2) est consacré à la présentation de cette contribution.

Enfin, le troisième challenge, l'automatisation de la méthode, a été développé dans ce chapitre. Dans une première section (cf. section 1.1), nous avons commencé par présenter une vue d'ensemble de notre méthode afin de fournir une compréhension générale du déroulement de la méthode. Ensuite, nous avons détaillé le processus de chacune des parties pour présenter le niveau d'automatisation de la méthode que nous avons catégorisée selon trois critères : *manuel*, *automatisé* et *automatisable* où les critères *automatisé* et *automatisable* définissent respectivement si nous avons développé une implémentation informatique ou non. Puis, dans la section 1.2.2, nous avons présenté comment notre méthode pouvait servir à automatiser partiellement une évaluation des risques de cybersécurité normée (CEI 62443-3-2). L'automatisation permet de réduire les efforts manuels nécessaires à l'application de ces normes, ce qui participe à réduire les erreurs humaines liées aux tâches manuelles et à augmenter la cohérence et la répétabilité des résultats grâce à l'uniformisation du processus. Enfin, selon Eggers et Le Blanc [22], l'outillage (automatisation) d'une procédure d'appréciation du risque participe aussi à son acceptation au sein d'une organisation.

2 Identification des vulnérabilités des ICS : Modélisation de la menace

Ce second chapitre se focalise sur la première partie (Partie 1) de notre méthode d'appréciation du risque. Cette partie automatise le processus d'identification des vulnérabilités à partir d'un outil de modélisation de la menace pour les systèmes de contrôle industriel. La modélisation de la menace est une méthode classique pour identifier les vulnérabilités des systèmes dans le domaine de l'IT (Informational Technology). Cependant, ces méthodes sont difficilement applicables aux systèmes OT (Operational Technology), dont font partie les ICS, notamment par manque de représentativité des équipements et des vulnérabilités spécifiques à ces infrastructures. Par conséquent, nous avons développé un processus de modélisation des équipements ICS et de leurs vulnérabilités à partir des *profils de protection pour les systèmes industriels* [30] produits par l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information). Ce processus a donné lieu à la création d'une base de données de composants et de vulnérabilités ICS et d'un programme, tous deux librement accessibles en ligne [1].

Microsoft Threat Modeling Tool (MTMT), l'outil de modélisation de la menace sur lequel est basée notre méthode, a un flux de travail en trois étapes : (1) création d'un template, (2) construction du modèle du système et (3) analyse du modèle. Le programme que nous avons développé traduit notre base de données vers un template (1) utilisable par MTMT ; et MTMT automatise par défaut l'analyse du modèle (3). Afin d'automatiser pleinement le flux de travail de MTMT, nous avons aussi conçu une méthode automatisable de construction du diagramme de flux de données (DFD — modélisation du système utilisée par MTMT) d'un système de contrôle industriel à partir d'un format normalisé (PLCopen [55]) des programmes des PLC.

Aperçu - La première section (section 2.1) de ce chapitre introduit les briques élémentaires de MTMT et la méthode de modélisation de la menace STRIDE sous-jacente à MTMT. Puis, nous détaillons dans la section 2.2 notre processus de modélisation des composants ICS et de leurs vulnérabilités dans MTMT. Nous présentons ensuite en détail (section 2.3) l'automatisation du flux de travail de MTMT : (1) Création d'un template, (2) Construction du modèle du système et (3) Analyse du modèle. Enfin, nous concluons par une synthèse des contributions détaillées dans ce chapitre et une présentation des perspectives de ce travail.

1. `https://github.com/StrideICS/StrideICS`

2.1 STRIDE et Microsoft Threat Modeling Tool

Dans notre méthode d'appréciation du risque, nous utilisons l'outil *Microsoft Threat Modeling Tool* (MTMT) pour automatiser le processus d'identification des vulnérabilités pour les systèmes de contrôle industriel. Cette première section introduit la méthode de modélisation de la menace STRIDE, sous-jacente à MTMT, puis présente plus en détail le fonctionnement de MTMT. Enfin, nous terminons cette section en introduisant les travaux connexes à l'application de la méthode STRIDE et de MTMT aux systèmes de contrôle industriel.

2.1.1 STRIDE

STRIDE est une méthode de modélisation de la menace développée en 2009 par Loren Kohnfelder et Praerit Garg [44] et adoptée par Microsoft dans son cycle de développement de la sécurité (Security Development Lifecycle - SDL). Soutenue par Microsoft, cette méthodologie est largement répandue chez les constructeurs et fournisseurs de solutions. STRIDE est l'acronyme de six familles de menaces correspondant à la violation d'une exigence de sécurité (précisée entre parenthèses) :

- ***Spoofing*** — **Usurpation d'identité (Authenticité) :** Prétendre être un utilisateur (personne, processus logiciel ou équipement) autre que soi-même.
- ***Tampering*** — **Falsification (Intégrité) :** Modifier une information sur un disque, sur un réseau ou dans la mémoire.
- ***Repudiation*** – **Répudiation (Non-répudiation) :** Affirmer que l'on n'a pas fait une action ou que l'on n'en est pas responsable. La répudiation est liée au manque de preuves pour attribuer une action à un utilisateur.
- ***Information disclosure*** – **Divulgation d'informations (Confidentialité) :** Fournir des informations à un utilisateur non autorisé.
- ***Denial of service*** – **Déni de service (Disponibilité) :** Absorber les ressources nécessaires à la fourniture du service.
- ***Elevation of privilege*** — **Élévation de privilège (Autorisation) :** Permettre à un utilisateur de faire une action qu'il n'est pas autorisé à faire.

La méthode STRIDE consiste à identifier les potentiels dysfonctionnements du système à partir des menaces STRIDE. Cette méthode se base sur la définition même d'une attaque : *Réalisation d'une menace par l'exploitation d'une ou plusieurs vulnérabilités.* Si un analyste trouve un moyen de réaliser une des menaces STRIDE, alors il vient d'identifier une vulnérabilité du système. Pour ce faire, la méthode s'appuie sur un diagramme de flux de données (DFD) du système. Un DFD est une représentation graphique des flux d'information échangés par les éléments du système. Dans la méthode STRIDE, un DFD comporte 5 types d'éléments [62] présentés dans le Tableau 2.1.

Dans la variante STRIDE utilisée dans MTMT, STRIDE par interaction, les vulnérabilités (dysfonctionnements) du système sont identifiées à partir des interactions du système (triplets source, destination, interaction du DFD) tel que, par exemple, un utili-

Élément	**Apparence**	**Signification**
Processus	Cercle	Code en cours d'exécution
Flux de données	Flèches directionnelles	Communications entre les éléments d'un système
Stockage de données	Deux lignes parallèles	Tout support qui peut stocker des données
Entité extérieure	Rectangles	Processus et personnes en-dehors de notre contrôle
Frontière de confiance	Lignes ou bordures en pointillé	Démarcation entre des entités ayant des privilèges différents

Table 2.1 – Éléments d'un DFD STRIDE

sateur non authentifié (source) qui interroge (interaction) un serveur Web (destination). Le Tableau 2.2, extrait de [62], liste l'ensemble des interactions possibles dans un DFD et les menaces STRIDE qui peuvent être appliquées à chacune d'entre elles.

Élément	**Interaction**	**S**	**T**	**R**	**I**	**D**	**E**
Processus	Le processus envoie des données sortantes vers un stockage de données.	X			X		
	Le processus envoie la sortie vers un autre processus	X		X	X	X	X
	Le processus envoie la sortie vers une entité extérieure (code).	X		X	X	X	
	Le processus envoie la sortie vers une entité extérieure (humain).			X			
	Le processus reçoit des données entrantes du stockage de données.	X	X			X	X
	Le processus reçoit des données entrantes d'un processus.	X		X		X	X
	Le processus reçoit des données entrantes provenant d'une entité extérieure.	X				X	X
Flux de données	Franchit les limites de la machine (frontière de confiance)		X		X	X	
Stockage de données	Le processus envoie des données sortantes vers le stockage de données.		X	X	X	X	
	Le processus reçoit des données entrantes du stockage de données.			X	X	X	
Entité extérieure	L'entité extérieure transmet des données au processus.	X		X	X		
	L'entité extérieure reçoit des données du processus.	X					

Table 2.2 – Interactions et applicabilité des menaces de STRIDE par interaction [62]

2.1.2 Microsoft Threat Modeling Tool

Microsoft Threat Modeling Tool (MTMT) est un outil graphique qui automatise la méthodologie STRIDE par interaction. L'outil fonctionne en 3 étapes : 1) création d'un *template*, 2) construction du modèle du système sous forme d'un diagramme de flux de données (DFD), et 3) analyse du modèle. Un template est une bibliothèque d'éléments mis à disposition de l'utilisateur pour modéliser le système. Il contient, à la fois, les éléments nécessaires à la création du DFD, appelés gabarits, et une liste de menaces. Microsoft fournit trois templates pour modéliser respectivement les équipements classiques des systèmes IT (*default.tb7*), les équipements médicaux (*MedicalDeviceTemplate.tb7*) et les architectures Cloud Azure (*Azure Cloud Services.tb7*). Ces templates sont en accès libre sur Github [2]. Le processus de modélisation et d'analyse de MTMT repose sur les éléments définis dans le template. Ce template dont la Figure 2.1 illustre une vue globale est composé de gabarits et de types de menaces pour lesquels nous proposons une description détaillée.

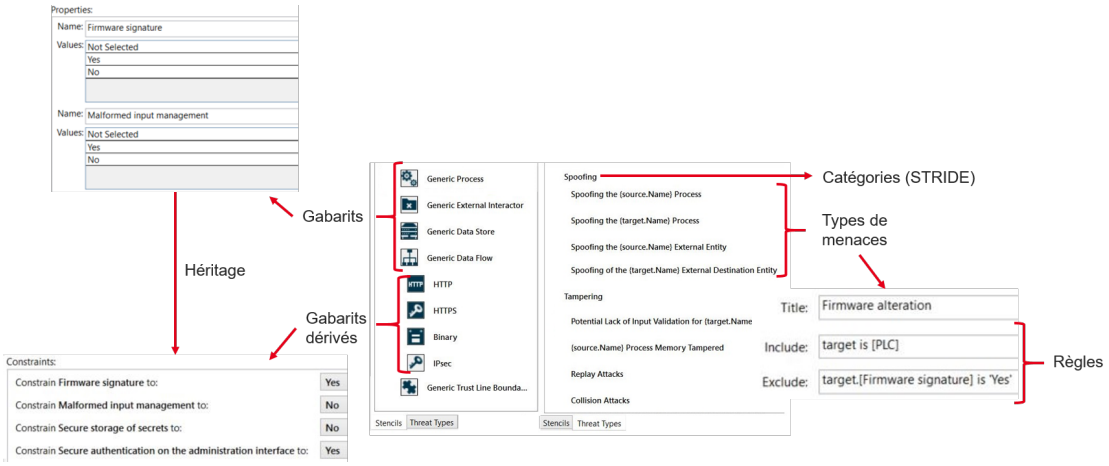


FIGURE 2.1 – Vue d'ensemble d'un template MTMT

Les gabarits

Les gabarits sont des éléments de MTMT qui modélisent des composants génériques tels que : un processus, un flux de données ou un stockage de données. Les gabarits sont caractérisés dans MTMT par :

- **Une forme (Shape) :** La forme détermine l'apparence du gabarit dans le DFD. Chaque apparence (forme) fait référence à un élément spécifique du DFD et a sa propre signification (cf. Tableau 2.1), tel que les lignes parallèles pour représenter un stockage de données ou un cercle pour définir un processus.
- **Un comportement (Behavior) :** Le comportement définit si un gabarit est une cible (Target), un flux (Flow), ou une frontière de confiance (Boundary) et permet de construire automatiquement les triplets {source, destination, interaction}.

2. https ://github.com/microsoft/threat-modeling-templates

- **Des propriétés (Properties) :** Les propriétés sont des attributs du composant que nous souhaitons modéliser. Ces attributs ont plusieurs valeurs possibles qui seront fixées par l'utilisateur lors de la modélisation du système en fonction des caractéristiques de l'élément à modéliser.

Le template par défaut de MTMT contient six sortes de gabarits représentant les cinq éléments d'un DFD présenté dans le Tableau 2.1 : *Les processus*, *Les flux de données*, *Les entités extérieures*, *Les stockages de données* et *Les frontières de confiances* (cet élément est séparé en deux gabarits différents). La Figure 2.2 présente l'interface d'édition du gabarit *Processus Générique* dans MTMT où nous pouvons voir son comportement (*Cible*), sa forme (*Cercle*), une partie de ses propriétés (*Code Type* et *Running As*) et les valeurs possibles des propriétés (*Managed*, *Unmanaged*, etc.).

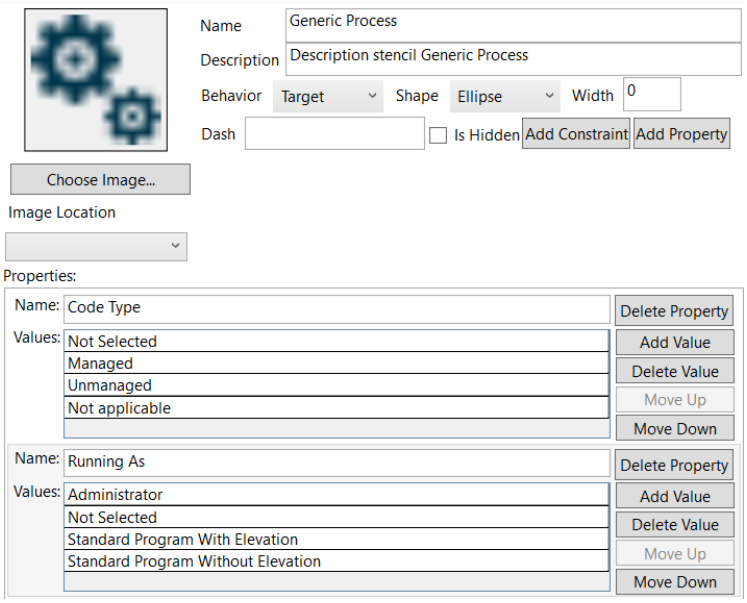


FIGURE 2.2 – Fenêtre d'édition du gabarit *Processus Générique* dans l'outil MTM

Dans MTMT, les gabarits peuvent être dérivés (gabarits dérivés) pour modéliser des éléments plus spécifiques. Par exemple, nous pouvons dériver le gabarit *Flux de Données Générique* pour modéliser un protocole de communication spécifique comme HTTP. Un gabarit dérivé hérite du même comportement et de la même forme qu'un gabarit de référence. Les propriétés du gabarit de référence deviennent des contraintes pour ses gabarits dérivés avec des valeurs prédéfinissable pour modéliser un équipement spécifique. Par exemple, le gabarit *Flux de Données Générique* du template par défaut possède plusieurs gabarits dérivés, dont *HTTP* présenté dans la Figure 2.3. *HTTP* hérite donc des propriétés du gabarit *Flux de Données Générique* (son gabarit de référence) sous forme de contraintes. HTTP étant un protocole non sécurisé, la valeur des contraintes *Source Authentifiée*, *Destination Authentifiée*, *Assure la confidentialité*, et *Assure l'intégrité* est fixée à *Non*. Dans son template par défaut, l'outil MTMT comptabilise une cinquantaine de gabarits dérivés représentant les composants caractéristiques des architectures IT.

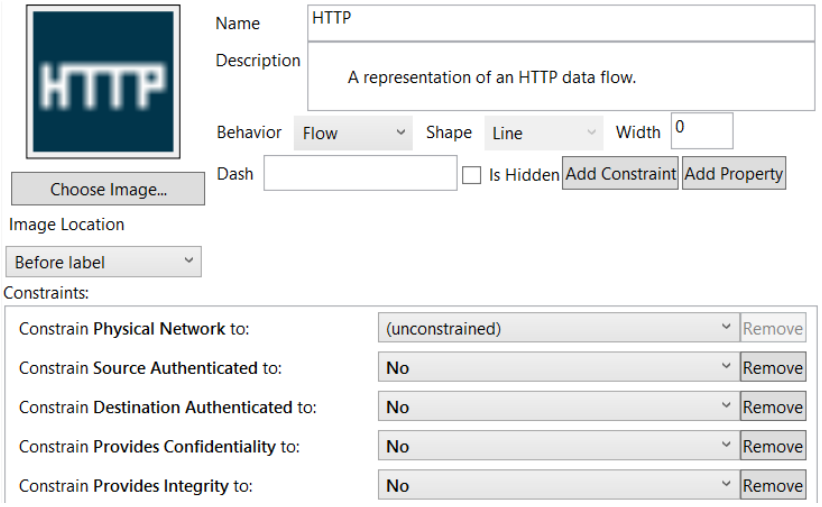


FIGURE 2.3 – Fenêtre d'édition du gabarit dérivé *HTTP*

Les menaces

L'analyse des menaces STRIDE du modèle est automatisée dans MTMT grâce aux menaces définies dans le template. Dans MTMT, les menaces sont appelées des *Types de menace* et sont catégorisées par famille de menace STRIDE (Usurpation d'identité, Falsification, etc.). Par exemple, La Figure 2.4 présente le type de menace *Compromission d'un Flux de Données Authentifié* appartenant à la catégorie *Falsification* de STRIDE.

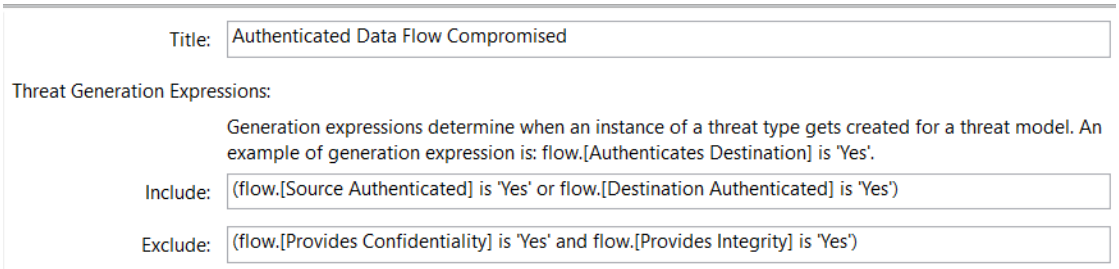


FIGURE 2.4 – Type de menace MTMT

Les types de menace dans MTMT sont définis sous forme de règles d'applicabilité. Une première règle, *Inclusion* (Include), détermine quand le type de menace doit être pris en compte et une seconde, *Exclusion*, prioritaire sur la règle d'inclusion, définit quand le type de menace n'est pas applicable. Par exemple, le type de menace présenté dans la Figure 2.4 doit être inclus si le flux de données authentifie sa source ou sa destination (règle *Inclure*) et doit être exclus (prioritaire sur la règle *Inclure*) si le flux de données est confidentiel et intègre (règle *Exclure*). Ces règles permettent d'automatiser l'analyse du modèle par MTMT en recherchant les types de menaces applicables au modèle.

2.1.3 État de l'art

En 2017, Khan et al. [43] présentent une méthode en cinq étapes pour une application systématique de STRIDE pour les systèmes cyberphysiques (CPS). Leur méthode est appliquée à travers un cas d'usage qui permet à la fois d'expliquer la méthode et de montrer sa pertinence. La première étape consiste à décomposer le système en composants. Les auteurs choisissent de ne pas prendre en compte dans l'analyse les composants physiques qui ne sont pas susceptibles de faire l'objet de cyberattaques, comme les connexions électriques entre le disjoncteur et l'alternateur ou l'alimentation électrique. Ensuite, les composants du système sont représentés dans un diagramme de flux de données. Dans une troisième étape, les auteurs identifient les intentions de l'attaquant, appelées conséquences de la menace, sur le procédé physique. Les conséquences sont déterminées par un expert du système. Ensuite, ils réalisent une approche classique STRIDE par élément en faisant correspondre les menaces découvertes avec leurs conséquences (intention de l'attaquant). Enfin, les vulnérabilités sont identifiées à partir des menaces (par exemple, le manque d'authentification) et une stratégie de remédiation est planifiée. Cette méthode est une application classique de STRIDE pour laquelle les auteurs proposent de faire correspondre les menaces identifiées avec leurs impacts sur le procédé. Contrairement à notre méthode, Khan et al. [43] proposent un processus manuel qui repose largement sur l'expertise de l'utilisateur à la fois pour déterminer les vulnérabilités exploitées afin de réaliser chaque menace identifiée ; et pour associer les menaces avec leurs conséquences sur le procédé. Dans notre travail, nous proposons une version automatisée du processus d'identification des vulnérabilités comprenant les vulnérabilités spécifiques aux systèmes industriels ainsi qu'une automatisation de l'identification des conséquences des vulnérabilités pour la sûreté du système (cf. Partie 2 de notre méthode — section 1.1.2).

De la même manière, Asif et al. [4] (2021) appliquent l'approche STRIDE dans un système IoT avec un cas d'usage basé sur l'agriculture de précision. Les auteurs appliquent la méthode STRIDE avec *Microsoft Threat Modeling Tool* qui identifie 58 menaces. Puis, ils énumèrent des mécanismes de défenses génériques qui pourraient être appliqués au cas d'usage pour sécuriser l'architecture.

En 2021, Fla et al. [28] proposent un template MTMT dédié aux réseaux électriques intelligents (Smart grids). Le template repose sur la création de gabarits qui sont représentatifs d'un réseau électrique intelligent générique et de menaces basées sur une revue de la littérature, des modèles de références existants et des cyberattaques connues sur les systèmes OT (Operational Technologies). Cet article montre la classification de ces menaces dans les catégories STRIDE mises en œuvre dans leur modèle MTMT.

AbuEmera et al. [3] proposent un template, appelé catalogue dans l'article, comprenant les composants décrits dans le modèle de référence ENISA [6]. À partir de ce template, les auteurs construisent un modèle générique d'une usine intelligente (*Smart Factory*) dans MTMT. Puis, ils appliquent la méthode de Khan et al. [43] sur leur modèle. Les auteurs attribuent ensuite des règles aux menaces identifiées pour les intégrer dans MTMT et évaluent la gravité de ces menaces selon le système de notation CVSS v3.1 [27] (Common Vulnerability Scoring System). Enfin, les auteurs proposent des recommandations pour la sécurité des usines intelligentes.

D'après nos connaissances, seulement AbuEmera et al. [3] et Fla et al. [28] proposent un template pour les systèmes industriels. AbuEmera et al. [3] ne fournissent ni leur template, ni les informations suffisantes pour reproduire leur template. En effet, les auteurs présentent une vue d'ensemble des composants qu'ils ont dans leur template MTMT mais sans aucune précision sur leurs propriétés ; et revendiquent d'avoir identifié 221 menaces, mais ils fournissent les règles d'inclusions et d'exclusions, nécessaires pour intégrer ces menaces dans MTMT, pour seulement six menaces. Dans notre travail, nous proposons une méthode reproductible de modélisation des composants ICS et de leurs vulnérabilités dans MTMT que nous avons intégrée dans un template dédié aux ICS librement accessible en ligne [3]. Fla et al. [28] proposent un template pour les réseaux électriques intelligents comprenant des gabarits représentatifs de ce type de système selon l'avis d'un expert. Dans notre travail, nous fournissons des composants génériques applicables à tous les systèmes de contrôle industriel, incluant les réseaux électriques intelligents, à partir d'un document de l'autorité nationale en cybersécurité (l'ANSSI).

Dans cette section, nous avons présenté la méthode de modélisation de la menace STRIDE, le fonctionnement du logiciel MTMT ainsi que leur application aux systèmes industriels dans l'état de l'art. Dans la prochaine section, nous détaillons notre processus de modélisation des composants ICS et de leurs vulnérabilités dans MTMT.

2.2 Modélisation des composants des ICS et de leurs vulnérabilités

Pour fonctionner, Microsoft Threat Modeling Tool (MTMT) a besoin d'un template. Ce template contient à la fois les éléments mis à disposition de l'utilisateur pour construire le modèle du système (gabarits et gabarit dérivés) et une description (règles d'inclusions/exclusions) d'un ensemble de menaces qui automatise l'analyse du modèle. Il n'existe actuellement aucun template librement accessible dédié aux systèmes industriels qui permettrait à des experts en sécurité de modéliser leur système et ses vulnérabilités. En effet, les composants des ICS manquent de représentativité dans les outils de modélisation de la menace, ce qui complexifie leur utilisation lorsqu'il s'agit d'identifier les vulnérabilités spécifiques aux ICS. Dans cette section, nous présentons le processus que nous avons développé pour modéliser les composants des ICS et leurs vulnérabilités dans MTMT. Ce processus alimente une base de données extensible de composants et de vulnérabilités qui est par la suite traduite en template MTMT dédié à la modélisation des ICS.

2.2.1 Une base de données extensible

Un template est un fichier au format XML propre à MTMT (extension .tb7). Dans ce travail, pour plusieurs raisons, nous proposons de baser notre méthode sur une base de données extensible que nous traduisons ensuite en template au lieu de développer directement un template. Premièrement, une base de données extensible facilite la création de

3. https ://github.com/StrideICS/StrideICS

composants industriels spécifiques et permet d'agréger les connaissances d'une entreprise sur les menaces, les composants, et les protocoles de communication des ICS sans passer par le logiciel et l'interface graphique de MTMT. En effet, actuellement, il existe des milliers de références de composants industriels sur le marché. Créer et maintenir à jour un template, tel que fourni par l'outil MTMT, incluant tous ces composants, serait une tâche fastidieuse. Deuxièmement, la base de données évite les conflits modèle/template. Dans MTMT, un modèle (DFD du système) est lié à un template, et par conséquent la modification d'un template pour le mettre à jour, ou, le fait d'ajouter un équipement peut entraîner un conflit entre ce nouveau template et les modèles créés à partir de l'ancienne version du template. De plus, MTMT n'est pas prévu pour sauvegarder les différentes versions d'un template, lorsqu'on sauvegarde une modification, nous perdons la version précédente et, s'il existe un conflit entre le nouveau template et un de nos modèles, nous ne sommes plus en capacité d'ouvrir le modèle à nouveau et nous en perdons l'usage. C'est pour ces raisons que nous avons choisi de développer une base de données extensible pour représenter les composants ICS dans MTMT.

Nous utilisons le format TOML [56] (Tom's Obvious, Minimal Language) pour notre base de données de composants. Le TOML est un format conçu pour être facilement lu et écrit et comprend une spécification open source. Ces caractéristiques sont particulièrement importantes dans notre cas, car la base de données a vocation à être régulièrement mise à jour par des personnes qui ne sont pas familières des formats de données classiques de l'IT comme le JSON [10] ou le XML [17]. Le format TOML intègre trois éléments principaux : les sections ([section.sous-section]) qui permettent de structurer les informations, les paires (clé = valeur) associent une valeur avec un nom générique (clé) et les commentaires (# commentaire).

```
[project]
title = ""
version = ""
author = ""
id = ""

[cveGathering]
        [cveGathering."Name"]
        ID = '<uuid4>'
        cpe = "<cpe:2.3>"
        period = ["YYYY-MM-DD","YYYY-MM-DD"]
        severity = ["CRITICAL", "HIGH", "MEDIUM", "LOW"]

[GenericElements]
    [GenericElements."NAME"]
    ID = ""
        Description = ""
        ParentElement = ""
        Image = "<base64>"
        Hidden = "false"
        Representation = ""
        StrokeThickness = "0"
        ImageLocation = ""
        StencilConstraints = [{SelectedStencilType = "Any", SelectedStencilConnection = "Any"}]
    #optional
        Attributes."Attr1" = ["Value1", "Value2"]
        Attributes."Attr2" = ["Value1", "Value2", "Value3"]

[StandardElements]
    [StandardElements."NAME"]
    ID = "<uuid4>"
        Description = ""
        ParentElement = ""
        Image = "<base64>"
        Hidden = "false"
        Representation = ""
        StrokeThickness = "0"
```

```
        ImageLocation = ""
        StencilConstraints = [{SelectedStencilType = "Any", SelectedStencilConnection = "Any"}]
    #optional
        Attributes."Attr1" = ["Value1", "Value2"]
        Attributes."Attr2" = ["Value1", "Value2", "Value3"]

[ThreatCategories]
        [ThreatCategories."NAME"]
        Id = ""
        ShortDescription = ""
        LongDescription = ""

[ThreatTypes]
        [ThreatTypes."NAME"]
        GenerationFilters.Include = ""
        GenerationFilters.Exclude = ""
        Id = ""
        Category = ""
        Description = ""
```

Listing 2.1 – Structure de la base de données

La structure de notre base de données, présentée dans la Liste 2.1, reflète celle d'un template MTMT. En effet, nous avons défini quatre sections qui comprennent les quatre éléments définis par un template MTMT, à savoir, les gabarits, les gabarits dérivés, les catégories de menaces et les types de menaces. En plus de ces sections, nous en avons rajouté deux autres, une première qui définit des informations propres au template comme le numéro de version (section *Project*) et une seconde qui fournit les informations nécessaires pour récupérer les CVE des composants. Nous reviendrons plus en détail sur la récupération des CVE dans la section 2.2.2. La Figure 2.5 présente une partie de la description en TOML du gabarit dérivé *HTTP* dans notre base de données extensible. À titre d'information, un template MTMT est un fichier XML dans lequel les gabarits s'appellent des *Generic Elements* et les gabarits dérivés des *Standard Elements*. Nous avons gardé la dénomination utilisée dans le fichier XML pour faciliter la traduction de notre base de données en template MTMT.

```
[StandardElements."HTTP"]
ID = 'bfae49d9-4651-42c2-a307-1581f073035f'
Description = "A representation of an HTTP data flow."
ParentElement = "GE.DF"
Image = "iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAABGdBTUEAALGOfPtRkwAAACBjSFJNAAC
Hidden = "false"
Representation = "Line"
StrokeThickness = "0"
ImageLocation = "Before Label"
StencilConstraints = [{SelectedStencilType = "Any", SelectedStencilConnection = "Any"}]
Attributes."Physical Network" = ""
Attributes."Source Authenticated" = "No"
Attributes."Destination Authenticated" = "No"
Attributes."Provides Confidentiality" = "No"
Attributes."Provides Integrity" = "No"
```

FIGURE 2.5 – Partie du gabarit dérivé *HTTP* dans la base de données TOML

Notre base de données extensible comporte les gabarits représentatifs des composants que l'on retrouve couramment dans les systèmes industriels. Cependant, nous estimons qu'il n'est pas suffisant de considérer uniquement les dispositifs ou les protocoles génériques (par exemple, un PLC), car certains équipements industriels sont plus vulnérables que d'autres. Il est difficile d'affirmer, par exemple, qu'un PLC générique présente une

menace de falsification. Pour pallier cette limitation, nous proposons d'intégrer des dispositifs spécifiques, par exemple des PLC du commerce, au template en tant que gabarits dérivés. Pour définir un nouveau composant spécifique, l'utilisateur doit simplement fixer les contraintes du gabarit dérivé (hérité du gabarit de référence). Par exemple, il est possible de modéliser un PLC *Centum VP* de l'entreprise *Yokogawa Electric* en créant un gabarit dérivé d'un PLC pour lequel nous fixons une valeur à ces contraintes.

La qualité des composants modélisés dépend donc de la pertinence des propriétés définies dans le gabarit de référence. Pour construire nos gabarits et définir leurs propriétés, nous avons utilisé les profils de protection [30] (PP) fournis par l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information). Ces profils de protection ont été rédigés dans le cadre du groupe de travail sur la cybersécurité des systèmes industriels (ISWG - Industrial Systems Work Group), qui comprend des entreprises telles que Phoenix Contact, Schneider Electric, Thales, ou Siemens, et intègrent les équipements et logiciels les plus courants des systèmes industriels [30]. Les profils de protection servent initialement de référence dans une démarche d'obtention d'une certification de sécurité de premier niveau (CSPN) [30]. Nous avons choisi la CSPN (reconnue par la France et l'Allemagne) comme base de travail, car, malgré un effort de structuration de l'écosystème de la cybersécurité industrielle pour créer un schéma de certification, il n'existe pas, à ce jour, de schémas de certification pour les systèmes industriels [25] largement reconnues par les agences de cybersécurité étatiques équivalentes aux critères communs [4]. Un profil de protection (PP) se présente en quatre parties :

1. **Descriptif du produit :** Le descriptif du produit comprend cinq parties couvrant : (1) une description générale du produit, (2) une description des fonctions du produit, (3) une description de l'utilisation du produit, (4) une description des utilisateurs, et enfin (5) les hypothèses sur l'environnement.
2. **Biens sensibles à protéger :** Les biens sensibles à protéger sont les éléments du composant qui sont essentiels à son fonctionnement. Les profils de protection distinguent les biens sensibles du composant et de son environnement. Pour chacune de ces deux catégories, le composant et son environnement, un tableau énumère leurs biens sensibles et leurs besoins de sécurité en termes de disponibilité, de confidentialité, d'intégrité et d'authenticité.
3. **Menaces :** Les PP présentent, dans leur partie de description des menaces, une description des agents menaçants et des menaces retenues comme pertinentes pour le composant et son environnement. La description des agents menaçants s'apparente à la définition des modèles d'attaquant et les menaces aux définitions des événements redoutés qui impactent les fonctions délivrées par le composant et son environnement.
4. **Objectifs de sécurité :** Les PP listent les mécanismes de protection à implémenter (objectifs de sécurité) pour que le composant puisse satisfaire aux besoins de sécurité au regard des menaces retenues.

Si nous faisons un parallèle entre l'implémentation de la méthode STRIDE dans MTMT et les PP, nous observons de nombreuses similitudes qui nous permettent de créer les

4. https ://www.commoncriteriaportal.org/index.cfm

gabarits de notre base de données selon une référence de certification. Par conséquent, la base de données que nous proposons fournit aux utilisateurs à la fois la capacité d'identifier les vulnérabilités de leur système industriel ou de leur produit et les aide à préparer une certification de sécurité de premier niveau (CSPN).

Nous avons donc créé les gabarits de notre base de données ICS à partir des profils de protection pour les systèmes industriels de l'ANSSI. Tel que précisé précédemment (section 2.1.2), un gabarit est caractérisé par : une forme, un comportement et des propriétés ; et une menace est caractérisée par une règle d'inclusion et une règle d'exclusion. Pour modéliser un composant ICS dans MTMT, nous faisons correspondre les caractéristiques des gabarits (forme, comportement et propriétés) et des menaces (catégorie et règles) que nous avons introduit dans la section 2.1.2 avec les profils de protection de l'ANSSI tel que présenté dans la liste suivante (cette liste est synthétisée dans la Figure 2.6) :

- **Gabarit** : Chaque PP est construit pour une cible de sécurité (ToE - Target of Security) pour laquelle nous créons un gabarit.
  - **Forme :** La forme d'un gabarit définit sa fonction, elle sera donc attribuée selon la description des fonctions du produit dans le PP. Par exemple, un PLC a une fonction d'exécution d'un programme PLC et est donc modélisé comme un processus (élément qui exécute du code) dans le DFD. À contrario, la fonction d'un serveur de journalisation (historian) est de stocker les alarmes issues de la supervision et sera donc modélisé par un stockage de données (deux lignes parallèles).
  - **Comportement :** Le comportement est assigné selon le type de gabarit : équipement (Target), protocole de communication (Flow) ou zone logique (Boundary). Dans le cas des profils de protection, nous modéliserons uniquement des cibles (Targets).
  - **Propriétés :** Les propriétés des équipements sont déterminées en fonction des objectifs de sécurité du PP. Par exemple, le firmware du PLC doit être signé (objectif de sécurité), nous avons donc créé une propriété *Signature du firmware* avec trois valeurs possibles : *Oui*, *Non*, et *Non sélectionné* dans le cas où l'information n'est pas connue ou que l'analyste ne souhaite pas prendre en compte cette menace.
- **Type de menaces :** Les PP définissent un ensemble de menaces retenues pour le composant et pour lesquelles nous créons un type de menace, par exemple une corruption du firmware.
  - **Catégorie :** La catégorie de menace (STRIDE) est déterminée selon le besoin de sécurité violé par la menace. Par exemple, la corruption du firmware est possible en cas de manque d'authentification (mise à jour illégitime) ou d'intégrité (modification du firmware). Nous ajoutons cette menace à la catégorie *Falsification* car l'événement redouté est la modification du firmware (corruption).
  - **Règles :** La règle *Inclure* du type de menace définit la ou les vulnérabilités qui permettent de réaliser cette menace. Par exemple, le PP présente trois vulnérabilités pour la menace *Corruption du firmware*, une substitution d'une

mise à jour corrompue à une mise à jour légitime, une injection de code et une modification de la version du firmware installé sur le PLC. La règle *Exclure* vérifie si le besoin de sécurité qui protège le composant est implémenté, tel que la signature du firmware pour la menace de *Corruption du firmware*.

FIGURE 2.6 – Synthèse de la modélisation depuis les profils de protection

Certains composants des ICS qui ne font pas l'objet d'un PP peuvent tout de même être modélisés par cette technique en dérivant le PP d'un composant proche. Par exemple, un RTU (Remote Terminal Unit) peut être considéré comme un PLC avec moins de fonctionnalités. En effet, un RTU possède un firmware et un mode de fonctionnement (*run* ou *stop* par exemple) comme les PLC mais ne possède pas de programme utilisateur. Par conséquent, nous pouvons tout de même modéliser un RTU en filtrant les informations du PLC qui ne lui sont pas applicables comme le programme utilisateur. Ce travail de dérivation des propriétés d'un profil de protection d'un composant vers d'autres composants doit être effectué par un expert afin de définir correctement les propriétés de l'équipement.

La base de données, que nous fournissons en accès libre [5], a vocation à être personnalisée par les utilisateurs en fonction de leurs besoins particuliers, en ajoutant des menaces provenant de leur CTI (Cyber Threat Intelligence), des gabarits pour représenter les composants spécifiques de leur domaine, etc. À ce jour, nous avons défini six gabarits à partir des profils de protection : Automate programmable industriel (*PLC*), Logiciel d'ingénierie (*Engineering workstation*), Client SCADA/MES (*SCADA MES client*), Serveur MES (*MES server*), Logiciel de journalisation (*Historian*), Serveur SCADA (SCADA server).

5. https ://github.com/StrideICS/StrideICS

2.2.2 Intégration des CVE dans MTMT

La base de données contient une section dédiée à la récupération des CVE. Dans cette section, l'utilisateur renseigne les composants pour lesquels il souhaite faire une recherche de CVE. Nous avons ajouté deux filtres (optionnels) sous forme de propriétés TOML pour fixer la plage de dates de publication des CVE et la criticité des CVE à récupérer. La gestion de la récupération des CVE et de leur intégration dans MTMT est gérée par un programme que nous avons développé. Ce programme requête la base de données de CVE NVD (National Vulnerability Database) du NIST selon les composants et les filtres renseignés dans la base de données. Puis, les CVE sont intégrées dans MTMT en leur associant une catégorie de menace STRIDE.

Notre programme requiert plusieurs informations pour effectuer la recherche de CVE. Tout d'abord, pour s'assurer de récupérer les CVE qui correspondent au composant modélisé, l'utilisateur doit renseigner l'identifiant unique assigné à ce composant par la base de données NVD (CPE) et par notre base de données (ID). Dans la base de données NVD, cet identifiant unique est basé sur le schéma de nommage CPE (Common Platform Enumeration), dont la liste complète est hébergée et maintenue par le NIST [52], et dans notre base de données l'identifiant est déterminé par le champ *ID* de la description du composant. Ensuite, l'utilisateur renseigne les filtres (optionnels) liées à la criticité et la plage de publication des CVE souhaités. La criticité est renseignée selon les valeurs qualitatives du système de notation CVSS v3.1 (Faible, Moyen, Haut, Critique). Ces valeurs qualitatives correspondent aux plages de scores CVSS suivantes : $Faible = 0.1 - 3.9$, $Moyen = 4.0 - 6.9$, $Haut = 7.0 - 8.9$, $Critique = 9.0 - 10.0$. Par exemple, si l'utilisateur spécifie qu'il souhaite récupérer les CVE comprenant une gravité haute ou critique, l'ensemble des CVE ayant un score CVSS compris entre 7.0 et 10.0 (10.0 étant la note maximale) seront récupérées. Ces informations à fournir pour la recherche de CVE ainsi que leur format sont listés dans la section *cveGathering* de la Liste 2.1.

Ensuite, notre programme intègre les CVE récupérées dans MTMT. MTMT utilise les catégories STRIDE pour la modélisation des menaces. Notre objectif est donc de faire correspondre les CVE aux catégories STRIDE. Pour cela, nous nous appuyons sur les travaux de Honkaranta et al. [31] qui proposent une correspondance, présentée dans le Tableau 2.3, entre les CWE et STRIDE. Comme le mentionnent Honkaranta et al, la base de données NVD fournit des liens entre CVE et CWE et donne donc la possibilité de faire correspondre les CVE à STRIDE (CVE vers CWE puis CWE vers STRIDE). Les CWE [50] sont des conditions logicielles et matérielles qui, dans certaines circonstances, pourraient contribuer à la création d'une vulnérabilité. La portée d'une CWE identifie le domaine d'application de la propriété de sécurité qui est violée et l'impact technique d'une CWE définit la conséquence de l'exploitation de cette faiblesse par un attaquant. Dans la base de données NVD, une CVE ne peut être associée qu'à une seule CWE. Cependant, plusieurs acteurs, comme le constructeur de l'équipement vulnérable, peuvent proposer une correspondante CVE/CWE. Dans notre méthode, nous nous basons uniquement sur l'association CVE/CWE proposée par le NIST.

STRIDE	**CWE/Impact Technique**	**CWE/Portée**
Usurpation d'identité	Obtenir des privilèges / Assumer une identité	Contrôle d'accès - Authentification
Falsification	Modifier des données	Intégrité
Répudiation	Cacher des activités	Non-Répudiation - Responsabilité
Divulgation d'informations	Lire des données	Confidentialité
Déni de service	DoS : exécution non fiable DoS : consommation des ressources	Disponibilité
Élévation de privilèges	Exécution de code ou de commande non-autorisé Contournement d'un mécanisme de protection	Confidentialité - Intégrité - Disponibilité - Contrôle d'accès Contrôle d'accès - Authentification

Table 2.3 – Correspondance entre CWE et STRIDE

Pour bien comprendre le Tableau 2.3, il est nécessaire de rappeler qu'une cyberattaque est la réalisation d'une menace par l'exploitation d'une ou plusieurs vulnérabilités. La correspondance faite dans le Tableau 2.3 entre l'impact technique *Obtenir des privilèges* de la CWE et la menace STRIDE *Usurpation d'identité* doit être comprise de la manière suivante. Un attaquant peut exploiter un manque d'authentification ou de contrôle d'accès (portée de la CWE) afin d'usurper l'identité d'un utilisateur (Menace) lui permettant d'obtenir des privilèges et/ou d'assumer une identité (impact technique de la CWE). Dans ce contexte, l'élévation de privilèges et l'usurpation d'identité ont une signification proche, mais elles se distinguent par leurs impacts.

La méthode de Honkaranta et al. [31] est applicable uniquement si une correspondance entre la CVE et une CWE existe, ce qui n'est pas toujours le cas. Dans la base de données NVD, les CVE pour lesquelles il n'est pas possible d'associer une CWE sont clairement identifiées et classifiées dans deux catégories :

1. **NVD-CWE-noinfo :** S'il n'y a pas suffisamment d'informations sur la CVE pour l'associer à une CWE, alors la CVE est associée à la catégorie *NVD-CWE-noinfo*. Cette catégorie représente en moyenne, sur la période 2013-2023, 15% des correspondances par année [6]

6. `https://nvd.nist.gov/vuln/search/statistics?form_type=Advanced&results_type=statistics&search_type=all&cwe_id=NVD-CWE-noinfo&isCpeNameSearch=false`

2. **NVD-CWE-other :** la base de données NVD utilise seulement un sous-ensemble de CWE pour classifier les CVE. S'il y a suffisamment d'information sur la vulnérabilité, mais que sa faiblesse n'appartient pas à ce sous-ensemble de CWE, alors la CVE est associée à la catégorie *NVD-CWE-other*. Cette catégorie représente en moyenne, sur la période 2013-2023, 3% des correspondances par année [7].

Ces deux catégories cumulées représentent à ce jour 23% de la base de données NVD, soit quasiment une CVE sur quatre. Pour combler cette lacune, de nombreux articles [18, 68, 41, 69] proposent des solutions basées sur l'IA (Intelligence Artificielle) afin de faire correspondre les CVE aux CWE. Cependant, aucune de ces solutions ne met à disposition son programme, son algorithme ou son code source pour l'utiliser. Nous proposons donc d'étendre la méthode de mise en correspondance des CVE avec les CWE proposée par Honkaranta et al. [31].

Pour les CVE qui n'ont pas d'attribution CWE, nous proposons de faire correspondre les CVE et STRIDE selon les métriques du système de notation CVSS (Common Vulnerability Scoring System) v3.1. la base de données NVD attribue un score à chaque CVE selon le système de notation CVSS qui contient notamment des métriques d'impact sur la confidentialité, l'intégrité et la disponibilité qualitativement évaluées comme None, Low ou High. La confidentialité, l'intégrité et la disponibilité sont respectivement les objectifs de sécurité visés par les menaces STRIDE de divulgation d'informations, de falsification et de déni de service. Nous mettons donc en correspondance les CVE, sans CWE associée, en fonction des métriques d'impact sur la confidentialité, l'intégrité et la disponibilité. Si une mesure d'impact est différente de None, nous associons la CVE à la menace STRIDE correspondante. Par exemple, si une CVE a le vecteur d'impact {Confidentialité :Faible ; Intégrité :Aucun ; Disponibilité :Haut}, nous l'associons à la fois à la catégorie *Divulgation d'informations* et de *Déni de service*.

Ces métriques ne donnent pas la possibilité d'associer les CVE aux catégories STRIDE d'*Usurpation d'identité*, de *Répudiation* et d'*Élévation de privilèges*, ce qui peut conduire à une correspondance partielle. Par exemple, WinCC CVE-2021-27384 "Une vulnérabilité a été identifiée [...] qui peut potentiellement entraîner l'exécution de code" a un impact élevé sur la confidentialité, l'intégrité et la disponibilité dans le vecteur de score CVSS. Cependant, si nous nous référons à la classification présentée dans le Tableau 2.3, cette CVE devrait être catégorisée comme une *Élévation des privilèges* car elle permet d'exécuter du code non autorisé. Malgré les approximations que comporte cette technique, elle permet d'intégrer l'ensemble des CVE dans MTMT afin qu'elles puissent être adressées par les équipes de cybersécurité.

Dans cette section, nous avons présenté notre méthode de modélisation des composants ICS et de leurs vulnérabilités dans MTMT. Dans la prochaine section, nous détaillons l'automatisation du flux de travail de MTMT et, donc, de l'identification des vulnérabilités des systèmes de contrôle industriel.

7. `https://nvd.nist.gov/vuln/search/statistics?form_type=Advanced&results_type=statistics&search_type=all&cwe_id=NVD-CWE-Other&isCpeNameSearch=false`

2.3 Automatisation du processus d'identification des vulnérabilités MTMT

Tel que présenté dans la section 2.1.2, *Microsoft Threat Modeling Tool* (MTMT) implémente la méthode STRIDE par interaction dans un processus en 3 étapes : (1) création d'un template, (2) création du modèle (DFD) du système, et (3) analyse du modèle. La Figure 2.7 propose une représentation graphique du processus de MTMT. Cette section introduit, pour chacune des étapes, les outils et les méthodes que nous avons conçus pour automatiser le processus d'identification des vulnérabilités de MTMT.

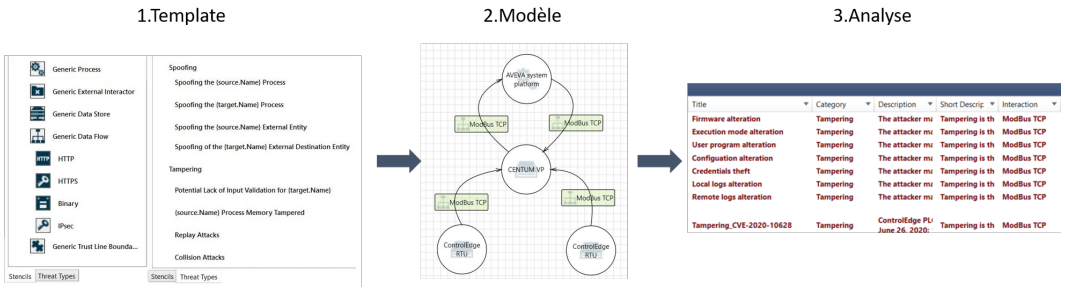


FIGURE 2.7 – Fonctionnement classique de l'outil MTM

2.3.1 Programme de génération de templates MTMT

Nous automatisons la première étape du processus de MTMT grâce à un programme qui traduit la base de données extensible TOML en template MTMT. Le programme comprend aussi la récupération et l'intégration des CVE dans le template MTMT ainsi qu'une automatisation de l'attribution d'un score de vraisemblance aux CVE. La Figure 2.8 présente l'intégration de ce programme dans le flux de travail MTMT.

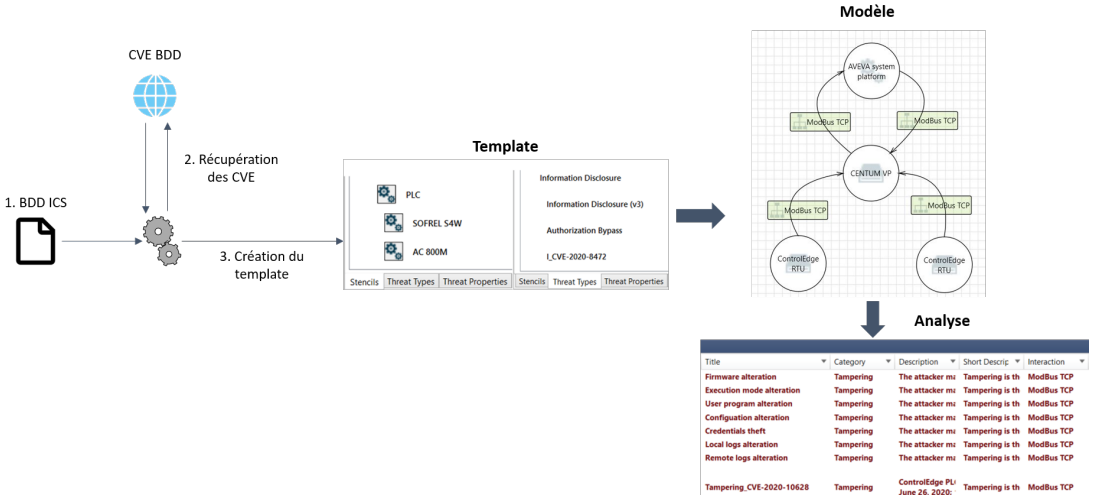


FIGURE 2.8 – Flux de travail de MTMT avec notre outil de génération de templates

Pour concevoir ce programme, nous avons dû faire une rétro-ingénierie d'un template MTMT pour comprendre sa structure et ses contraintes. Un template est un fichier XML pour lequel aucun schéma de description (fichier XSD) n'est fourni. Nous avons donc déterminé la structure du fichier à partir du template par défaut fourni par Microsoft et des messages d'erreurs retournés par l'outil MTMT.

Notre programme implémente aussi l'automatisation de la récupération des CVE depuis la base de données NVD (la base de données de CVE du NIST), leur association aux catégories STRIDE et leur intégration dans le template MTMT. la base de données NVD propose une interface de programmation (API) pour récupérer les CVE, mais possède une double contrainte liée au nombre de requêtes et à la plage de publication des CVE maximales autorisées. En effet, l'API n'autorise que 5 requêtes sur une fenêtre mobile de 30 secondes sans clé d'API [8] (50 avec une clé d'API) et autorise soit de récupérer toutes les CVE soit de filtrer les dates de publication des CVE sur une période de temps de 120 jours consécutifs maximum, c'est-à-dire que si nous souhaitons récupérer les CVE de l'année 2021, nous devons effectuer trois requêtes. Nous avons donc intégré à la fois des temps de pause pour gérer le nombre de requêtes maximal et un découpage de la plage de publication des CVE souhaitée par l'utilisateur en période de 120 jours pour respecter les contraintes de l'API. Une fois les CVE récupérées, elles sont modélisées en STRIDE puis intégrée dans le template sous le format *X_ CVE-NAME* où *X* identifie la catégorie STRIDE associée à la CVE (*S* pour usurpation d'identité, *T* pour falsification, etc.) suivi de l'identifiant de la CVE, par exemple *E_ CVE-2020-8472* (*E* pour élévation de privilèges).

Pour rappel, la première partie de notre méthode (cf. Chapitre 1) comprend une attribution d'une vraisemblance aux vulnérabilités identifiées. Notre programme intègre donc l'ajout d'une nouvelle colonne éditable dans la fenêtre d'analyse de MTMT (liste des vulnérabilités identifiées) pour attribuer une vraisemblance qualitative à chacune des vulnérabilités identifiées. L'attribution de la vraisemblance est automatisée dans notre programme pour les vulnérabilités issues de CVE.

2.3.2 Automatisation de la construction du modèle du système

La deuxième étape du flux de travail de MTMT propose de modéliser le système sous forme d'un diagramme de flux de données (DFD). Nous présentons dans cette partie un processus automatisable de construction du DFD d'un système de contrôle de commande industriel. Ce processus est basé sur la description des fonctions de communication des PLC écrite dans un langage de programmation appelé diagramme de blocs de fonction (Function Block Diagram - FBD) sous le format PLCopen (XML). Ces trois composantes, la description des fonctions de communication, le langage de programmation FBD et le format PLCopen sont standardisés dans la norme CEI 61131 [34, 35, 33].

8. Une clé d'API peut être obtenue gratuitement à l'adresse `https://nvd.nist.gov/developers/request-an-api-key`

Présentation de la CEI 61131

La CEI 61131 [34, 35, 33] est une série de 10 normes qui décrit les différents aspects des PLC. La CEI 61131-3 [34] (2013) spécifie la syntaxe et la sémantique d'une suite de langages de programmation pour développer des applications de contrôle logique et de traitement des signaux et des données pour les PLC. Cette norme comprend deux langages textuels, l'Instruction List (IL) [9] et le Structured Text (ST), ainsi que trois langages graphiques, le Ladder Diagram (LD), le Function Block Diagram (FBD), et le Sequential Function Chart (SFC). Dans ce chapitre, nous présentons uniquement le langage FBD, car les fonctions de communication des PLC sont normalisés sous forme de blocs de fonction qui sont les briques élémentaires du langage FBD.

Le FBD est un langage de programmation qui interconnecte des blocs de fonction avec des lignes de flux de signaux (signal flow lines). Les blocs de fonction sont représentés par des boîtes comprenant : le nom de la fonction qu'ils fournissent, un ensemble de lignes à sa gauche pour les entrées de la fonction, et un ensemble de lignes à sa droite pour les sorties. Dans la CEI 61131-3, il est simplement spécifié que le FBD est un langage de programmation des PLC qui est conforme, dans la mesure du possible, avec la norme CEI 60617-12. Un programme FBD peut comprendre plusieurs réseaux, c'est-à-dire des ensembles de blocs de fonction interconnectés autonomes, dont l'ordre d'exécution est déterminé par le constructeur du PLC. La Figure 2.9 présente un exemple d'un réseau FBD où le bloc de fonction *ADD* réalise une addition entre les valeurs des entrées *IN1* et *IN2* puis transmet le résultat à l'entrée *IN2* du bloc de fonction *GT*. *GT* vérifie si la valeur de *IN1* est supérieurs à *IN2* et transmet le résultat (vrai ou faux) à sa sortie *OUT*. Ce réseau correspond à l'implémentation de l'inégalité 2.1.

$$\text{LT\_VA\_HYD} + \text{BANDE\_MORTE\_VA\_HYD} < \text{HYD\_LT\_EAU\_BARRAGE\_PV} \tag{2.1}$$

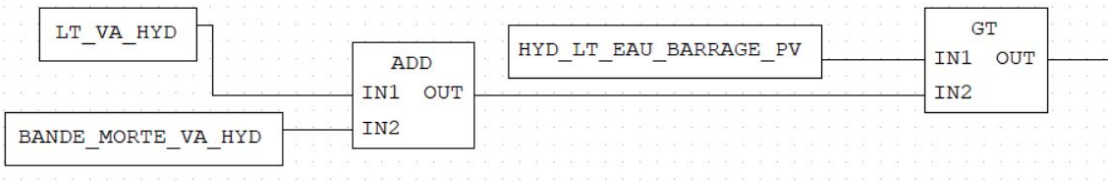


Figure 2.9 – Exemple d'un réseau FBD

Par ailleurs, la CEI 61131-5 [35] (2000) spécifie comment les PLC peuvent communiquer avec d'autres dispositifs, soit en tant que client, soit en tant que serveur. En outre, cette partie de la norme définit les fonctions de communication des PLC et leurs blocs fonctionnels (FB) correspondants. Cela permet au développeur de décrire les fonctions de communication d'un PLC à l'aide d'un programme FBD.

9. La norme définit ce langage comme *"déprécié et ne sera pas contenu dans la prochaine édition de cette norme"*

Enfin, la norme CEI 61131-10 [33] (2019) spécifie un format standard pour l'échange de programmes 61131-3 en XML, également connu sous le nom de format PLCopen. Ce format permet l'échange de programmes de PLC de manière standardisée, indépendamment de l'environnement de développement. Nous utilisons ce fichier, car il nous permet, théoriquement, de généraliser notre méthode de construction du DFD d'un système, indépendamment du fabricant et de l'environnement de développement du programme.

Construction du modèle MTMT

Dans ce travail, nous proposons un processus automatisable de construction du modèle dans MTMT à partir du programme des PLC. Pour rappel, MTMT fonctionne en trois étapes : (1) création d'un template (cf. section 2.1.2), (2) création d'un modèle du système sous forme d'un DFD que nous proposons d'automatiser dans cette section, et (3) analyse du modèle du système qui est déjà automatisé par MTMT. Nous avons vu dans la section précédente que la norme CEI 61131 propose un langage de programmation (CEI 61131-3), le FBD, qui permet de décrire les fonctions de communications d'un PLC (CEI 61131-5) dans d'un format normalisé (CEI 61131-10) nommé PLCopen. Nous allons maintenant montrer comment nous utilisons cette description des fonctions de communication des PLC pour construire le modèle MTMT du système.

Fonction de communication	**Bloc de fonction(s)**
Vérification du dispositif	STATUS, USTATUS
(Scrutée) Acquisition de données	READ
(Programmée) Acquisition de données	USEND & URCV, BSEND & BRCV
(Paramétrique) Contrôle	WRITE
(Verrouillé) Contrôle, Synchronisation entre les applications utilisateur	SEND & RCV
(Programmé) Rapport d'alarme	NOTIFY, ALARM
Exécution du programme et contrôle des E/S	None
Transfert du programme d'application	None
Gestion des connexions	CONNECT

Table 2.4 – Fonctions de communication des PLC et leurs blocs de fonction

Un PLC exécute un programme contenant à la fois le code de contrôle (gestion du procédé et de la sûreté) et une description de ses fonctions de communication. Ces fonctions de communication déterminent les échanges de données réalisées par un PLC avec le reste des équipements du système. D'après la norme CEI 61131-5, un PLC fournit jusqu'à huit fonctions à un système de contrôle en utilisant le sous-système de communication, dont seulement six ont des blocs de fonction spécifiés dans la norme CEI 61131-5, tel que présenté dans le Tableau 2.4. Ces blocs permettent de créer la description des fonctions de communication du PLC directement dans son programme. À titre d'information, les

Fonction de communication	**Flux de données des blocs de fonction**
Vérification du dispositif	1.STATUS $\xrightarrow{\text{Request}}$ Partenaire distant 2.STATUS $\xleftarrow{\text{(PHY, LOG, LOCAL)}}$ Partenaire distant, Partenaire distant $\xrightarrow{\text{(PHY, LOG, LOCAL)}}$ USTATUS
(Scrutée) Acquisition de données	1.READ $\xrightarrow{\text{VAR\_1 :VAR\_n}}$ Partenaire distant 2.READ $\xleftarrow{\text{RD\_1 :RD\_n}}$ Partenaire distant
(Programmée) Acquisition de données	USEND $\xrightarrow{\text{SD\_1 :SD\_n}}$ URCV, BSEND $\xrightarrow{\text{SD\_1}}$ BRCV
(Paramétrique) Contrôle	WRITE $\xrightarrow{\text{(VAR\_1, SD\_1) :(VAR\_n, SD\_n)}}$ Partenaire distant
(Verrouillé) Contrôle, Synchronisation entre les applications utilisateur	1.SEND $\xrightarrow{SD_{SEND}\_1 : SD_{SEND}\_n}$ RCV 2.SEND $\xleftarrow{SD_{RCV}\_1 : SD_{RCV}\_m}$ RCV
(Programmé) Rapport d'alarme	NOTIFY $\xrightarrow{\text{SEVERITY, EV\_ID}}$ Partenaire distant, 1.ALARM $\xrightarrow{\text{SEVERITY, EV\_ID}}$ Partenaire distant 2.ALARM $\xleftarrow{\text{ACK\_UP, ACK\_DN}}$ Partenaire distant

TABLE 2.5 – Flux de données des blocs de fonction

blocs de fonction reliés ensemble par le signe `&` dans le Tableau 2.4 doivent être tous les deux instanciés pour produire une instance de la fonction de communication. Par exemple, une instance d'un bloc de fonction *USEND* **et** une instance du bloc de fonction *URCV* fournissent une fonction *Acquisition de données programmée* d'un PLC.

Pour construire un DFD à partir de la description des fonctions de communication, nous avons tout d'abord identifié pour chaque bloc de fonction les flux de données qu'ils génèrent. Ensuite, pour un système donné, nous déterminons l'ensemble des flux de données générés par chaque PLC à partir des blocs de fonction de communication définis dans leur fichier PLCopen. Enfin, nous construisons un DFD selon l'ensemble des flux de données générés par les PLC du système. Sachant que les PLC sont, entre autres, une passerelle entre le réseau de terrain et le réseau de contrôle, le DFD que nous construisons permet de modéliser l'ensemble des communications de contrôle. En effet, le réseau de terrain interconnecte les capteurs, les actionneurs, les interfaces homme-machine locales et le PLC ; et le réseau de contrôle interconnecte la supervision (salle de contrôle) aux PLC ainsi que les PLC entre eux. Ainsi, toute la chaîne de contrôle est prise en compte dans le DFD que nous construisons. Le Tableau 2.5 présente les flux de données générés par chacun des blocs de fonction de la norme CEI 61131-5. Dans la suite, nous présentons en détail les flux de données générés par chaque bloc de fonction ainsi que des informations complémentaires nécessaires à la création du DFD.

Tout d'abord, pour créer un flux de données, nous devons identifier la source et la destination de l'échange. Tous les blocs de fonction listés dans le Tableau 2.5 possèdent un paramètre *ID* qui permet d'identifier le partenaire distant de la communication. Par exemple, dans une communication TCP/IP, ce paramètre sera l'adresse IP du partenaire distant. Par la suite, nous présentons chaque bloc de fonction de communication du Tableau 2.5 ainsi que les flux de données qu'ils génèrent :

Vérification du dispositif. La vérification de l'appareil est une fonction de communication spécifique qui permet d'envoyer ou de demander des informations sur l'état d'un appareil. Les blocs de fonction de la vérification du matériel possèdent, comme tous les autres, le paramètre *ID* pour identifier son partenaire de communication. Cependant, les blocs transmettent deux variables qui sont spécifiques à cette fonction, *PHYS* et *LOG*, qui respectivement fournissent des informations sur l'état physique et logique de l'appareil. Des informations additionnelles peuvent être transférées via la variable *LOCAL*. La Figure 2.10 présente les blocs de fonction de vérification de l'appareil :

- ***STATUS*** **:** Le bloc de fonction *STATUS* permet de demander les informations sur l'état d'un appareil distant, puis l'appareil distant répond avec ses informations d'état. Ce bloc génère donc deux flux de données, la demande et la réponse, avec les informations *PHYS*, *LOG* et *LOCAL*.
- ***USTATUS*** **:** Le bloc de fonction *USTATUS* place le PLC en tant que serveur, il attend que l'appareil distant lui envoie son état et donc génère un seul flux de données comprenant les informations *PHYS*, *LOG* et *LOCAL* transmises par l'appareil distant vers le PLC.

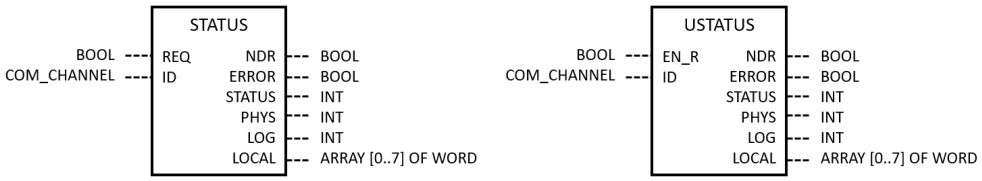


FIGURE 2.10 – Blocs de fonction pour la vérification de l'appareil

Afin d'identifier les flux de données émis par les autres blocs de fonction, nous utilisons trois paramètres qui identifient les données échangées par les différentes fonctions de communication. Le premier paramètre VAR_i détermine le nom d'une variable stockée dans un équipement distant dont la valeur va être lue ou modifiée par le bloc de fonction. Ensuite, le paramètre SD_i définit la donnée envoyée par le bloc de fonction à son partenaire distant. Enfin, RD_i identifie la dernière donnée que le bloc de fonction a reçue de son partenaire distant.

Acquisition des données. La Figure 2.11 présente le bloc de fonction d'acquisition des données scrutées (*READ*) et les blocs de fonction d'acquisition des données programmés (*USEND*/*URCV* et *BSEND*/*BRCV*). Ces deux fonctions permettent de récupérer des données depuis un partenaire distant soit faisant une requête (*READ*) soit par un envoi spontané du partenaire (*USEND*/*URCV* et *BSEND*/*BRCV*).

- **Acquisition des données scrutée**
  - ***READ*** **:** Le bloc *READ* génère deux flux de données : (1) demande de la valeur d'un ensemble de variables stockées dans le partenaire distant (*VAR_ 1* à *VAR_ n*), puis (2) réceptionne la réponse du partenaire distant contenant la valeur des variables.
- **Acquisition des données programmée**
  - ***USEND/URCV*** **:** La paire *USEND/URCV* génère un flux de données qui transmet la valeur des variables du bloc *USEND* (paramètres *SD_ 1* à *SD_ n*) aux paramètres de sorties *RD_ 1* à *RD_ n*) au bloc *URCV*. Le type de chacune des variables échangées est fixé à l'avance.
  - ***BSEND/BRCV*** **:** La paire *BSEND/BRCV* génère, elle aussi, un seul flux de données. Le bloc *BSEND* envoie toutes ses données dans une seule variable (paramètre *SD_ 1*) interprétée comme une séquence d'octets reçue par le paramètre *RD_ 1* du bloc *BRCV* où la taille de la séquence d'octet est fixée par le paramètre *LEN*.

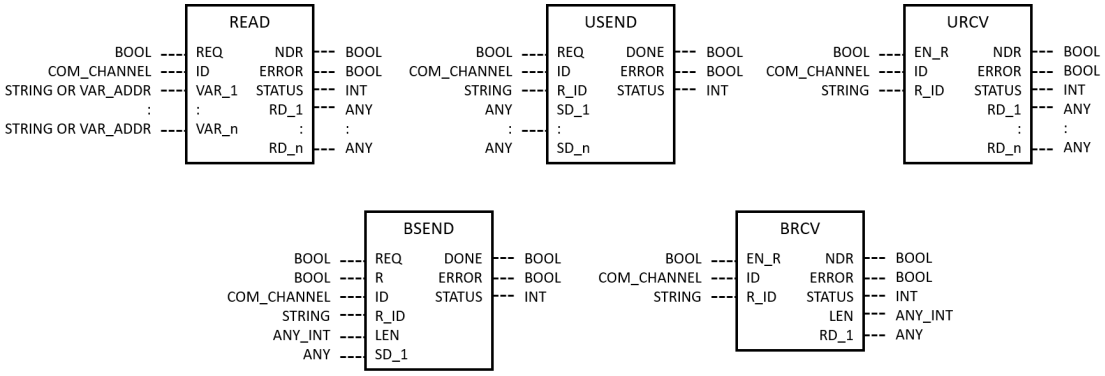


FIGURE 2.11 – Blocs de fonction pour l'acquisition des données

À titre d'information, les blocs de fonction œuvrant par paire possèdent un paramètre supplémentaire, le paramètre *R_ ID*. Ce paramètre permet à une paire de blocs d'une instance de s'identifier dans le cas où plusieurs instances d'une même fonction, plusieurs paires de blocs *USEND/URCV* par exemple, échangent dans un même canal de communication.

Contrôle et Synchronisation. La Figure 2.12 présente une description du bloc de fonction *WRITE* et de la paire de blocs *SEND/RCV* qui respectivement implémentent les fonctions de contrôle paramétrique et verrouillée.

- ***WRITE*** **:** Le bloc de fonction *WRITE* réalise un contrôle paramétrique. Il permet de modifier la valeur d'une variable sur un appareil distant grâce aux paires de paramètres *VAR_ i/SD_ i* où *VAR_ i* détermine le nom de la variable à modifier et *SD_ i* la nouvelle valeur à écrire dans cette variable distante.

- ***SEND/RCV*** **:** Le contrôle verrouillé s'effectue par l'intermédiaire d'une paire de blocs de fonction *SEND/RCV*. Le contrôle verrouillé se comporte comme un appel de procédure à distance, c'est-à-dire que le bloc *SEND* envoie des données (paramètres SD_{SEND}_1 à SD_{SEND}_n) au bloc *RCV* qui traite les informations reçues (le traitement peut être de tous types : une opération arithmétique, une écriture dans une variable, etc.) puis envoie les résultats (paramètres SD_{RCV}_1 à SD_{RCV}_m) au bloc *SEND*. En plus de fournir un contrôle verrouillé, la paire de blocs permet de synchroniser les programmes d'application des instances *SEND* et *RCV* car le bloc *SEND* doit attendre la réponse du bloc *RCV*.

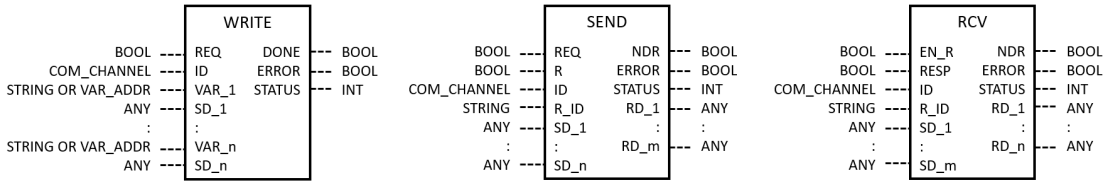


FIGURE 2.12 – Blocs de fonction pour la synchronisation entre les applications utilisateur et le contrôle

Rapport d'alarme. La fonction de rapport d'alarme permet d'envoyer un message d'alarme sans capacité d'acquittement avec le bloc de fonction *NOTIFY* et avec capacité d'acquittement en utilisant le bloc *ALARM*. Une description graphique de ces blocs est présenté dans la Figure 2.13. Mis à part le processus d'acquittement, les deux blocs envoient spontanément un message d'alarme à un partenaire distant dès lors que le PLC détecte un nouvel événement. Ce message se compose à minima d'un identifiant d'événement (EV_ID) et de sa sévérité ($SEVERITY$). Des informations complémentaires sur l'événement peuvent être fournies grâce aux paramètres SD_i (optionnel).

Le bloc de fonction *ALARM* fournit deux sorties booléennes qui permettent respectivement d'indiquer si le partenaire distant a acquitté (ou non) l'arrivée du nouvel événement (ACK_UP) et la prise en charge de l'événement (ACK_DN). Ce mécanisme permet de gérer uniquement l'acquittement de l'événement le plus récent. Le bloc de fonction *NOTIFY* génère donc un seul flux de données et le bloc *ALARM* génère deux flux de données (envoi de l'alarme et réception de l'acquittement).

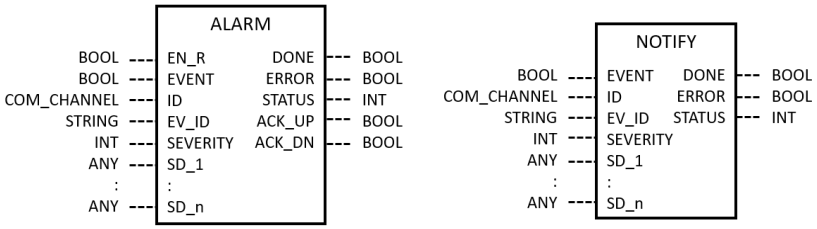


FIGURE 2.13 – Blocs de fonction pour le rapport d'alarme

Nous venons de présenter les flux de données générés par chaque fonction de communication des PLC. Ces flux de données, déterminés à partir des programmes des PLC, sont nécessaires à la construction d'un DFD du système, mais ne sont pas suffisants. En effet, un modèle MTMT représente les flux entre des éléments du système modélisés par des gabarits et gabarits dérivés (cf. Section 2.1.2). Ainsi, pour construire le DFD dans MTMT, nous devons associer à chaque composant du système, c'est-à-dire les PLC et les dispositifs avec lesquels ils communiquent, un gabarit ou un gabarit dérivé pour les modéliser dans MTMT. De plus, pour modéliser les flux de données, nous devons faire correspondre chaque paramètre *ID* (identification du partenaire distant de la communication) avec le dispositif auquel il fait référence. Enfin, nous devons associer chaque programme avec le PLC qui l'exécute. Nous considérons que ces informations sont données par une tierce personne dans un fichier que nous nommons *fichier de description* et qui suit la structure suivante :

- **Nom du dispositif :** Nom unique du dispositif, déterminé par l'utilisateur, pour identifier l'équipement dans le DFD.
- **Nom du gabarit ou du gabarit dérivé :** Le nom du gabarit ou du gabarit dérivé, comme défini dans le template (la librairie de composants), correspondant au dispositif.
- **Le programme exécuté :** Le chemin vers le fichier PLCopen du programme exécuté. Pour rappel, PLCopen est un format d'échange de programmes CEI 61131-3 XML standardisé dans la norme CEI 61131-10. Si le dispositif n'exécute pas de programme CEI 61131-3, ce champ prend la valeur `None`
- **Le paramètre *ID* :** La valeur du paramètre *ID* associée au dispositif.

Dans un modèle MTMT, les flux de données sont également représentés par des gabarits (ou gabarits dérivés) qui définissent le protocole de communication utilisé. En général, les blocs de fonction détaillés dans la CEI 61131-5 sont dérivés par les fabricants pour chaque protocole de communication avec un nom de bloc unique. Le nom du bloc de fonction nous permet donc de déterminer le gabarit que nous devons utiliser pour modéliser le flux de données.

Exemple

Pour illustrer le processus de construction d'un modèle MTMT, nous détaillons un exemple d'un bloc de fonction pour une communication entre deux PLC. Pour construire le modèle, nous commençons par créer le fichier de description comprenant les entrées suivantes :

- AP1 (gabarit : PLC, PLCopen : ap1.xml, ID : 192.168.1.1)
- AP2 (gabarit : PLC, PLCopen : ap2.xml, ID : 192.168.1.2)

Pour l'exemple, nous détaillons le bloc de fonction *SendTo* qui est une implémentation Schneider Electric du bloc de fonction *BSEND* pour UDP (User Datagram Protocol). Ce bloc est instancié dans le fichier PLCopen `ap1.xml` du PLC *AP1*. Nous supposons que le bloc de fonction *BRCV* est implémenté dans le PLC *AP2*. Le bloc *SendTo* contient les paramètres suivants :

- i_sPeerIP : Adresse de destination du message (ID) ;
- i_uiPeerPort : Port de destination du message (R_ID) ;
- i_pbySendBuffer : Adresse de départ du tampon contenant les données à envoyer (SD_1) ;
- i_udiNumBytesToSend : Nombre d'octets à envoyer (LEN)

Après avoir fait une analyse syntaxique (parsing) du fichier XML de l'automate, nous récupérons les valeurs suivantes :

- i_sPeerIP = 192.168.1.2 (AP2) ;
- i_uiPeerPort = 2000 ;
- i_pbySendBuffer = emplacement 16 bits 225 dans la mémoire (%MW225) ;
- i_udiNumBytesToSend : 1

Ce bloc de fonction envoie donc par UDP le premier octet stocké à l'emplacement 16 bits 225 de la mémoire d'*AP1* vers l'application située au port 2000 d'*AP2* (192.168.1.2 :2000). Nous pouvons donc créer un DFD comprenant deux PLC avec un flux de données UDP comme présenté en Figure 2.14.

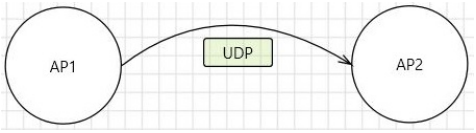


FIGURE 2.14 – DFD généré par l'exemple

Nous pouvons remarquer que toutes les informations n'apparaissent pas dans le DFD. Ces informations, rendues disponibles par cette étape, seront exploitées dans la section suivante pour identifier les variables vulnérables du système.

2.3.3 Analyse du modèle

La dernière étape du processus, l'analyse du modèle du système, est déjà automatisée par MTMT. Cependant, l'analyse du modèle nativement implémentée dans MTMT n'est pas suffisante pour réaliser la Partie 1 de notre méthode d'appréciation du risque. Pour rappel, dans le chapitre précédent (cf. chapitre 1), nous avons présenté une vue d'ensemble de notre méthode. La première partie de cette méthode (Partie 1) identifie les vulnérabilités du système pour : (1) alimenter la méthode d'identification des risques de cybersécurité pour la sûreté du système (Partie 2) et (2) déterminer la vraisemblance des vulnérabilités nécessaire à l'évaluation du risque (Partie 3). Dans cette section, nous présentons l'attribution de la vraisemblance aux vulnérabilités qui sera transmise à la Partie 3 ainsi que l'identification des variables vulnérables du système nécessaire à la Partie 2 de notre méthode.

Attribution du score de vraisemblance aux vulnérabilités

L'analyse du modèle dans MTMT génère une liste de menaces applicables à chacune des interactions du modèle (triplets source, destination, interaction). Cette énumération des menaces est effectuée grâce aux règles d'inclusions/exclusions des menaces et des propriétés/contraintes des gabarits que nous avons définis lors de la modélisation du système.

Le programme que nous avons développé (cf. Section 2.3.1), ajoute un nouveau champ nommé *Likelihood* (Vraisemblance) à la liste de menace pour associer à chaque vulnérabilité sa vraisemblance. Pour rappel, la vraisemblance est attribuée automatiquement pour les CVE. Pour les vulnérabilités personnalisées, le champ *Likelihood* permet à l'expert en cybersécurité de renseigner leur vraisemblance manuellement. MTMT comporte une fonction d'export des vulnérabilités au format CVS qui servira de fichier d'entrée à la Partie 3 de notre méthode.

Identification des variables vulnérables

La méthode d'identification des risques de cybersécurité pour la sûreté du système que nous avons conçue (Partie 2) nécessite d'identifier les variables vulnérables manipulées par le programme de contrôle. Dans la section 2.3.2, nous avons déterminé les flux de données générés par chaque bloc de fonction de communication ainsi que les paramètres échangés. Le programme des PLC contient la description des fonctions de communication, mais aussi le code de contrôle. Nous pouvons donc déterminer les variables du code de contrôle qui transitent dans chaque flux de données du système. Sachant qu'un flux de données relie deux composants par une flèche, nous pouvons aussi déterminer les composants du système qui manipulent ces variables. Nous exportons ces informations, la correspondance entre les gabarits du DFD (composants et flux de données) et les variables qu'ils manipulent, dans un fichier annexe à MTMT. Ce fichier nous permet, après l'analyse du modèle, d'identifier les variables vulnérables du code de contrôle qu'un attaquant peut manipuler. En effet, nous faisons l'hypothèse que si un gabarit est vulnérable, alors les données qu'il manipule sont aussi vulnérables.

2.4 Conclusion du chapitre

Dans ce chapitre, nous avons fait un zoom sur la première partie (Partie 1) de notre méthode d'appréciation du risque sûreté-sécurité. Cette partie contribue à :

1. Modéliser des composants ICS et leurs vulnérabilités dans MTMT.
2. Automatiser le flux de travail de MTMT pour identifier les variables vulnérables du système pour la Partie 2 de notre méthode d'appréciation du risque et la vraisemblance de ces vulnérabilités pour la Partie 3.

Dans la première partie de ce chapitre, nous avons introduit *Microsoft Threat Modeling Tool* (MTMT) ainsi que la méthode de modélisation de la menace sous-jacente STRIDE par interaction. Puis, dans une seconde section, nous avons présenté notre base de données extensible de composants et de vulnérabilités ICS ainsi que le processus que nous avons déployé pour construire cette base de données à partir des profils de protection de l'ANSSI. Ces profils de protection sont initialement élaborés pour fournir une référence de sécurité pour une obtention d'une Certification de Sécurité de Premier Niveau (CSPN). Notre modélisation peut donc aider les organisations à préparer une telle certification. Cependant, à ce jour, la CSPN est uniquement reconnu par la France (ANSSI)

et l'Allemagne (BSI) [10]. Nous souhaiterions dans un futur travail éclaircir l'état actuel et les perspectives du schéma de certification européen de cybersécurité des composants des systèmes d'automatisation et de commande industriels (Industrial Automation & Control Systems Components Cybersecurity Certification Scheme - ICCS) et notamment les liens entre la CSPN et l'ICCS. Nous souhaiterions aussi étudier la possibilité de construire des templates pour MTMT à partir de notre base de données, spécifique à un modèle d'attaquant. L'exigence ZCR 5.1 de la norme CEI 62443-3-2 nécessite d'identifier les menaces du système selon les caractéristiques suivantes : la source de la menace, la capacité ou niveau de qualification de la menace, le vecteur de menace et les actifs potentiellement affectés par la menace. L'idée serait de créer un modèle d'attaquant à partir de ces caractéristiques afin d'effectuer un premier filtrage des vulnérabilités de la base de données à inclure dans le template MTMT. Cela permettrait d'envisager plusieurs scénarios pour une même menace. Par exemple, pour modifier un firmware, un attaquant sur le réseau peut impacter une procédure de mise à jour (mise à jour illégitime), là où un utilisateur malveillant, ayant un compte sur l'équipement, peut contourner la politique de sécurité pour modifier le firmware. Une première étape serait de créer un premier ensemble de modèles d'attaquant et de menaces associées à partir des agents menaçants retenus par les profils de protection de l'ANSSI.

Dans la section 2.3, nous avons détaillé les méthodes et outils que nous avons développés pour automatiser l'identification des vulnérabilités du système et la détermination de leur vraisemblance avec MTMT. Pour cela, nous avons automatisé le flux de travail de MTMT : (1) création d'un template, (2) création du modèle du système, et (3) analyse du modèle. Nous avons réalisé la création du template à l'aide d'un programme que nous avons développé, qui traduit notre base de données en template MTMT et intègre les CVE des composants dans le template. Ensuite, nous avons détaillé notre processus de construction d'un DFD du système, représentation utilisée par MTMT, à partir d'un format standard de description des fonctions de communication des PLC (PLCopen). Enfin, la troisième étape, l'analyse du modèle, est automatisée par MTMT. Nous avons étendu cette analyse pour générer des informations nécessaires aux Parties 2 et 3 de notre méthode d'appréciation du risque. Premièrement, nous avons ajouté la propriété *Likelihood* (Vraisemblance) aux vulnérabilités dans l'interface d'analyse de MTMT. Deuxièmement, nous avons intégré dans notre processus de construction du DFD (étape 2 du flux de travail de MTMT), une corrélation entre les flux de données du système et les données qu'ils manipulent afin d'identifier les variables qu'un attaquant peut manipuler en exploitant les vulnérabilités du système identifié par l'analyse du modèle.

Une dernière perspective de ce travail serait d'étudier la mise en relation des types de menace définis dans MTMT et les exigences de la norme CEI 62443-3-3 [39]. La CEI 62443-3-3 décrit les contrôles techniques de cybersécurité à implémenter dans un système pour satisfaire à l'objectif de sécurité défini par l'évaluation du risque (CEI 62443-3-2). L'idée serait, à partir des règles *Inclure* et *Exclure* des types de menaces applicables au système (MTMT), de définir le niveau de sécurité atteint par le système, le conduit ou la zone et de déterminé les contrôles de sécurité manquant pour satisfaire le niveau

10. https ://cyber.gouv.fr/presentation-de-la-certification-de-securite-de-premier-niveau-cspn

de sécurité voulu. Nous pourrions aussi penser à développer une base de données qui associe des solutions correctives aux contrôles techniques afin de proposer à l'utilisateur un ensemble optimisé de solutions correctives qui atteint le niveau de sécurité voulu.

Dans le prochain chapitre, nous nous intéresserons en détail à la Partie 2 de notre méthode d'appréciation du risque qui identifie les conséquences des vulnérabilités, identifiées dans ce chapitre, pour la sûreté du système.

3 Identification du risque de cybersécurité pour la sûreté du système

La principale difficulté dans l'appréciation du risque sûreté-sécurité est l'équilibre entre le niveau de granularité, c'est-à-dire le niveau de détail inclus dans le modèle et la complexité du système. Afin de prendre en compte les propriétés de sûreté et de sécurité, nous souhaitons fournir, à la fois, une description très détaillée des cyberattaques pour mettre en œuvre les contre-mesures les plus pertinentes ; et une description précise du procédé pour identifier clairement les conséquences sur la sûreté. Cependant, une fine description de la cybersécurité et de la sûreté implique un espace d'état du modèle sûreté-sécurité ingérable (explosion combinatoire). Dans ce chapitre, nous présentons les détails de notre méthode d'identification du risque de cybersécurité pour la sûreté du système, méthode qui propose un compromis entre la modélisation de la sûreté et de la sécurité afin d'être applicable à des systèmes de grande taille.

Aperçu - Pour identifier le risque de cybersécurité pour la sûreté du système, nous proposons tout d'abord de modéliser le système de contrôle industriel à partir de la logique (programmes) des PLC (section 3.1). Ce processus de modélisation a l'avantage de représenter uniquement les états du système nécessaires au système de contrôle afin de maintenir la sûreté de l'infrastructure.

Ensuite, dans la section 3.2, nous présentons notre modèle de menace sûreté-sécurité. Ce modèle se base sur l'exploitation des boucles de contrôle des systèmes industriels. Nous verrons qu'un système de contrôle industriel garantit la sûreté par l'intermédiaire d'une boucle de contrôle qui, à partir d'informations provenant du procédé, fait évoluer sa commande. Cependant, les informations de la boucle de contrôle transitent au travers d'une infrastructure informatisée et par conséquent peuvent être l'objet de cyberattaques.

Dans la section 3.3, nous faisons converger notre modèle de menace et notre modèle du système afin d'identifier des scénarios d'attaque qui compromettent la sûreté du système. Nous poursuivons ensuite dans la section 3.4 par une présentation de l'état de l'art des méthodes intégrées d'appréciations du risque sûreté-sécurité que nous comparons avec notre travail du point de vue de la globalité du processus et de sa scalabilté (capacité à passer à l'échelle sur des systèmes complexes). Enfin, nous concluons ce chapitre (section 3.5) en rappelant les contributions mises en œuvre dans ce chapitre et nous présentons nos travaux futurs.

3.1 Modélisation du système selon les programmes des PLC

Un PLC agit selon un programme généralement décrit graphiquement (par exemple, SFC) qui définit l'état du système, les transitions entre ces états, ainsi que les commandes émises. Ce programme garantit la fourniture d'un service (par exemple, la production d'électricité) tout en maintenant la sûreté du système, à savoir la protection des personnes, des biens et de l'environnement contre les dangers liés à l'exploitation des procédés industriels. Les programmes des PLC offrent donc une modélisation du procédé comprenant uniquement les états et événements du système nécessaires au système de contrôle (fourniture du service et sûreté). De ce fait, nous utilisons dans ce travail le programme des PLC pour modéliser le système. Afin d'exploiter pleinement cette modélisation, nous proposons de traduire les programmes écrits en langages de programmation graphiques dans un modèle formel plus explicite, à savoir les automates à états finis, car les langages graphiques ne définissent pas explicitement tous les états et événements qu'ils modélisent. Nous commençons cette section par la présentation du langage de programmation graphique Sequential Function Chart (SFC) utilisé dans les PLCs et les machines de Mealy (automates à états finis) utilisés pour notre modélisation initiale (section 3.1.1). Ce premier processus de modélisation basé sur les machines de Mealy a une complexité combinatoire qui ne permet pas de modéliser des systèmes de grande taille (section 3.1.2). Nous décrivons donc plusieurs optimisations que nous avons réalisées afin de faire passer à l'échelle le processus de modélisation (section 3.1.3). Enfin, nous présentons l'intégration dans le modèle : (1) des formules de logique du premier ordre et (2) des transitions SFC écrites dans les langages Ladder Diagram (LD), Function Block Diagram (FBD) et ST (section 3.1.4).

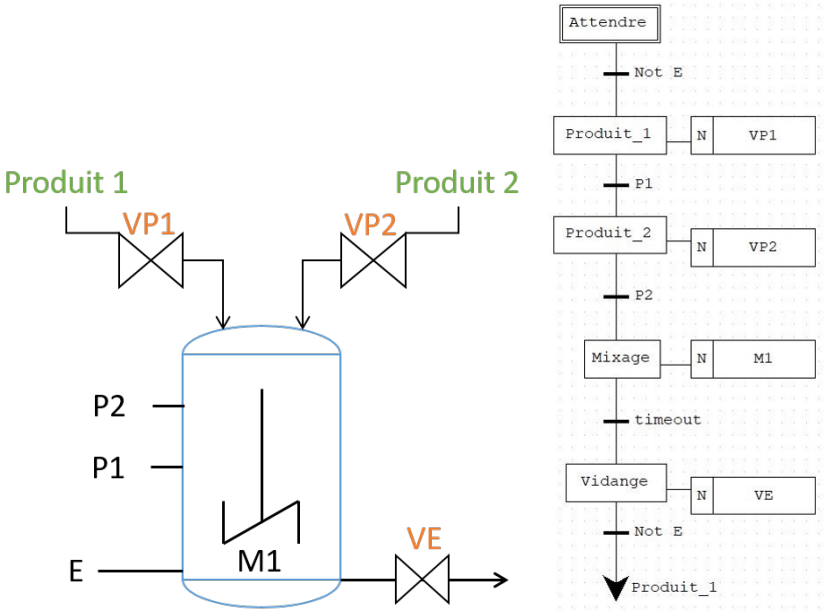
3.1.1 Contexte

Afin d'explorer de manière exhaustive le comportement des programmes des PLC, nous proposons de traduire les programmes écrits en Sequential Function Chart (SFC), langage spécifié par la norme CEI 61131-3, en une machine de Mealy grâce à un programme nommé TELOCO [53]. Cette partie propose de présenter le contexte, à savoir (1) le langage de programmation SFC et (2) la machine de Mealy.

Sequential Function Chart (SFC)

Le SFC est un langage de programmation défini par la norme CEI 61131-3 qui permet de structurer de manière séquentielle les fonctions de contrôle d'un programme PLC selon un ensemble d'étapes et de transitions. Une étape est une situation associée à des actions réalisées par le programme et une transition est un lien direct depuis et vers une ou plusieurs étapes. Une étape d'un SFC est soit active, soit inactive. Si l'étape est active, l'action associée est exécutée selon un quantificateur, sinon elle ne l'est pas. Sans rentrer dans les détails, chaque action possède un quantificateur qui permet de moduler son exécution par rapport à l'activité de l'étape associée, tels que : un retard (exécution de l'action n secondes après que l'étape soit active), une durée (exécution de l'action durant n secondes à partir de l'activation de l'étape), etc. Une transition permet

de transférer l'état actif de ses étapes amont à ses étapes aval. Pour cela, l'ensemble des étapes amont doivent être actives et une condition de transition doit être satisfaite (expression booléenne sur les variables du PLC). Les actions associées aux étapes et les conditions de déclenchement des transitions peuvent être décrites avec n'importe quel langage de programmation de la norme CEI 61131-3 tel que le Function Block Diagram (FBD) présenté dans la section 2.3.2. Un réseau SFC est une suite d'étapes et de transitions comprenant une étape initiale. Un programme SFC commence par activer l'étape initiale de chacun de ses réseaux, puis chaque réseau évolue en fonction de l'état du système. La Figure 3.1 présente un exemple de la gestion d'une cuve comprenant : une vue schématique du procédé (Figure 3.1a) et le programme SFC de gestion de la cuve (Figure 3.1b).



(a) Procédé (b) SFC

FIGURE 3.1 – Exemple d'une gestion de cuve

Le procédé de gestion de la cuve commence dans l'étape initiale (*Attendre*) qui attend que le réservoir soit vide (niveau du liquide inférieur au capteur de niveau *E*, c'est-à-dire *Not E*). Ensuite, le programme envoie une commande d'ouverture de la vanne *VP1* qui remplit la cuve avec le premier produit jusqu'à ce que le capteur *P1* soit *Vrai*. Quand le niveau de la cuve atteint *P1*, le programme ferme la vanne *VP1* (dans un programme SFC, si les variables de sorties ne sont pas explicitement Vraies, alors elles sont forcément Fausses) et ouvre la vanne *VP2* jusqu'à ce que le niveau *P2* soit atteint. La vanne *VP2* est ensuite fermée et les deux produits sont mélangés jusqu'à ce que le délai de temporisation soit écoulé (*timeout*). Enfin, la cuve est vidangée en ouvrant la vanne *VE* jusqu'à ce que la cuve soit vide (niveau du liquide inférieur au capteur de niveau *E*, soit *Not E*). Enfin, quand la cuve est vide, un nouveau cycle de remplissage commence.

Machine de Mealy

Une machine de Mealy est un transducteur fini (aussi appelé transducteur à états finis), c'est-à-dire un automate fini avec sorties dont les sorties dépendent à la fois de son état courant et de ses entrées. Le terme *fini* ou à *états finis* précise que le transducteur ou l'automate à un nombre fini d'états. Dans une machine de Mealy les transitions sont composées d'une entrée et d'une sortie. Formellement, une machine de Mealy est un 6-uplet comprenant :

- Un ensemble non vide d'entrées I_M
- Un ensemble non vide de sorties O_M
- Un ensemble non vide d'états S_M
- Un état initial $s_{InitM} \in S_M$
- Une fonction de transition $\delta_M : S_M \times I_M \longrightarrow S_M$
- Une fonction de sortie $\lambda_M : S_M \times I_M \longrightarrow O_M$

Pour faciliter la lecture, nous utiliserons le terme plus générique ***automate*** à la place du terme ***transducteur***. Pour rappel, les termes automate **fini** ou **à états finis** sont équivalents et seront utilisés indistinctement dans ce manuscrit.

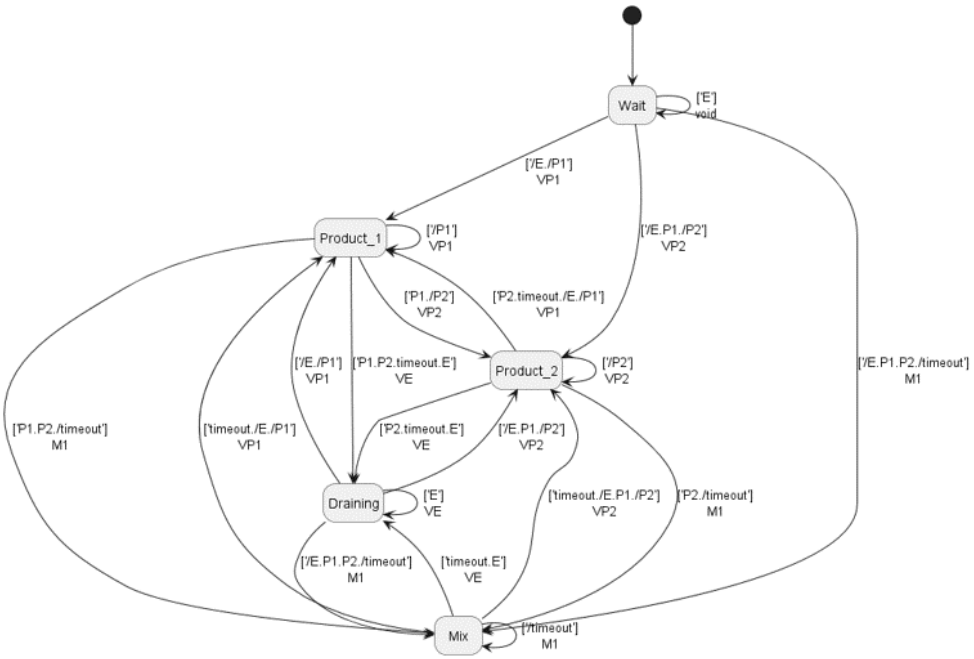


FIGURE 3.2 – Machine de Mealy du programme de gestion de la cuve

La Figure 3.2 présente la machine de Mealy du programme de gestion de la cuve introduit dans la Figure 3.1b. Dans cette représentation, entre crochets sont définies les conditions de transition et en dessous les sorties émises lors du franchissement de

la transition. Le cercle plein noir indique l'étape initiale de la machine de Mealy. Nous pouvons remarquer que la machine de Mealy comprend plus de transitions (20 transitions) que le programme SFC (5 transitions), car la machine de Mealy explicite les transitions entre tous les états.

Dans cette première partie, nous avons présenté le langage SFC et la machine de Mealy qui sont respectivement l'entrée et le résultat de notre modélisation initiale du système. Dans la partie suivante, nous présentons l'ensemble de notre processus initial de modélisation du système basé sur les programmes SFC des PLC.

3.1.2 Processus initial de modélisation du système

Notre processus initial de modélisation présenté dans la Figure 3.3 comprend trois étapes : (1) Modélisation des programmes des PLC, (2) Traduction des programmes des PLC en automate fini et (3) Simplification booléenne.

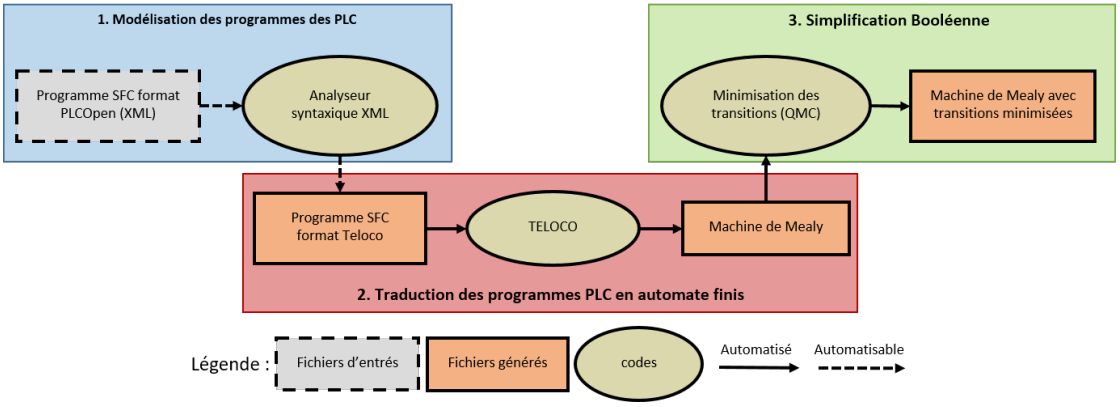


FIGURE 3.3 – Processus initial de modélisation du système

Étape 1 : Modélisation des programmes des PLC

L'utilisation de TELOCO [53], que nous détaillons dans l'Étape 2, demande de remplir manuellement un fichier de description du programme SFC qui est spécifique à l'outil. Afin d'automatiser la création de ce fichier, nous utilisons le format PLCopen (cf. Section 2.3.2) pour modéliser le programme des PLC. Ce format a l'avantage d'être standardisé et de représenter les programmes des PLC en XML. Ainsi, à l'aide d'un analyseur syntaxique, il est possible de créer automatiquement le fichier d'entrée de TELOCO.

Dans la Figure 3.4, nous présentons une partie du fichier XML PLCopen généré par le logiciel OpenPLC Editor [5] (à gauche) du programme SFC de gestion de la cuve (à droite) introduit précédemment dans la Figure 3.1. Dans cet exemple, nous montrons comment deux étapes reliées par une transition sont modélisées dans un fichier PLCopen. Les étapes *Attendre* et ***Produit_1*** sont modélisées par des balises *step* respectivement numérotées **1** et **5**. L'attribut *name* détermine le nom de l'étape. Les deux étapes sont reliées entre elles par une **transition**, dont la condition est ***Not E*** respectivement identifiés par les

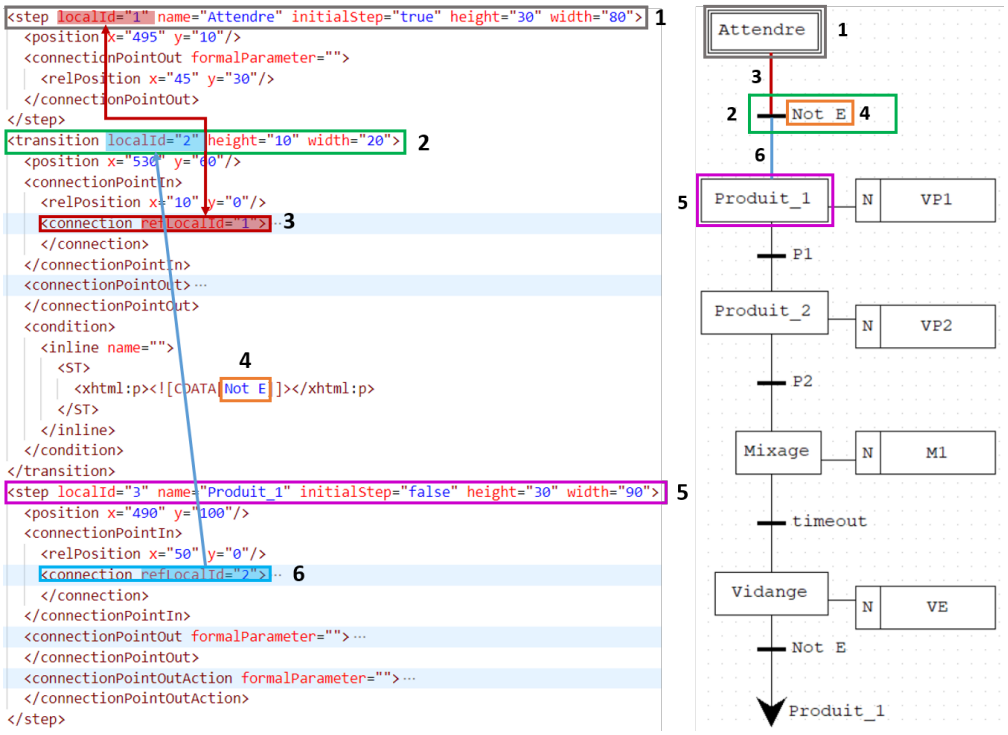


FIGURE 3.4 – Exemple d'analyse syntaxique de PLCopen

numéros **2** et **4** dans la Figure 3.4. Pour relier des éléments entre eux (étape, transitions, etc.), chaque élément possède un identifiant local. Par exemple, l'étape *Attendre* possède l'identifiant local *1* qui permet de préciser que la **transition** à une **connexion entrante** venant de l'étape *Attendre* (voir encadré **3**). La même notation est utilisée pour la **connexion** entre la **transition** et l'étape ***Produit_1*** présenté par l'encadré **6**.

Étape 2 : Traduction des programmes des PLC en automate fini

Pour traduire les programmes des PLC, nous nous appuyons sur les travaux de Provost et al. [57]. Dans ce travail, Provost et al. fournissent un outil appelé TELOCO qui automatise la traduction d'une spécification GRAFCET vers une machine de Mealy à des fins de test de conformité. Le GRAFCET est un langage de spécification avec une syntaxe proche du langage du SFC (Sequential Function Chart). La principale différence sémantique entre le GRAFCET et le SFC réside dans l'évolution des modèles qui, dans le cas du GRAFCET, est instantanée et, pour le SFC, est déterminée par le temps de cycle du PLC entre chaque lecture des entrées. Pour illustrer l'impact de cette différence, nous pouvons nous appuyer sur l'exemple du programme SFC de la Figure 3.1b rappelé dans la Figure 3.5. Dans le cas d'un GRAFCET, si l'étape *Produit_1* est active et que les variables d'entrée *P1* et *P2* sont vraies, alors le GRAFCET évolue instantanément de l'étape *Produit_1* à l'étape *Mixage*. Dans le cas d'un SFC ayant un temps de cycle

de 20 ms, le programme évoluera tel que présenté dans la Figure 3.5 où :

1. Les variables sont lues uniquement en début de cycle
2. Chaque étape est exécutée (même si *P1* et *P2* sont vrais, l'étape *Produit_2* est exécutée).

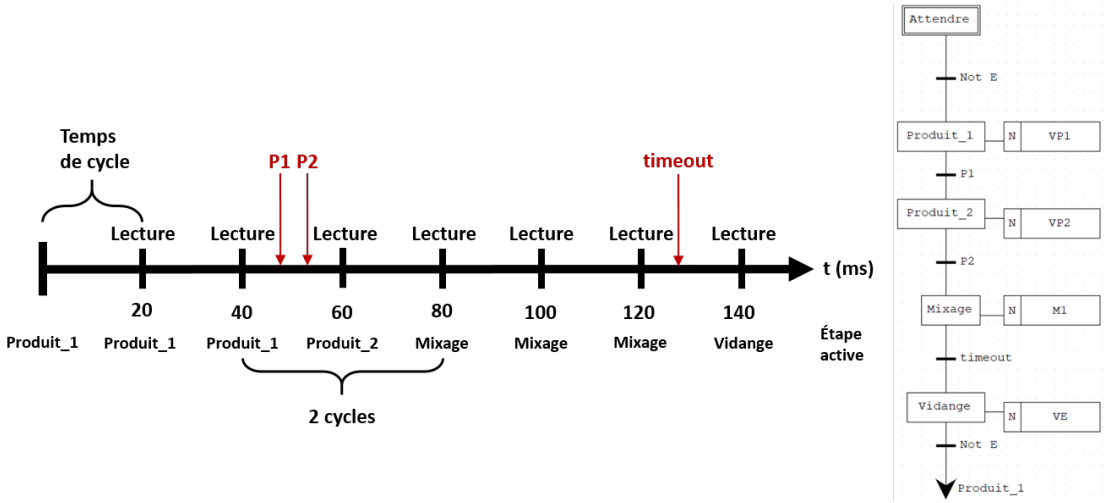


FIGURE 3.5 – Exemple d'évolution d'un SFC

Dans notre cas, nous considérons le programme SFC comme une spécification du comportement du PLC et nous supposons donc qu'un SFC fonctionne de la même manière qu'un GRAFCET. Cette hypothèse impacte uniquement la mise en œuvre des scénarios d'attaque qui est en dehors du périmètre de cette thèse. De ce fait, nous utilisons TELOCO pour automatiser la traduction d'un programme SFC vers une machine de Mealy.

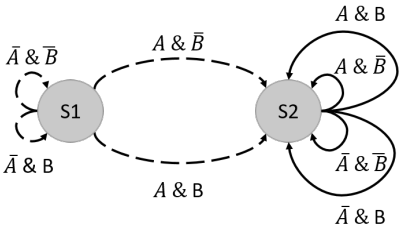


FIGURE 3.6 – Exemple d'une machine de Mealy complètement spécifiée

Dans cette méthode, la machine de Mealy construite est complètement spécifiée, c'est-à-dire que chaque état du graphe comprend une transition par combinaison d'entrées possibles ($|transitions| = 2^{|entrees|} \times |etats|$). Par exemple, pour une machine de Mealy avec 2 entrées (A et B) et 2 états ($S1$ et $S2$), la machine de Mealy complètement spécifiée comprend 8 transitions, 4 transitions ($2^{|entrees|}$) depuis l'état $S1$ et 4 transitions depuis l'état $S2$. Un exemple possible d'un tel automate est présenté dans la Figure 3.6. Afin

d'améliorer la lisibilité de la figure, nous omettons les sorties sur les transitions de la machine de Mealy et différencions les transitions provenant de $S1$ (lignes pointillées) et de $S2$ (lignes pleines).

Étape 3 : Simplification booléenne

Le programme TELOCO génère une machine de Mealy complètement spécifiée où toutes les transitions possibles sont représentées. Cependant, nous souhaitons représenter dans notre modèle non pas toutes les combinaisons possibles, mais la combinaison minimale qui permet de passer d'un état à un autre état. Cette combinaison minimale est la condition de transition. Par conséquent, dans notre première modélisation, nous modifions la machine de Mealy générée par TELOCO pour représenter les combinaisons d'entrées minimales pour transiter d'un état à un autre. Cette modification nécessite de calculer la fonction booléenne (condition de transition) minimale acceptant l'ensemble des transitions ayant le même état d'origine et le même état de destination. Par exemple, dans la Figure 3.6, il y a deux transitions qui font une boucle sur $S1$ (même état d'origine et même état de destination). Pour regrouper ces deux transitions, nous calculons l'expression minimale de la disjonction (OU logique) des expressions booléennes des deux transitions $((\neg A \wedge \neg B) \vee (\neg A \wedge B))$, car deux transitions depuis et vers un même état symbolisent deux moyens distincts (OU) pour passer d'un état à un autre. L'expression booléenne minimale de la disjonction de ces deux transitions est égale à $\neg A$ tel que présenté dans l'équation 3.1.

$$\begin{aligned} Eq &= (\neg A \wedge \neg B) \vee (\neg A \wedge B) && \# \; Factorisation \\ Eq &= \neg A \wedge (\neg B \vee B) && \# \; \neg B \vee B \text{ est toujours Vrai} \\ Eq &= \neg A \end{aligned} \tag{3.1}$$

Par conséquent, nous regroupons les deux transitions en une seule en utilisant l'expression booléenne $\neg A$. La Figure 3.7 présente la machine de Mealy de la Figure 3.6 avec ses transitions minimisées. Le symbole ε signifie que la condition de transition est toujours *Vrai*. Pour calculer cette minimisation booléenne, nous appliquons l'algorithme Quine-McCluskey (QMC) pour chaque groupe de transitions (même origine et même destination).

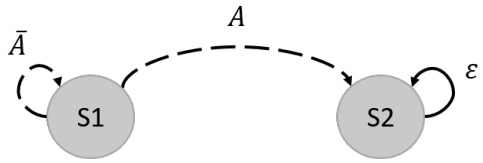


FIGURE 3.7 – Exemple de machine de Mealy avec des transitions minimisées

Cette première modélisation représente de manière explicite l'ensemble des états du système ainsi que les conditions minimales pour transiter d'un état à un autre. Cependant, ce premier processus de modélisation ne passe pas à l'échelle sur des systèmes de grande

taille en raison de l'algorithme de Quine-McCluskey, dont le temps de calcul évolue exponentiellement en fonction du nombre de variables d'entrée (problème NP-complet avec une évolution $\mathcal{O} = 3^n/\sqrt{n}$). Dans la suite de cette section, nous présentons les optimisations apportées à cette première modélisation afin de passer à l'échelle.

3.1.3 Optimisations de la modélisation

Pour optimiser notre modélisation initiale, nous avons étudié la construction de la machine de Mealy proposée par TELOCO. Pour construire la machine de Mealy à partir du programme SFC, Provost et al. [57] utilisent un modèle intermédiaire nommé automate à localisation stable (Stable Location Automaton - SLA). Cet automate détermine tous les états stables du programme, appelés localisations stables, et les évolutions entre ces états. Dans un SLA, une localisation est dite stable s'il existe une expression booléenne sur les entrées, nommée condition de stabilité, qui permet de maintenir cette localisation. Ainsi, lorsqu'une localisation est active et que sa condition de stabilité est vraie, le SLA reste dans cette localisation, sinon il évolue vers une nouvelle localisation.

Dans leur cas, Provost et al. [57] ont besoin d'un automate complètement spécifié et transforment donc le SLA en machine de Mealy complètement spécifiée. Pour rappel, un automate est complètement spécifié lorsque chaque expression booléenne sur les entrées est représentée par une transition depuis chaque état. Dans notre cas, au contraire, nous souhaitons modéliser sous forme de transition uniquement l'expression booléenne minimale sur les entrées pour passer d'un état à un autre de la même manière que dans le SLA. De ce fait, la première amélioration de notre modèle initial a été d'utiliser le SLA au lieu d'une machine de Mealy pour modéliser les programmes SFC. Nous avons donc amélioré le processus de modélisation initiale en supprimant les étapes de traduction du SLA en machine de Mealy et en supprimant la simplification booléenne (algorithme de Quine-McCluskey). Cette amélioration est résumée dans la Figure 3.8.

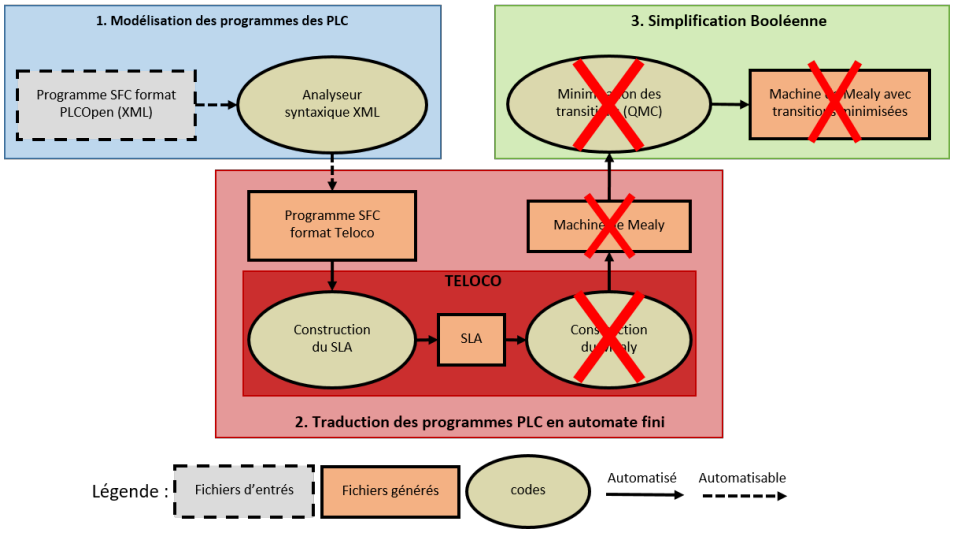


FIGURE 3.8 – Processus de modélisation avec le SLA

Malgré cette amélioration, notre modèle ne passe toujours pas à l'échelle. Pour comprendre la limite de cette modélisation, nous devons d'abord présenter les étapes de construction d'un SLA :

1. **Définition de la localisation initiale :** La construction du SLA commence par la définition de la localisation initiale qui représente l'état initial du SFC (activation des étapes initiales du programme). La suite de la construction du SLA est itérative, nous généralisons donc les étapes suivantes en utilisant le terme de localisation actuelle qui, dans la première itération, correspond à la localisation initiale.
2. **Identification des transitions activées :** Une transition est dite activée lorsque l'ensemble de ses étapes amont sont actives. Dans cette étape, toutes les transitions activées par la localisation actuelle sont déterminées, comme présenté par l'algorithme 1).

Algorithm 1: Identification des transitions activées

```
Data: SFC ; localisation actuelle
Result: Transitions activées
begin
    l_act ⟵ Localisation actuelle ;
    E_actives(l_act) ⟵ Étapes actives de la localisation l_act ;
    foreach transition ∈ SFC do
        E_amont(t) ⟵ Étapes en amont de la transition ;
        if E_amont(t) ∈ E_actives(l_act) then
            Ajoute transition in Transition_activees ;
        else
            suivant ;
        End foreach
    End
```

3. **Identification de l'ensemble des transitions simultanément franchissables :** Une transition est franchie à partir d'une localisation, si elle est activée (cf. étape 2) et si sa condition de transition (expression booléenne) est satisfaite. Deux transitions sont simultanément franchies si elles sont actives et si leurs conditions de transition sont satisfaites simultanément par une même expression booléenne sur les entrées. Enfin, l'ensemble des transitions simultanément franchissables comprend toutes les combinaisons de transitions franchissables qui sont satisfiables. Par exemple, si nous avons deux transitions franchissables $t1$ et $t2$, il existe 4 combinaisons possibles ($2^{|transitionsExecutables|}$) :

$$(1)\ t1 \wedge t2; \qquad (2)\ t1 \wedge \neg t2; \qquad (3)\ \neg t1 \wedge t2; \qquad (4)\ \neg t1 \wedge \neg t2 \tag{3.2}$$

pour lesquelles nous vérifions si elles sont satisfiables. Ainsi, chaque expression entre $t1$ et $t2$ qui est satisfiable sera intégrée dans l'ensemble des transitions simultanément franchissables. Si l'expression (2) $t1 \wedge \neg t2$ est satisfiable, cela signifie qu'il existe une expression booléenne sur les entrées qui permet de franchir la transition $t1$ sans franchir la transition $t2$.

4. **Identification des localisations stables :** Une localisation est stable s'il existe une expression booléenne sur les entrées $Cond$ qui maintient le programme SFC dans la même localisation. La localisation est donc dite stable, car elle n'évolue pas tant que l'expression $Cond$ est vraie. Pour chaque expression Ex (par exemple, $t1 \wedge t2$) incluse dans l'ensemble des transitions franchissables simultanément de la localisation actuelle :
   a) Franchir les transitions présentes dans l'expression Ex dans le SFC et déterminer la localisation d'arrivée l_{dest}, à savoir l'ensemble des étapes et actions actives du SFC après exécution de l'expression booléenne Ex par le programme.
   b) Déterminer l'ensemble des transitions activées par la nouvelle localisation l_{dest} (faire les étapes 1 et 2 pour la nouvelle localisation l_{dest}).
   c) S'il existe une expression booléenne sur les entrées $Cond$ qui :
      - satisfait l'expression Ex ($Cond \in Ex$) ; et
      - ne permet de franchir aucune des transitions activées de la localisation l_{dest} ($Cond \notin$ l'ensemble des transitions activées par la nouvelle localisation l_{dest}).

      alors la localisation l_{dest} est dite **stable** et devient alors un nouvel état du SLA (localisation stable). Cette nouvelle localisation comprendra une condition dite de stabilité égale à l'expression $Cond$.
   d) Sinon, la localisation l_{dest} est **transitoire** et les étapes 3 à 4.c sont refaites jusqu'à la découverte d'une localisation stable. Dans le cas où la séquence de déclenchement est infinie, comme un cycle d'évolution d'une situation transitoire à une autre, l'expression booléenne sur les entrées (l'évolution) est non stable et n'est pas modélisée dans le SLA.

La construction d'un SLA implique donc le calcul de l'ensemble des transitions franchissables simultanément (cf. étape 3 de la construction du SLA), à partir de l'état initial du SFC, pour découvrir de nouveaux états. Puis, à partir de ces nouveaux états, nous calculons à nouveau l'ensemble des transitions franchissables simultanément pour découvrir de nouveaux états et ainsi de suite jusqu'à ce que tous les états SFC et leurs évolutions aient été découverts.

Vérifier que deux transitions sont exécutables simultanément est un calcul ayant une complexité linéaire en fonction de la taille des entrées. Cependant, ce calcul doit être effectué pour chaque combinaison de transitions franchissables à partir de l'état actuel. Par exemple, pour 3 transitions, il existe 8 combinaisons possibles ($2^{|transitionsExecutables|}$) depuis chaque état, soit $|Etats| \times 8$ vérifications dans le pire des cas. Cette formule est généralisée par l'équation 3.3.

$$|Verifications| = \sum_{i=1}^{n} 2^{Te(i)} \tag{3.3}$$

où $Te(i)$ nombre de transitions franchissables depuis l'état i

L'évolution du nombre de vérifications évolue donc selon deux variables : le nombre de transitions franchissables depuis l'état i ($Te(i)$) et le nombre d'états ($|Etats|$). L'exécu-

tion en parallèle de réseaux SFC fait évoluer ces paramètres respectivement de manière linéaire (le nombre de transitions franchissables) et de manière exponentielle (le nombre d'états) :

- **Transitions franchissables :** Chaque réseau d'un programme SFC augmente le nombre de transitions franchissables d'au moins un. En effet, deux réseaux SFC impliquent au moins deux transitions franchissables (une pour chaque réseau SFC), l'exécution parallèle de trois réseaux SFC implique au moins trois transitions franchissables, et ainsi de suite. Sachant que l'algorithme a une complexité exponentielle 2^n, il augmente le nombre de vérifications de manière exponentielle.
- **Nombre d'états :** L'espace d'état (nombre d'états) d'un ensemble de réseaux SFC exécutés en parallèle est égal au produit de leurs espaces d'état respectif. Par exemple, l'espace d'état de 3 réseaux SFC ayant chacun 3 états est de 27 états ($3 \times 3 \times 3$). Par conséquent, le nombres d'état évolue lui aussi de manière exponentielle pour des réseaux SFC exécutés en parallèle comme présenté par l'équation 3.4.

$$|SLA(g)_{Etats}| = \prod_{i=1}^{n} |SFC(i)_{Etats}| \tag{3.4}$$

où $|SFC(i)_{Etats}|$ espace d'état du réseau SFC i ;
$|SFC(g)_{Etats}|$ espace d'état de l'ensemble des réseaux SFC du programme

L'exécution des réseaux SFC en parallèle implique donc deux algorithmes de complexité exponentielle sur le nombre de vérifications à effectuer et la taille de l'espace d'état. Afin de réduire, en partie, cette complexité nous utilisons le produit fort de graphes.

Produit fort de graphes

Un produit fort de deux graphes G et H est un graphe tel que l'ensemble des sommets est le produit cartésien des sommets de G et H, et que deux sommets distincts (ug, uh) et (vg, vh) sont adjacents si et seulement si (un exemple graphique est fourni dans la Figure 3.9) :

1. $ug = vg$ et uh et vh sont adjacents (une transition n'est déclenchée que dans le SFC H), ou
2. $uh = vh$ et ug and vg sont adjacents (une transition n'est déclenchée que dans le SFC G), ou
3. ug est adjacent à vg et uh est adjacent à vh (une transition est déclenchée à la fois dans le SFC G et le SFC H)

Ce nouveau graphe capture les transitions déclenchées uniquement dans le premier graphe (règle 2), uniquement dans le second graphe (règle 1), et dans les deux graphes simultanément (règle 3), tout comme un SLA pourrait le faire pour tous les SFC du système. Nous ajoutons une autre règle aux trois précédentes qui consiste à vérifier si les nouvelles transitions sont compatibles. En effet, deux programmes PLC peuvent partager une même variable qui doit avoir une valeur compatible, c'est-à-dire que la variable n'a

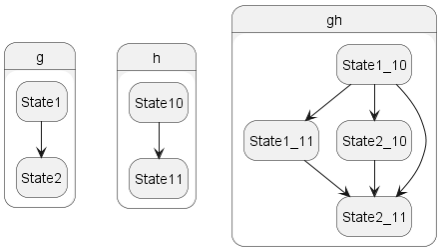


FIGURE 3.9 – Exemple d'un produit fort de graphes

pas une valeur opposée sur la nouvelle transition. Par exemple, dans la Figure 3.9, si les deux transitions (celles qui relient $State1$ à $State2$ et celles qui relient $State10$ à $State11$) partagent une variable avec des valeurs opposées, nous ne pourrons pas déclencher les deux transitions en même temps, et donc nous ne modéliserons pas cette transition dans le graphe final comme présenté dans la Figure 3.10.

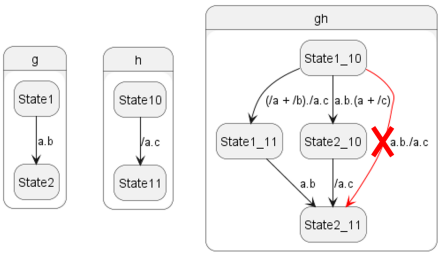


FIGURE 3.10 – Transitions incompatibles

Le produit fort de graphes permet de fusionner plusieurs SLA dans un seul graphe (SLA). Ainsi, nous pouvons construire le SLA du programme en traduisant, tout d'abord, chaque réseau SFC en SLA que nous fusionnons dans un second temps en réalisant un produit fort de graphes (des SLA des réseaux SFC), comme illustré dans la Figure 3.11. Cette technique permet de limiter le nombre de vérifications doublement exponentielles faites par TELOCO lors de la modélisation de réseau SFC en parallèle.

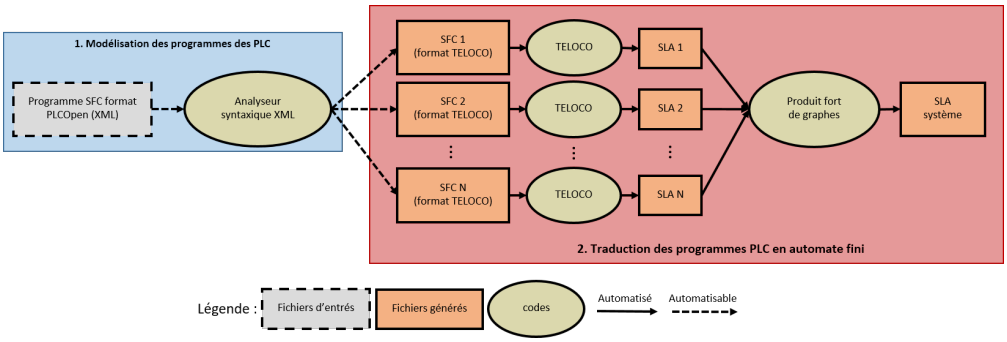


FIGURE 3.11 – Produit fort de SLA

Ce processus de modélisation permet de construire le même SLA que précédemment tout en diminuant le nombre de vérifications effectuées par TELOCO. Par exemple, pour 3 réseaux SFC exécutés en parallèle comportant chacun 5 états, TELOCO effectue à minima 3 375 vérifications pour créer le SLA des 5 réseaux directement (cf. équation 3.5) et à minima 120 vérifications pour créer les 5 SLA séparément (cf. équation 3.6).

$$
\begin{aligned}
&|Verification_{min}| = Etats \times Te \\
&|Verification_{min}| = 5^3 \times 2^3 = 3375 \\
&\text{où } Etats \text{ nombre d'états de l'espace produit ;} \\
&\quad Te \text{ nombre de transitions exécutables simultanément}
\end{aligned}
\tag{3.5}
$$

$$
\begin{aligned}
&|Verification_{min}| = \sum_{i=1}^{n} Etat(i) \times Te(i) \\
&|Verification_{min}| = (5 \times 2^3) \times 3 = 120 \\
&\text{où } Etats(i) \text{ nombre d'états du SFC } i \text{ ;} \\
&\quad Te(i) \text{ nombre de transitions exécutables simultanément du SFC } i
\end{aligned}
\tag{3.6}
$$

Cette amélioration reste cependant limitée, car le produit fort de graphes évolue toujours de manière exponentielle (cf. l'équation 3.4 du produit des espaces d'états). Par exemple, pour trois graphes comprenant respectivement 6, 7 et 8 états, le graphe final comprendra jusqu'à 336 états et 112 896 transitions. Par conséquent, nous proposons une dernière amélioration pour limiter l'espace d'état des réseaux SFC exécutés en parallèle, qui consiste à diviser le système en sous-procédés.

Division du système en sous-procédés

La dernière optimisation consiste à diviser le système global en sous-procédés indépendants (du point de vue du programme de contrôle) pour réduire la complexité. Cette division est réalisée en fonction des variables communes aux réseaux SFC des programmes de contrôle. Si un ou plusieurs réseaux SFC partagent une même variable, alors nous modélisons ces réseaux ensemble dans un même sous-procédé (produit fort de graphes). Notre processus final de modélisation du système est présenté dans la Figure 3.12.

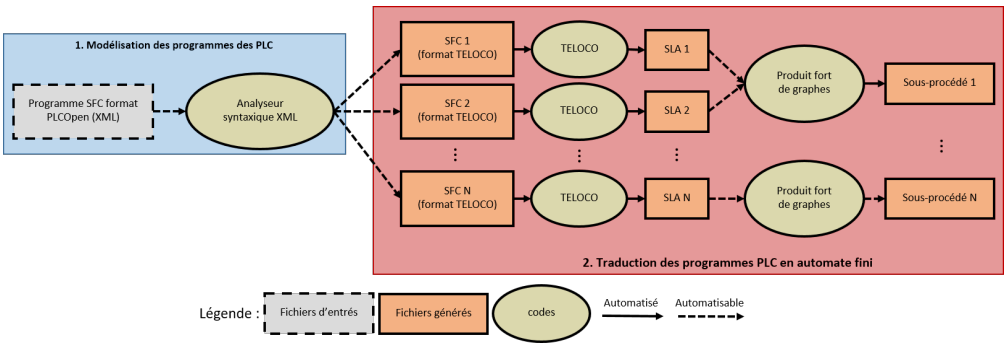


FIGURE 3.12 – Processus final de modélisation du système

Ce processus comprend une première étape qui traduit la description XML de chaque réseau SFC dans un format spécifique à l'outil TELOCO. Ensuite, TELOCO traduit chaque réseau SFC en SLA. Enfin, par l'intermédiaire de produits forts de graphes, nous regroupons les SLA partageant une variable commune dans un sous-procédé. Ainsi, nous modélisons le système global par l'intermédiaire de sous-procédés indépendants (du point de vue des programmes de contrôle).

Nous venons de présenter l'ensemble des optimisations réalisées pour que notre processus de modélisation initial puisse passer à l'échelle sur des systèmes complexes. Nous avons identifié que la principale limite de notre processus était liée à l'évolution exponentielle de la modélisation pour les tâches exécutées en parallèle. À partir de cette observation, nous avons proposé plusieurs améliorations qui réduisent le nombre d'étapes de notre processus de modélisation ainsi que l'espace d'état manipulé par chacune de ces étapes afin de modéliser des systèmes complexes dans un temps raisonnable.

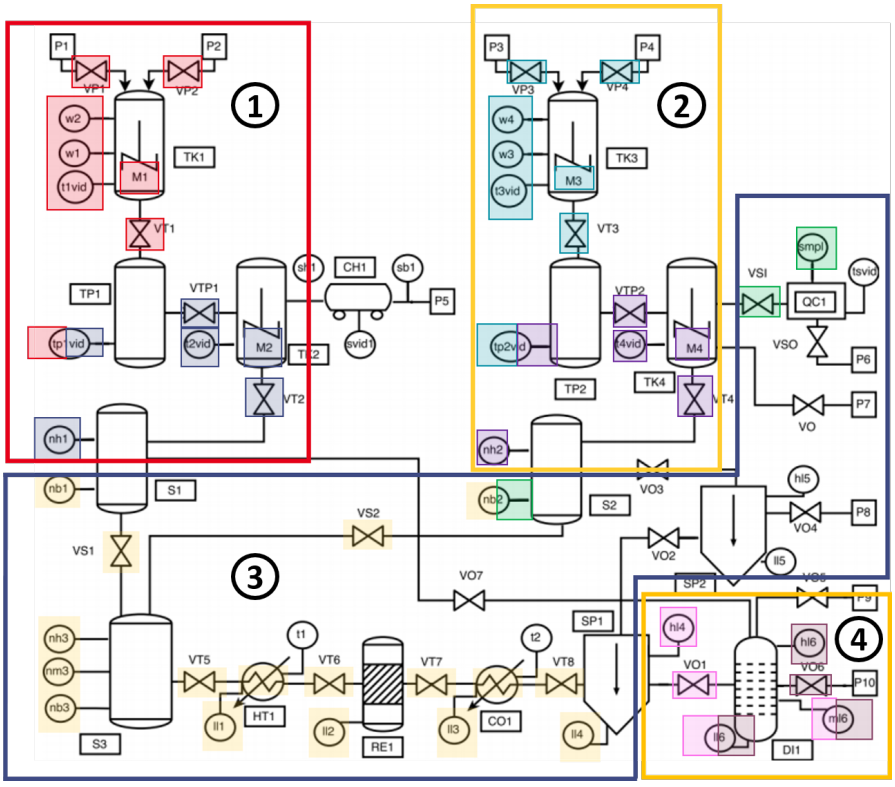


FIGURE 3.13 – Division du procédé chimique Tennessee Eastman par sous-procédés

La Figure 3.13 décrit le procédé chimique de Tennessee-Eastman [19] considéré comme une référence pour la modélisation et la simulation de systèmes industriels dont une description étendue est proposée dans l'Annexe A. Dans la Figure 3.13, nous avons identifié, par une même couleur, les capteurs et les actionneurs appartenant au même réseau SFC du programme de contrôle. Nous pouvons remarquer que, du point de vue des

programmes de contrôle, la plupart des réseaux SFC ne partagent aucune variable en commun. Par conséquent, modéliser ces réseaux SFC ensemble n'a pas d'intérêt et complexifie le modèle du système (espace d'état et temps de modélisation). Nous avons donc séparé le procédé global, en sous-procédé comprenant les réseaux SFC qui partagent une ou plusieurs variables (mesures de capteurs et/ou commandes d'actionneurs). Ces sous-procédés sont représentés dans la Figure 3.13 par des encadrés numérotés de 1 à 4. Nous développons dans cette section uniquement les temps de modélisation pour le procédé 3 afin de présenter les impacts des différentes optimisations que nous avons présentées. Le détail complet des temps de modélisation et des espaces d'états des modèles ainsi que les limites que nous avons atteintes sont disponibles dans l'Annexe A.

Le Tableau 3.1 présente le temps nécessaire pour modéliser le réseau SFC 3.1 (capteurs et actionneurs jaune), le réseau SFC 3.2 (capteurs et actionneurs vert) ainsi que le sous-procédé 3 (réseaux SFC 3.1 et 3.2) de la Figure 3.13 avec chacune de nos améliorations. À titre d'information, le SLA du sous-procédé 3 comprend 52 états et 2000 transitions, là où la machine de Mealy complètement spécifiée comprend 3 407 872 transitions (52×2^{16} transitions, où 16 est le nombre de variables d'entrée incluant 6 variables internes au PLC qui ne sont pas représentées dans la Figure 3.13). Les modèles présentés dans le Tableau 3.1 ont été générés avec un ordinateur portable équipé d'un processeur Intel(R) Core(TM)i5-8365U @1,60GHz-1,90GHz et de 16 Go de RAM.

Modèle	Méthode	Temps
SFC 3.1	Mealy + QMC	0 min 17 s 279 ms
	Mealy	0 min 0 s 140 ms
	SLA	0 min 0 s 42 ms
SFC 3.2	Mealy + QMC	0 min 0 s 374 ms
	Mealy	0 min 0 s 0 ms
	SLA	0 min 0 s 0 ms
SFC 3.1 + 3.2 (sous-procedé 3)	Mealy + QMC	32 min 50 s 0 ms
	PF Mealy + QMC	0 min 20 s 770 ms
	Mealy	0 min 18 s 931 ms
	SLA	0 min 1 s 264 ms
	PF SLA	0 min 0 s 201 ms

*QMC = Quine-McCluskey (minimisation des transitions)
*PF = Produit fort de graphes

Table 3.1 – Temps de Modélisation du Sous-procédé Bleu Selon les Optimisations

3.1.4 Intégrations complémentaires à la modélisation

TELOCO est construit initialement pour traduire des spécifications GRAFCET manipulant uniquement des variables booléennes [32], ce qui n'est pas le cas pour le SFC. Nous avons donc intégré dans notre modèle la prise en compte des autres types de variables définis dans la norme 61131-3 ainsi que les descriptions des transitions SFC écrites avec les langages de programmation LD, FBD et ST de la CEI 61131-3.

Modélisation de formules de logique du premier ordre

Le SLA de notre modélisation ne peut traiter que des variables booléennes. À l'exception des *Bits*, il n'est pas possible de modéliser directement dans le SLA des équations logiques du premier ordre (par exemple, $x > 1 \vee y < 5$) avec les types de données définis par la CEI 61131-3 :

- Entiers
- Réels
- Durées
- Bits
- Chaînes de caractère
- Caractères
- Dates

Pour cela, nous remplaçons la formule de logique du premier ordre en une formule propositionnelle. Par exemple, la formule $x > 1 \wedge y + x = 5$ sera remplacée par $A \wedge B$ où A représente sous forme d'une variable binaire le résultat de $x > 1$ (soit x est plus grand que 1 soit il ne l'est pas) et B représente sous forme d'une variable binaire le résultat de $y + x = 5$. Cette nouvelle proposition $A \wedge B$, nous permet de travailler avec une représentation binaire, utilisable dans un SLA, d'une expression contenant des types de données qui ne sont pas booléens.

Nous conservons le lien entre la variable binaire et la formule qu'elle représente, par exemple $plein : lvl > 48$, de cette manière si la variable *plein* est comprise dans un scénario d'attaque, nous pouvons définir le nom de la variable qu'un attaquant doit manipuler (lvl) ainsi que plage de valeurs de la variable lvl qui permet de réaliser l'attaque, soit $lvl > 48 \Leftrightarrow lvl \in [48; +\infty[$ où lvl est un entier naturel. La définition de la plage de valeurs des variables non booléenne est réalisée à l'aide d'un solveur SMT. Un solveur SMT est un logiciel permettant de fournir une décision à une instance d'un problème SMT, à savoir un problème de décision pour des formules du premier ordre combiné à des théories comme la théorie des nombres réels.

Modélisation des transitions en LD, FBD et ST

Le SLA ne permet pas de modéliser directement une description des transitions du SFC dans les langages de la CEI 61131-3. La norme CEI 61131-3 définit 5 cinq langages de programmation pour les PLC dont deux langages textuels, l'Instruction List (IL) [1] et

1. La norme définit ce langage comme *"déprécié et ne sera pas contenu dans la prochaine édition de cette norme"*

le Structured Text (ST), ainsi que trois langages graphiques, le Ladder Diagram (LD), le Function Block Diagram (FBD), et le Sequential Function Chart (SFC). Dans cette partie, nous proposons de modéliser les transitions SFC écrites avec les langages de programmations LD, FBD et ST. Nous n'incluons pas le langage IL, car il est considéré comme déprécié par la norme CEI 61131-3.

Une transition dans un programme SFC est représentée par une condition de transition binaire. De facto, les programmes LD, FBD et ST qui décrivent une transition SFC ne dérogent pas à la règle et sont écrits sous la forme d'une équation dont la sortie/le résultat est binaire. Nous traduisons donc les programmes LD, FBD et ST sous forme d'équations dont les sorties binaires servent à définir la condition de transition. Dans le cas d'une équation non booléenne, nous appliquons la modélisation des formules du premier ordre (cf. section 3.1.4) afin de manipuler une représentation binaire de l'équation. Pour illustrer cette modélisation, nous proposons un exemple pour chacun des langages.

FBD - Le langage FBD ainsi que l'exemple de la Figure 3.14 ont été introduits dans la section 2.3.2 du Chapitre 2. Pour rappel, Le FBD est un langage de programmation qui interconnecte des blocs de fonction avec des lignes de flux de signaux (signal flow lines). Les blocs de fonction sont représentés par des boîtes comprenant : le nom de la fonction qu'ils fournissent, un ensemble de lignes à sa gauche pour les entrées de la fonction, et un ensemble de lignes à sa droite pour les sorties. La Figure 3.14 présente un exemple d'un réseau FBD où le bloc de fonction *ADD* réalise une addition entre les valeurs des entrées *IN1* et *IN2* puis transmet le résultat à l'entrée *IN2* du bloc de fonction *GT*. *GT* vérifie si la valeur de *IN1* est supérieur à *IN2* et transmet le résultat (vrai ou faux) à sa sortie *OUT*. Ce réseau correspond à l'implémentation de l'inéquation 3.7.

$$\text{LT\_VA\_HYD} + \text{BANDE\_MORTE\_VA\_HYD} < \text{HYD\_LT\_EAU\_BARRAGE\_PV} \tag{3.7}$$

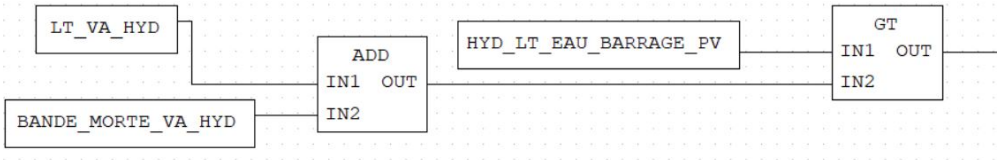


Figure 3.14 – Exemple d'un réseau FBD

Cette inéquation utilise des variables non booléennes (flottants). Ainsi, pour modéliser cette inéquation dans une transition SFC, nous remplaçons cette expression (formule de logique du premier ordre) par une variable binaire, par exemple *Plein*. Cette variable (*Plein*) représentera le résultat de l'inéquation, à savoir vrai ou faux.

LD - Le LD est un langage graphique d'équations booléennes combinant des contacts (entrées) représentés par des doubles barres verticales et des bobines (sorties) symbolisées par une paire de parenthèses. Un réseau LD fonctionne comme un schéma électrique

transmettant un flux d'énergie depuis la barre d'alimentation gauche à la barre d'alimentation droite. Chaque ensemble autonome transmettant un flux d'énergie s'appelle un échelon du réseau LD. Un réseau LD se lit échelon par échelon et de haut en bas.

Un contact transfère le flux d'énergie de sa gauche à sa droite selon la valeur de la variable booléenne associée au contact. Par exemple, dans la Figure 3.15, si la variable *Entrée1* est vraie, alors le contact laisse passer le flux d'énergie, sinon il le bloque. Une bobine copie la valeur du flux d'énergie de sa gauche à sa droite sans modification et stocke la valeur du flux d'énergie dans une variable. Il existe plusieurs types de contacts et de bobines qui permettent de modifier leur comportement initial.

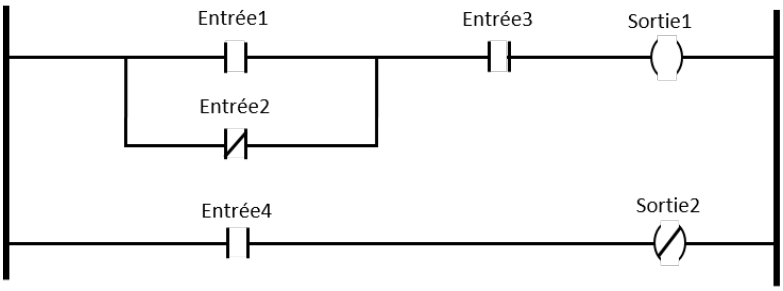


FIGURE 3.15 – Exemple d'un programme Ladder Diagram

La Figure 3.15 présente un réseau LD composé de deux échelons. Le quantificateur contenu dans le contact *Entrée2* du premier échelon transmet le flux d'énergie si *Entrée2* est Faux. Le quantificateur contenu dans la bobine *Sortie2* du deuxième échelon enregistre la valeur inverse du flux d'énergie. Nous avons donc les équations suivantes pour chacune des bobines de la Figure 3.15 :

1. $Sortie1 = (\text{Entrée1} \vee \neg\text{Entrée2}) \wedge \text{Entrée3}$
2. $Sortie2 = \neg(\text{Entrée4})$

Nous pouvons ainsi modéliser une condition de transition (équation avec une sortie booléenne) à partir des variables booléennes $Sortie1$ et $Sortie2$. Par exemple, nous pourrions déterminer une condition de transition égale à $Sortie1 \wedge Sortie2$. Cette modélisation est directement incluse dans le programme SFC avant même la traduction en SLA.

ST - Le ST est un langage de programmation textuel dérivé du Pascal dont la Figure 3.1 présente un exemple de programme.

```
1
2  PROGRAM Exemple_ST
3
4      VAR
5          A: BOOL;
6          B: BOOL;
7          C: REAL;
8          D: REAL;
9      END_VAR
10
11     A := NOT B AND (C <> D);
12
13 END_PROGRAM
```

Listing 3.1 – Exemple d'un programme ST

Un programme ST commence par la balise ouvrante *PROGRAM* suivie du nom du programme et se termine avec la balise fermante *END_PROGRAM*. Ensuite, les variables du programme sont définies ainsi que leur type (par exemple, $A : BOOL$;). Enfin, dans la Figure 3.1, nous déterminons une variable booléenne A qui enregistre le résultat de l'équation $\neg B \wedge (C \neq D)$. Cette variable A peut être utilisée telle quelle pour définir une condition de transition dans un SFC. Si la variable A est comprise dans un scénario d'attaque, nous utiliserons un solveur SMT afin de définir la plage de valeurs des variables réelles ($REAL$) C et D.

Dans cette première section, nous avons présenté notre processus de modélisation du système à partir des programmes des PLC. Dans la section suivante, nous introduisons notre modèle de menace sûreté-sécurité.

3.2 Modèle de menace sûreté-sécurité

Dans cette section, nous introduisons notre modèle de menace sûreté-sécurité. Pour cela, nous commencerons par présenter comment nous modélisons la sûreté par l'intermédiaire de fonctions de sûreté garanties par le système (section 3.2.1). Ensuite, nous présentons de manière macroscopique comment un attaquant peut profiter du transit des variables de contrôle au travers du système pour compromettre les fonctions de sûreté (section 3.2.2). Puis, nous précisons cette vue macroscopique afin de définir un modèle d'attaque (cyberattaques) qui permet de compromettre les fonctions de sûreté du système (section 3.2.3). Enfin, nous terminons cette section en présentant la création de notre modèle de menace sûreté-sécurité en corrélant le modèle d'attaque avec les violations des propriétés de cybersécurité qui rendent possible ces attaques (section 3.2).

3.2.1 Modélisation des fonctions de sûreté

Dans un système de contrôle industriel, la sûreté est garantie par les boucles de contrôle. Ces boucles de contrôle déterminent l'état du système à partir des capteurs et agissent sur le procédé grâce aux actionneurs. Ce lien entre un état du système et la commande appropriée peut être modélisé par une implication logique $A \implies B$ où A est un état du système et B la commande associée. Nous pouvons donc modéliser les fonctions de sûreté assurées par les boucles de contrôle sous forme d'une implication logique entre un **état limite** du système (A) et la **commande de protection** correspondante (B). Un état limite est une situation qui nécessite l'application d'une commande de protection pour ne pas évoluer vers un **état dangereux** ($A \wedge \neg B$) pour les personnes, les biens ou l'environnement. Dans une boucle de contrôle, les fonctions de sûreté sont implémentées dans les PLC dont le comportement est décrit par un programme. Dans la section précédente, nous avons modélisé le système à partir de ces programmes et donc ce modèle inclut la sûreté du système. Dans le SLA (modèle du système), les transitions traduisent les entrées du programme (états du système) en sorties (commandes) comme pour une

implication logique. Ainsi, les fonctions de sûreté du système seront représentées dans le SLA par des transitions.

1. Dans la suite de ce manuscrit, nous généralisons les explications à partir de l'implication logique $A \implies B$ où A représente un **état limite** du système et B la **commande de protection** associée.
2. L'implication logique $A \implies B$ est équivalente à l'expression booléenne $\neg A \vee B$. Nous utilisons chacune de ces notations indistinctement en fonction du contexte afin de faciliter les explications.
3. Par conséquent, une violation d'une fonction de sûreté (implication logique) correspond à sa négation $\neg(\neg A \vee B) \Leftrightarrow A \wedge \neg B$, en d'autres termes si une fonction de sûreté est violée $A \wedge \neg B$, alors le système est dans un **état dangereux**, c'est-à-dire qu'il est dans un état limite (A) sans que la commande de protection soit exécutée ($\neg B$)

Dérivation des fonctions de sûreté à partir du danger

Une fonction de sûreté a pour objectif d'interdire un état dangereux du système par l'intermédiaire de commandes et donc, afin de déterminer les fonctions de sûreté du système, nous devons d'abord identifier les états dangereux ainsi que leur commandes de protection respectives. Nous déterminons les états dangereux du système à partir d'une analyse préliminaire des risques (APR) préalablement réalisée par un expert en sûreté. À l'issue de l'APR, l'expert en sûreté modélise les états dangereux du système en fonction des variables des programmes de contrôle. Par exemple, si l'APR identifie un danger quand une cuve déborde, l'expert modélise cette situation selon les variables des programmes comme $(lvl > 48) \wedge (V1 \vee V2)$ où :

- ***lvl*** : est une variable associée à un capteur qui mesure la hauteur du liquide dans la cuve
- **48** : est la valeur maximum de la variable (niveau de liquide) avant débordement (Cuve pleine)
- ***V*1** : est une variable de sortie du PLC qui contrôle la vanne de remplissage (actionneur) $Vanne1$
- ***V*2** : est une variable de sortie du PLC qui contrôle la vanne de remplissage (actionneur) $Vanne2$

À partir de cette modélisation des états dangereux du système en fonction des variables des programmes de contrôle des PLC, nous pouvons dériver la fonction de sûreté que le système doit assurer. Un PLC peut agir sur le procédé uniquement avec ses sorties (commande aux actionneurs). Ainsi, pour interdire l'état dangereux, la fonction de sûreté doit envoyer une commande qui est la négation des variables de sorties de l'état dangereux. Par exemple, dans l'état dangereux $(lvl > 48) \wedge (V1 \vee V2)$, la fonction de sûreté interdit cet état en envoyant la **commande de protection** $\neg(V1 \vee V2)$ (négation des variables de sortie de l'état dangereux) lorsque le système est dans **l'état limite** $lvl > 48$. Nous pouvons donc construire une fonction de sûreté sous forme d'une implication logique ($A \implies B$) entre un état limite (A, dans l'exemple $lvl > 48$) et la commande de pro-

tection déduite de l'état dangereux (B, dans l'exemple $\neg(V1 \vee V2)$) tel que présenté par l'équation 3.8.

$$\begin{aligned}&\textbf{SI}\\&\text{État dangereux : } (lvl > 48) \wedge (V1 \vee V2)\\&\text{où } lvl > 48 \text{ est l'état limite } (A)\\&\\&\textbf{Alors}\\&\text{Sûreté : } (lvl > 48) \implies \neg(V1 \vee V2) \quad \Leftrightarrow \quad lvl > 48 \implies \neg V1 \wedge \neg V2\\&\text{où } \neg V1 \wedge \neg V2 \text{ est la commande de protection } (B)\end{aligned} \tag{3.8}$$

3.2.2 Compromission des fonctions de sûreté

L'objectif d'un attaquant est de compromettre une fonction de sûreté implémentée par les PLC afin de mettre le système en danger. Pour appliquer ces fonctions, le programme doit être en mesure de déterminer l'état du système par la lecture des variables d'état (entrées et variables internes du PLC) et d'agir sur le système par l'écriture de variables de sorties (commandes). Ces variables, que ce soit pour la lecture ou pour l'écriture, transitent au travers d'une architecture de communication informatisée (logiciel, réseau, matériel, etc.) afin d'être prises en compte par le programme (lecture) ou par les actionneurs (commandes). Dans le cas d'un système industriel, cette chaîne de transmission de l'information s'appelle une boucle de contrôle telle que présentée dans la Figure 3.16.

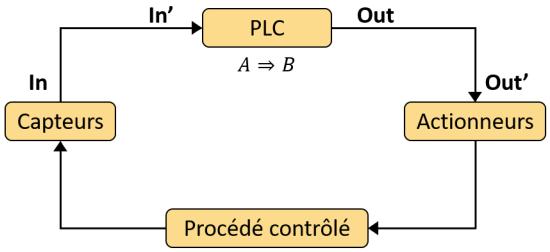


FIGURE 3.16 – Chaîne de transmission de l'information

Dans la Figure 3.16, In représente l'ensemble des valeurs de capteurs (courant, tension, température, etc.) transmises au programme du PLC. Ensuite, le PLC exécute son cycle : lecture des données reçues (In'), exécution de son programme (applique, par exemple, la fonction de sûreté $A \implies B$), et mise à jour des sorties (Out). Enfin, les sorties du PLC (les commandes) transitent jusqu'aux actionneurs qui exécutent la commande (Out'). Dans ce schéma, nous avons volontairement représenté les entrées et les sorties par deux variables différentes afin de distinguer les valeurs de variables émises (In et Out) et les valeurs de variables reçues (In' et Out'), car un attaquant peut profiter de

cette différence, entre les variables émises et reçues, afin de compromettre les fonctions de sûreté du système, tel que présenté dans la Figure 3.17.

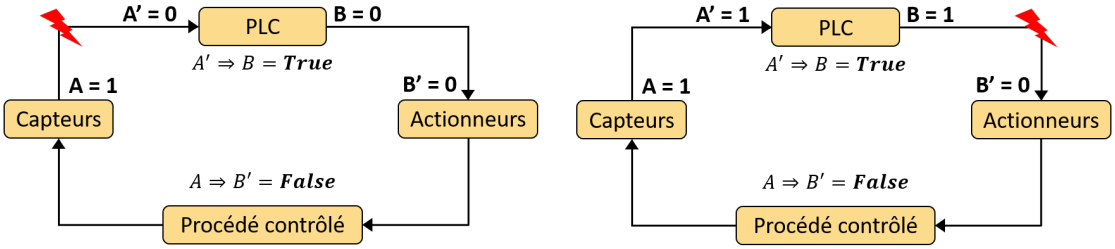


FIGURE 3.17 – Compromission de la chaîne de transmission d'information

La Figure 3.17 présente la manière dont un attaquant peut manipuler la valeur émise d'un état limite (A) ou d'une commande de protection (B) afin de compromettre une fonction de sûreté. Dans cette figure :

- $\boldsymbol{A}$: représente la valeur de l'état limite **émise** par les capteurs. Si $A = 1$ le système est dans un état limite, sinon il ne l'est pas.
- $\boldsymbol{A'}$: représente la valeur de l'état limite **reçue** par le PLC. Si $A' = 1$ le programme du PLC déduit que le système est dans un état limite.
- $\boldsymbol{B}$: représente la valeur de la commande de protection **émise** par le PLC. Si $B = 1$ la commande de protection est émise, sinon elle ne l'est pas.
- $\boldsymbol{B'}$: représente la valeur de la commande de protection **reçue** par le PLC. Si $B' = 1$ la commande de protection est exécutée par les actionneurs, sinon elle ne l'est pas.

Afin de compromettre les fonctions de sûreté du système, un attaquant peut profiter du transit des données au travers de la boucle de contrôle pour manipuler les variables d'entrées/sorties du PLC et ainsi violer la fonction de sûreté à l'échelle du système ($A \implies B'$, capteurs et actionneurs) sans violer la fonction de sûreté implémentée dans le PLC ($A' \implies B$, entrées et sorties du PLC). Pour cela, un attaquant peut manipuler : soit (1) les entrées du PLC (cf. schéma à gauche de la Figure 3.17), soit (2) les sorties du PLC (cf. schéma à droite de la Figure 3.17).

3.2.3 Modèle d'attaque

Pour comprendre toutes les spécificités de ces manipulations, nous devons d'abord introduire le Tableau 3.2 qui présente toutes les évolutions possibles des fonctions de sûreté ($A \implies B$) depuis un instant t ($A_t \implies B_t$) vers un instant $t+1$ ($A_{t+1} \implies B_{t+1}$). Pour alléger le Tableau 3.2, nous présentons uniquement les expressions booléennes où $A_{t+1} = 1$ car, selon la table de vérité de l'implication logique $\neg A \vee B$, B (la commande de protection) n'est conditionné par A (l'état limite) si et seulement si $A = 1$. Pour rappel, nous qualifions un état du système où $A = 1$ d'état limite, car si la commande de protection n'est pas active ($B = 1$) alors le système est en danger.

Dans la première partie du Tableau 3.2, nous avons regroupé ensemble les évolutions qui ne sont pas réalisables par un PLC, car elles violent la fonction de sûreté à l'instant t

A_t	B_t	$A_t \implies B_t$	A_{t+1}	B_{t+1}	$A_{t+1} \implies B_{t+1}$	Évolutions
0	0	Vrai	1	0	**Faux**	Remplissage → Débordement
0	1	Vrai	1	0	**Faux**	EtatX → Débordement
1	0	**Faux**	-	-	-	Débordement
1	1	Vrai	1	0	**Faux**	Plein → Débordement
0	0	Vrai	1	1	Vrai	**Si** $\uparrow A$ **Alors** $\uparrow B$
0	1	Vrai	1	1	Vrai	**Si** $\uparrow A$ **Alors** $B \rightarrow B$
1	1	Vrai	1	1	Vrai	**Si** $A \rightarrow A$ **Alors** $B \rightarrow B$

$\uparrow$ = Front montant
$\rightarrow$ = Passage de X_t à X_{t+1} ($X_t \rightarrow X_{t+1}$)

TABLE 3.2 – Évolutions des fonctions de sûreté

ou à l'instant *t+1*. Par exemple, si nous définissons la fonction de sûreté *Cuve pleine (A)* ***implique*** *vanne de remplissage fermée (B)* pour protéger le système d'un débordement de la cuve, alors la première ligne modélise :

- à l'instant t : que la cuve est en phase de remplissage. La cuve n'est pas pleine ($A_t = 0$) et la vanne de remplissage est ouverte ($B_t = 0$) ; et
- à l'instant $t + 1$: que la cuve déborde. La cuve est pleine ($A_{t+1} = 1$) et la vanne de remplissage est ouverte ($B_{t+1} = 0$).

Cet état dans lequel la cuve déborde ($A_{t+1} = 1$ et $B_{t+1} = 0$) n'est pas possible, car il viole la fonction de sûreté "*Cuve pleine* ***implique*** *vanne de remplissage fermée*". À contrario, la deuxième partie du Tableau 3.2 rassemble les évolutions qui ne violent pas la fonction de sûreté, que ce soit à l'instant t ou à l'instant $t + 1$. Cela nous permet de définir trois évolutions possibles de la fonction de sûreté :

1. **Déclenchement (Si $\uparrow A$ Alors $\uparrow B$) :** Si la cuve devient pleine et que la vanne de remplissage est ouverte alors fermeture de la vanne. L'apparition de l'état limite ($\uparrow A$) ***déclenche*** la commande de protection ($\uparrow B$).
2. **Verrouillage (Si $\uparrow A$ alors $B \rightarrow B$) :** Si la cuve devient pleine et que la vanne de remplissage est fermée, alors maintenir la vanne fermée. L'apparition de l'état limite ($\uparrow A$) ***verrouille*** la commande de protection ($B \rightarrow B$). Ce cas n'a pas de réalité du point de vue du contrôle, car il n'est pas possible que le système rentre dans un état limite si la commande de protection est active. Cette évolution ne sera donc pas étudiée dans la suite de ce manuscrit.
3. **Maintien (Si $A \rightarrow A$ alors $B \rightarrow B$) :** Si la cuve reste pleine, alors maintenir la vanne de remplissage fermée. La persistance de l'état limite ($A \rightarrow A$) ***maintient*** la commande de protection ($B \rightarrow B$).

À partir de ces évolutions, il devient évident que les manipulations de l'attaquant pour compromettre le déclenchement ou le maintien de la fonction de sûreté n'auront pas le même objectif au niveau du contrôle. En effet, pour compromettre le déclenchement de la fonction de sûreté, l'attaquant doit empêcher l'exécution de la commande de protection par l'actionneur (***attaque par blocage***) ; et pour compromettre le maintien de la fonction

de sûreté, l'attaquant doit forcer la désactivation (passage de $B_t = 1$ à $B_{t+1} = 0$) de la commande de protection (***attaque par forçage***). Ces attaques peuvent, comme présenté dans la Figure 3.17, être réalisées en manipulant soit les entrées du PLC (Manipulation du programme), soit les sorties du PLC (Manipulation des commandes).

1. **Manipulation des commandes :** La manipulation des commandes est assez triviale. L'attaquant va directement agir sur les commandes pour :
   - bloquer l'envoi de la commande de protection du PLC lors du déclenchement d'une fonction de sûreté (Si $\uparrow A$ Alors $\uparrow B$). Par exemple, un attaquant peut bloquer la commande de fermeture de la vanne de remplissage quand la cuve devient pleine.
   - envoyer une commande malveillante ($B_{t+1} = 0$) directement à l'actionneur quand la fonction de sûreté est maintenue (Si $A \rightarrow A$ alors $B \rightarrow B$). Par exemple, un attaquant peut envoyer une commande d'ouverture de la vanne de remplissage lorsque la cuve est pleine.
2. **Manipulation du programme du PLC :** Le second moyen d'agir est de manipuler le programme du PLC afin d'empêcher une commande de protection ou de forcer une commande malveillante.
   - Pour bloquer une commande de protection, l'attaquant doit empêcher qu'un PLC lise un événement que nous appelons critique. Un événement critique est un changement d'état du système depuis un état stable (non limite) vers un état limite ($\uparrow A$). Cet évènement est critique car s'il n'est pas lu par le PLC, alors la commande de protection ne sera pas déclenché et par conséquent le système évoluera vers un état dangereux ($A \wedge \neg B$). Par exemple, dans la Figure 3.18, l'événement critique est le passage d'un état non plein de la cuve (schéma de gauche) à l'état plein (schéma central). Si l'attaquant empêche cet événement critique d'être lu par le PLC (schéma central, différence entre la valeur du capteur et la valeur lue par le PLC), alors le PLC continuera de remplir la cuve jusqu'à son débordement (schéma de droite).

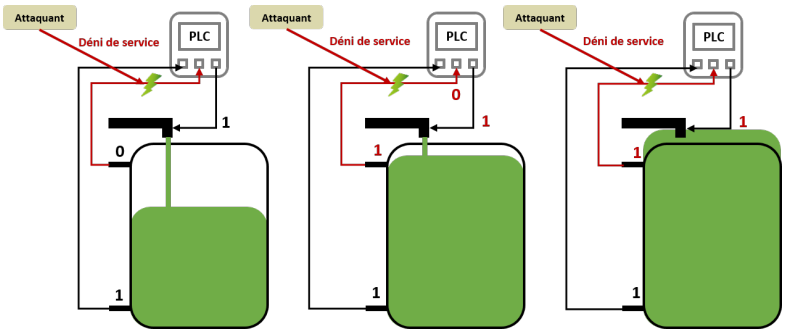


FIGURE 3.18 – Manipulation du programme PLC par blocage

   - Pour forcer une commande malveillante, l'attaquant peut envoyer un faux événement au PLC lorsque le système est dans un état limite afin d'engendrer

l'envoi d'une commande malveillante par le PLC ($\neg B$). Par exemple, un attaquant peut envoyer l'événement *Cuve vide* au PLC lorsque la cuve est pleine, ce qui engendre la réouverture de la vanne de remplissage et fait ainsi déborder la cuve comme présenté dans la Figure 3.19.

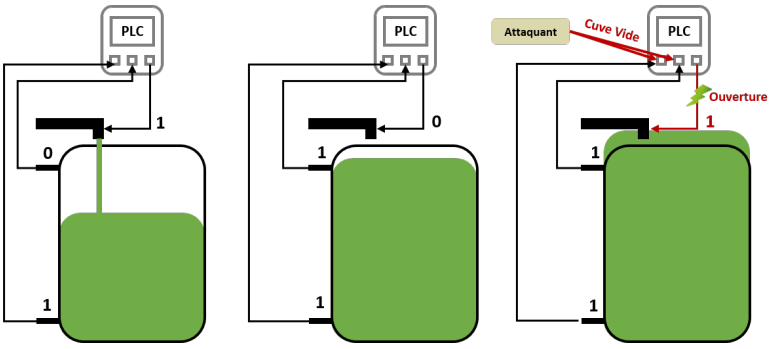


FIGURE 3.19 – Manipulation du programme PLC par forçage

Ces attaques, ***par forçage*** et ***par blocage***, nous permettent de construire un modèle d'attaque sûreté-sécurité, résumé par le Tableau 3.3, qui met en lien des manipulations d'origine informatiques et leur impact sur la sûreté du système.

Conséquence	**Menace**	**Moyen utilisé**
Compromission de la sûreté du système	Bloquer une commande	Manipulation du programme (entrées) Manipulation des commandes (sorties)
	Forcer une commande	Manipulation du programme (entrées) Manipulation des commandes (sorties)

TABLE 3.3 – Modèle d'attaque

3.2.4 Création du modèle de menace sûreté-sécurité

La réalisation d'une attaque sous-entend la violation d'une propriété de cybersécurité. Les propriétés de cybersécurité d'un système sont couramment présentées par le triptyque Confidentialité [2], Intégrité, Disponibilité et généralement étendu avec la propriété d'Authenticité. Pour créer notre modèle de menace sûreté-sécurité, nous avons identifié dans le Tableau 3.4 les propriétés de cybersécurité qu'un attaquant doit violer pour réaliser une attaque par blocage ou par forçage. Nous énumérons ces violations sous forme d'interactions entre les composants du système (sources, destination, canal de communication), comme dans la méthode STRIDE par interaction (cf. section 2.1), afin d'obtenir

2. Dans notre modèle, nous ne prenons pas en compte la confidentialité, car elle n'a pas d'impact direct pour la sûreté du système, mais reste une propriété de sécurité à garantir puisqu'elle peut aider à réaliser une attaque.

un résultat compatible avec les vulnérabilités identifiées dans *Microsoft Threat Modeling Tool* (cf. Chapitre 2).

Attaque	**Propriété violée**	**Cible**	**Impact**
Bloquer une commande	Disponibilité	Src	La source est indisponible, les variables ne sont pas transmises
		Dest	La destination est indisponible, les variables ne sont pas reçues
		Com	Le canal de communication est indisponible, les variables ne transitent plus
	Intégrité	Src	Les variables stockées sur la source sont falsifiées, les variables envoyées sont falsifiées
		Dest	N/A
		Com	Les variables envoyées sont falsifiées durant le transit, les variables reçues sont falsifiées
	Authenticité	Src	N/A
		Dest	Les variables sont envoyées à la mauvaise destination, les variables ne sont pas reçues
		Com	N/A
Forcer une commande	Disponibilité	Src	N/A
		Dest	N/A
		Com	N/A
	Intégrité	Src	Les variables stockées sur la source sont falsifiées, les variables envoyées sont falsifiées
		Dest	N/A
		Com	Les variables envoyées sont falsifiées durant le transit, les variables reçues sont falsifiées
	Authenticité	Src	Les variables sont envoyées par un attaquant, les variables envoyées sont forgées
		Dest	N/A
		Com	N/A

Src = Source, Dest = Destination, Com = Communication

TABLE 3.4 – Réalisation des menaces du modèle

À partir de ce tableau, nous pouvons faire correspondre les vulnérabilités identifiées dans *Microsoft Threat Modeling Tool* avec les attaques par blocage ou forçage qu'elles permettent de réaliser. En effet, les vulnérabilités dans *Microsoft Threat Modeling Tool* sont classées par famille de menace STRIDE. Chaque menace STRIDE correspond à la violation d'une propriété de cybersécurité. Ainsi, nous pouvons corréler une vulnérabilité dans *Microsoft Threat Modeling Tool* avec l'attaque par blocage ou forçage qu'elle réalise en fonction de la propriété de cybersécurité violée par sa catégorie STRIDE tel que présenté dans le Tableau 3.5.

Attaque	**Propriété violée**	**STRIDE**
Bloquer une commande	Disponibilité	Déni de service (*Denial of service*)
	Intégrité	Falsification (*Tampering*)
	Authenticité	Usurpation d'identité (*Spoofing*)
Forcer une commande	Disponibilité	Déni de service (*Denial of service*)
	Intégrité	Falsification (*Tampering*)
	Authenticité	Usurpation d'identité (*Spoofing*)

TABLE 3.5 – Corrélation entre STRIDE et le modèle de menace

La Figure 3.20 présente un exemple d'attaque par blocage qui viole la disponibilité de la source de l'interaction (cf. Tableau 3.2), à savoir un capteur de niveau de liquide d'une cuve. Cette attaque permet de compromettre le déclenchement de la fonction de sûreté *Cuve Pleine* ***implique*** *vanne de remplissage fermée*. Le schéma gauche de la Figure 3.20, présente le système dans un état stable de remplissage de la cuve (*Cuve pleine* = 0 et *vanne de remplissage* = 1). Le schéma à droite présente le passage du système d'un état stable à un état limite (↑ *Cuve pleine*). Si un attaquant a préalablement violé la

disponibilité du capteur fournissant la valeur de variable *Cuve pleine* au PLC, alors l'attaquant empêchera la lecture par le PLC de l'événement critique ↑ *Cuve pleine*, ce qui inhibe la commande de fermeture de la vanne de remplissage et engendre le débordement de la cuve.

Si *Microsoft Threat Modeling Tool* identifie une vulnérabilité capable de réaliser un déni de service sur le capteur, alors un attaquant pourra exploiter cette vulnérabilité pour réaliser l'attaque ***par blocage*** que nous venons de décrire.

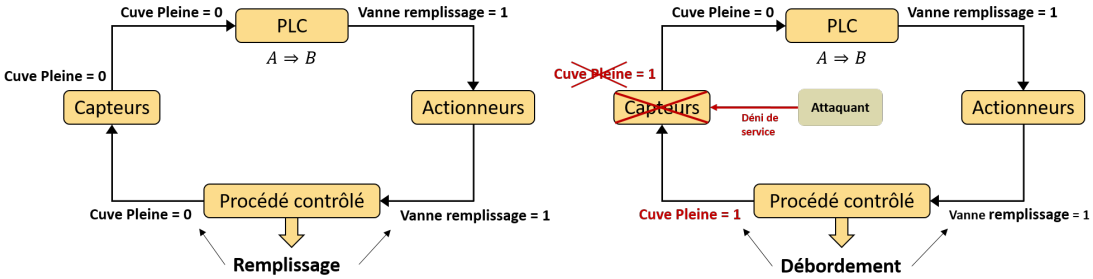


FIGURE 3.20 – Exemple d'attaque par déni de service de la source

Dans cette section, nous venons de présenter notre modèle de menace sûreté-sécurité qui permet de mettre en lien l'exploitation d'une vulnérabilité avec ses conséquences pour la sûreté du système. Dans la section suivante, nous présentons l'intégration de ce modèle de menace dans un SLA (modèle du système) afin de construire des scénarios d'attaque qui compromettent la sûreté d'un système industriel.

3.3 Identification des scénarios d'attaque qui compromettent la sûreté

Dans la section précédente, nous avons présenté notre modèle de menace sûreté-sécurité qui met en lien les vulnérabilités du système avec des attaques de compromission de la sûreté. Dans cette section, nous présentons l'intégration de ce modèle de menace dans le SLA afin d'identifier toutes les attaques par blocage ou forçage réalisables que nous appelons scénarios d'attaque.

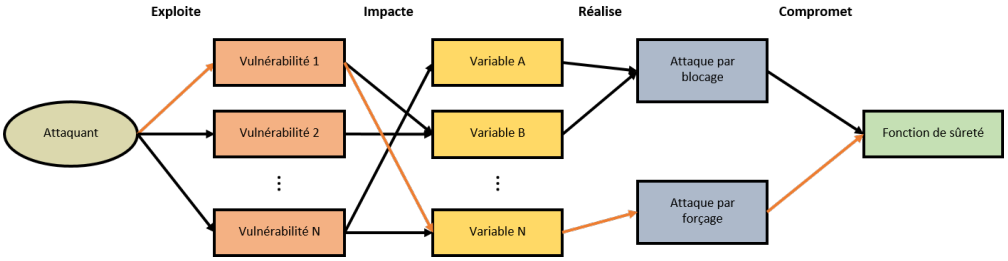


FIGURE 3.21 – Schéma d'un Scénario d'Attaque

La Figure 3.21 synthétise le processus d'un scénario d'attaque. Dans un premier temps, un attaquant exploite une vulnérabilité du système qui impacte une ou des variables du programme de contrôle. Ces impacts permettent de réaliser une attaque par blocage ou par forçage afin de compromettre la sûreté du système. Dans le chapitre précédent (cf. Chapitre 2), nous avons identifié les vulnérabilités du système et dans la section précédente (cf. section 3.2), nous avons présenté comment une vulnérabilité pouvait impacter une variable du programme de contrôle pour réaliser une attaque par blocage ou par forçage. Dans cette section, nous expliquons comment, à partir du SLA, nous identifions les scénarios d'attaque spécifiques au système à l'étude ainsi que les variables nécessaires à impacter pour réaliser ces scénarios.

Dans notre modèle de menace, nous avons défini deux attaques : les attaques par forçage et celles par blocage qui peuvent être toutes les deux réalisées soit par manipulation du programme du PLC (entrées), soit par manipulation des commandes du PLC (sorties). Chacune de ces deux attaques a pour objectif de compromettre l'exécution de la fonction de sûreté, respectivement le déclenchement de la fonction de sûreté pour l'attaque par blocage et le maintien de la fonction de sûreté pour l'attaque par forçage. L'exécution de la fonction de sûreté, que ce soit le *déclenchement* ou le *maintien*, a une temporalité entre un instant t et $t+1$ tel que présenté dans le Tableau 3.2. Dans un SLA, cette temporalité implique la combinaison d'une paire de transitions consécutives, une pour l'instant t et une seconde pour l'instant $t+1$. En théorie des graphes, une suite consécutive d'arcs (transitions) s'appelle un chemin dont la longueur est le nombre de transitions compris dans le chemin. Dans notre cas, les évolutions de la fonction de sûreté sont des chemins de longueur deux.

Identification des attaques par blocage - Les attaques par blocage ont pour objectif de bloquer le déclenchement de la fonction de sûreté en empêchant la commande d'être exécutée par l'actionneur. Les attaques par blocage consistent donc soit à :

- empêcher la lecture de l'évènement critique ($\uparrow A$) par le PLC ; soit à
- empêcher l'exécution de la commande de protection ($\uparrow B$) par l'actionneur.

Pour déterminer les scénarios d'attaque par blocage, nous devons donc d'abord identifier les déclenchements de la fonction de sûreté dans le SLA. Puis, nous définissons les variables qu'un attaquant doit manipuler pour : (1) empêcher la lecture de l'évènement critique ($\uparrow A$) par le PLC ou (2) empêcher l'exécution de la commande de protection ($\uparrow B$) par l'actionneur.

Le déclenchement d'une fonction de sûreté (**Si** $\uparrow A$ **Alors** $\uparrow B$) peut être modélisé par un chemin du SLA $(\neg A \wedge \neg B)_t \rightarrow (A \wedge B)_{t+1}$ que nous appelons chemin de déclenchement de la fonction de sûreté. Dans un premier temps, nous identifions donc l'ensemble des chemins de taille 2 dans le SLA représentant un déclenchement de la fonction de sûreté $(\neg A \wedge \neg B)_t \rightarrow (A \wedge B)_{t+1}$.

Ensuite, pour chaque chemin identifié, nous identifions les variables à manipuler (commandes d'actionneurs ou mesures de capteurs). Dans le cas où A et B représente respectivement une seule valeur de mesure d'un capteur et une seule valeur de commande d'un actionneur, l'identification de la variable à manipuler pour réaliser le scénario d'attaque

est triviale. Par exemple, si la fonction de sûreté correspond $lvl > 48 \implies \neg V1$ où lvl est une variable qui stocke la mesure du capteur de niveau de liquide de la cuve, 48 est le niveau de liquide maximum avant débordement de la cuve et $V1$ est une commande d'ouverture de la vanne $Vanne1$. Pour (1) empêcher la lecture de l'évènement critique, l'attaquant devra manipuler la variable lvl et pour (2) empêcher l'exécution de la commande de protection, l'attaquant devra manipuler la variable $V1$.

Dans le cas où A et B représentent plusieurs valeurs de variables, l'identification des variables à compromettre est plus complexe et dépend de la relation entre les variables (ET logique et OU logique) :

- **Conjonction de commandes d'actionneur (ET logique) :** Dans le cas d'une conjonction de commandes d'actionneur, la fonction de sûreté est compromise si au moins une commande d'actionneur de la commande de protection n'est pas exécuté. Nous pouvons donc traiter la fonction de sûreté comme un ensemble de sous-fonction comportant une seule commande d'actionneur à compromettre. Par exemple, si B correspond à $X \wedge Y$, la fonction de sûreté peut être traitée comme deux sous-fonctions à satisfaire $A \implies X$ et $A \implies Y$ comportant chacune une seule commande d'actionneur, comme présenté dans l'équation 3.9.

$$\begin{aligned}
&\text{Sûreté} : A \implies B \Leftrightarrow \neg A \vee B \\
&\text{Sûreté} : \neg A \vee (X \wedge Y) \qquad\qquad \#B = X \wedge Y \\
&\text{Sûreté} : (\neg A \vee X) \wedge (\neg A \vee Y) \qquad \#\text{Développement} \\
&\text{Sûreté} : (A \implies X) \wedge (A \implies Y) \\
&\text{Sous-fonction 1} : (A \implies X) \qquad \text{Sous-fonction 2} : (A \implies Y)
\end{aligned} \tag{3.9}$$

- **Disjonction de commandes d'actionneur (OU logique) :** Dans le cas d'une disjonction de commandes d'actionneur, la fonction de sûreté est compromise si aucune des commandes d'actionneur de la commande de protection n'est exécutée. Par exemple, si B correspond à $X \vee Y$, alors il existe trois chemins possibles pour appliquer la fonction de sûreté :
  1. $\uparrow X$: seule la commande d'actionneur X est envoyée $(\neg X \wedge \neg Y)_t \rightarrow (X \wedge \neg Y)_{t+1}$
  2. $\uparrow Y$: seule la commande d'actionneur Y est envoyée $(\neg X \wedge \neg Y)_t \rightarrow (\neg X \wedge Y)_{t+1}$
  3. $\uparrow X \wedge \uparrow Y$: les deux commandes d'actionneur X et Y sont envoyées $(\neg X \wedge \neg Y)_t \rightarrow (X \wedge Y)_{t+1}$

  Si une seule commande d'actionneur est envoyée, l'attaquant doit manipuler cette variable pour réaliser le scénario d'attaque (par exemple X dans le premier chemin). Sinon, l'attaquant doit manipuler les deux commandes d'actionneur pour empêcher la fonction de sûreté d'être déclenchée (par exemple, X et Y dans le chemin 3).
- **Conjonction de mesures de capteur (ET logique) :** Dans le cas d'une conjonction de mesures de capteur, la fonction de sûreté est déclenchée lorsque l'ensemble des mesures de capteur de l'état limite sont atteintes. Ainsi, nous devons identifier les mesures de capteur qui font passer le système d'un état stable à un état limite.

Par exemple, si A correspond à $V \wedge W$, alors l'événement critique $\uparrow A$ peut provenir de trois chemins différents dans le SLA :

1. $\uparrow V$ fait passer le système dans un état limite $(\neg V \wedge W)_t \rightarrow (V \wedge W)_{t+1}$
2. $\uparrow W$ fait passer le système dans un état limite $(V \wedge \neg W)_t \rightarrow (V \wedge W)_{t+1}$
3. $\uparrow V \wedge \uparrow W$ font passer le système dans un état limite $(\neg V \wedge \neg W)_t \rightarrow (V \wedge W)_{t+1}$.

Dans le cas où une seule mesure de capteur entraîne le passage du système dans un état limite, l'attaquant doit manipuler cette variable pour réaliser le scénario d'attaque (par exemple V dans le premier chemin). Sinon, l'attaquant doit manipuler une seule des mesures de capteur qui entraîne le système dans un état limite (par exemple, V ou W dans le chemin 3).

- **Disjonction de mesures de capteur (OU logique) :** Dans le cas d'une disjonction de mesures de capteur, chaque mesure de capteur définit un état limite différent du système. Nous séparons donc ces mesures de capteurs dans des fonctions de sûreté différentes. Par exemple, si A correspond à $V \vee W$, alors nous définissons deux fonctions de sûreté : $V \implies B$ et $W \implies B$.

Identification des attaques par forçage - Les attaques par forçage ont pour objectif d'empêcher le maintien de la fonction de sûreté en forçant l'exécution d'une commande par l'actionneur. De la même manière que pour les attaques par blocages, nous devons d'abord identifier les maintiens de la fonction de sûreté dans le SLA. Le maintien d'une fonction de sûreté (**Si** $A \rightarrow A$ **Alors** $B \rightarrow B$) peut être modélisé par un chemin du SLA tel que $(A \wedge B)_t \rightarrow (A \wedge B)_{t+1}$ que nous appelons chemin de maintien de la fonction de sûreté. Cependant, le maintien d'une fonction de sûreté n'est pas toujours clairement défini dans un programme PLC, car le concepteur connaît le procédé. Par exemple, dans la gestion d'une cuve, si le mélange de liquide nécessite d'être refroidi puis d'être mélangé, alors la cuve restera pleine dans deux états différents du système (*Refroidissement* et *Mélange*). Cependant, le concepteur du programme ne définira pas explicitement le maintien de la fonction de sûreté pendant le refroidissement et le mélange, car il n'y a aucune raison physique qui nécessiterait que la vanne de remplissage soit rouverte. Par conséquent, nous ne trouverons pas toujours un chemin de maintien de la fonction de sûreté. Néanmoins, nous pouvons appliquer l'attaque par forçage directement après une transition $(A \wedge B)_t$, puisque nous savons que le système est encore dans un état limite $(A = 1)$. Si un attaquant force une commande malveillante $(\neg B)$ à ce moment-là, alors le système est en danger. Le forçage d'une commande malveillante peut être réalisé soit par manipulation du programme du PLC, soit par manipulation des commandes.

- **Manipulation des commandes :** Au moment où le système est dans un état limite $(A \wedge B)_t$, un attaquant peut envoyer directement aux actionneurs une commande malveillante $(\neg B)$. Il est donc possible de compromettre le maintien de la fonction de sûreté en envoyant une commande malveillante après n'importe quelle transition franchie par le PLC qui comprend pour entrée l'état limite (A). Il existe autant de commandes de compromission que de combinaison booléenne possible par la négation de la commande de protection. Par exemple, si la commande de

protection B représente la fermeture des vannes de remplissage $\neg V1 \wedge \neg V2$, la commande malveillante est donc égale à $\neg(\neg V1 \wedge \neg V2)$ soit $V1 \vee V2$. La commande malveillante est vraie pour trois combinaisons de valeurs de $V1$ et $V2$, comme présenté dans le Tableau 3.6. Chacune de ces trois combinaisons sont des commandes malveillantes valides pour compromettre la sûreté et réalisent donc trois scénarios d'attaque différents.

$V1$	$V2$	$V1 \vee V2$
0	0	Faux
0	1	Vrai
1	0	Vrai
1	1	Vrai

TABLE 3.6 – Table de vérité $V1 \vee V2$

- **Manipulation du programme :** La manipulation des entrées du programme vise à forcer une transition du SLA (programme de l'automate) qui a pour sortie une commande malveillante ($\neg B$). Pour cela, nous recherchons dans le SLA un chemin tel que $(A \wedge B)_t \rightarrow (x \wedge \neg B)_{t+1}$ où x symbolise une combinaison d'entrées. Dans ce chemin, juste après l'exécution par le programme de $(A \wedge B)_t$, nous savons que le système est dans un état limite ($A = 1$). À ce moment-là, un attaquant peut falsifier les variables d'entrée pour que le PLC lise x et exécute la sortie liée à cette transition qui est une commande malveillante ($\neg B$). Comme expliquer précédemment, pour la conjonction des mesures de capteurs, x peut représenter plusieurs mesures de capteurs. Nous devons donc, de la même manière, identifier les mesures de capteurs qui permettent satisfaire $\uparrow x$ à l'instant $t + 1$ selon les différents chemins de *maintien* de la fonction de sûreté identifiés. Par exemple, La Figure 3.22 présente le programme de gestion d'une cuve comprenant deux étapes : (1) Remplissage ($Vanne\_remplissage = 1$ et $Vanne\_vidange = 0$) et (2) Vidange ($Vanne\_remplissage = 0$ et $Vanne_vidange = 1$). Si au moment où le programme rentre dans l'étape *Vidange* ($Plein = 1$ et $Vanne_vidange = 1$), un attaquant envoie l'information $Vide$ au PLC, le PLC transitera dans l'étape *Remplissage* et enverra une commande d'ouverture de la vanne de remplissage alors que la cuve est pleine faisant ainsi déborder la cuve.

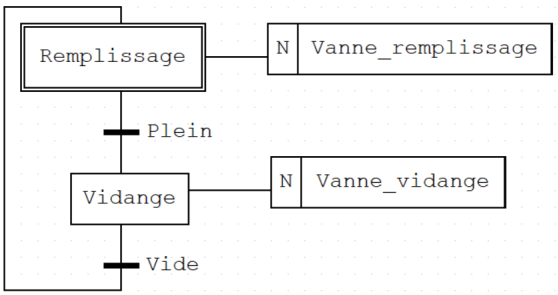


FIGURE 3.22 – Exemple gestion d'une cuve à deux états

Identification des scénarios d'attaque réalisables

Nous venons de présenter pour chacune des attaques de notre modèle de menace comment identifier les scénarios d'attaque dans le SLA (chemin du SLA ainsi que les variables à compromettre). Pour rappel, dans la Figure 3.21, nous avons défini nos scénarios d'attaque comme un processus en 4 étapes :

1. Un attaquant exploite une ou plusieurs vulnérabilités du système. Ces vulnérabilités sont identifiées avec l'outil *Microsoft Threat Modeling Tool* (cf. Chapitre 2).
2. Ces vulnérabilités impactent une ou plusieurs variables des programmes de contrôle des PLC. Ces impacts que nous appelons des manipulations sont listés dans le Tableau 3.4 (cf. section 3.2).
3. Ces manipulations de variables permettent la réalisation d'une attaque par blocage ou par forçage (cf. section 3.2.3).
4. Enfin, ces attaques entraînent une compromission d'une fonction de sûreté.

Ainsi pour finaliser les scénarios d'attaque, nous devons : (1) identifier les impacts des vulnérabilités du système sur les variables des programmes de contrôle ; (2) déterminer si les manipulations des variables (permises par les vulnérabilités du système) permettent de réaliser les scénarios d'attaques que nous avons précédemment identifiés dans cette section.

Dans le Chapitre 2, nous avons généré un fichier qui énumère les vulnérabilités liées à chaque variable des programmes de contrôle. À partir de ce fichier et du Tableau 3.4, nous associons à chaque variable les manipulations qu'un attaquant peut réaliser. Ainsi, nous pouvons déterminer les scénarios d'attaque réalisables en corrélant les manipulations réalisables par un attaquant (vulnérabilités) et les manipulations nécessaires pour réaliser le scénario d'attaque. Par exemple, dans le programme présenté dans la Figure 3.22, un attaquant peut faire déborder la cuve en empêchant la lecture du capteur *Plein* (La cuve devient pleine) afin d'empêcher la fermeture de la vanne de remplissage (attaque par blocage). Pour réaliser ce scénario d'attaque, le système doit contenir une vulnérabilité qui permet réaliser une des menaces identifiées dans le Tableau 3.4, c'est-à-dire de :

- Violer la disponibilité
  - d'un équipement (source ou destination) qui manipule cette variable
  - d'un canal de communication qui manipule cette variable
- Violer l'intégrité
  - d'un équipement (source) qui manipule cette variable
  - d'un canal de communication qui manipule cette variable
- Violer l'authenticité d'un équipement (destination) qui manipule cette variable

Dans le cas où la variable *Plein* transite au travers d'un canal de communication non sécurisé qui ne fournit pas d'intégrité, un attaquant pourra réaliser le scénario d'attaque suivant :

Un attaquant falsifie la variable Plein en exploitant le manque d'intégrité de la communication entre la base de données et le programme PLC afin d'empêcher la lecture de

l'événement critique ↑ Plein (cuve devient pleine) et ainsi inhibe la fermeture de la vanne de remplissage entraînant un débordement de la cuve.

À ce moment-là, nous vérifions si la variable *Plein* est une variable booléenne ou une représentation binaire d'une formule logique du premier ordre (cf. section 3.1.4). Si la variable n'est pas booléenne, alors nous utilisons un solveur SMT pour déterminer la plage de valeurs de *Plein*.

3.4 Comparaison avec l'état de l'art

Dans cette section, nous comparons notre travail à l'état de l'art des méthodes intégrées d'appréciation du risque sûreté-sécurité. Dans ce travail, nous avons mis au point une nouvelle méthode intégrée d'appréciation du risque sûreté-sécurité pour les ICS. Plus précisément, nous avons étudié la dépendance conditionnelle [54] entre la sûreté et la sécurité, où la sûreté du système dépend de son niveau de cybersécurité. Dans cette section, nous comparons notre travail avec les travaux connexes aux méthodes intégrées d'appréciation du risque de cybersécurité pour la sûreté des systèmes de contrôle industriel (Industrial Control System - ICS).

Winther et al. (2001) ont adapté la méthode HAZOP à des fins de sécurité selon un schéma adapter à la syntaxe anglaise suivante : **Pré-mot guide Attribut** *of* **composant** *due to* **Post-Mot guide** tel que Deliberate manipulation of the firewall due to insider (en français : Manipulation délibérée du pare-feu par une personne en interne). Ensuite, à partir de chaque expression générée par le schéma, ils identifient la menace, les causes et les conséquences. Enfin, la méthode a été appliquée à un cas d'utilisation du système téléphonique du chef de train (supervision du trafic ferroviaire).

Cárdenas et al. (2011) proposent une méthode d'appréciation du risque basée sur une simulation du procédé. Ils simulent un procédé chimique Tennesse-Eastman simplifiée où le modèle du procédé est mis en œuvre en FORTRAN et la loi de contrôle des régulateurs PID est décrite en MATLAB. Les auteurs ont ensuite simulé des attaques d'intégrité et de déni de service sur le réseau de capteurs du système afin d'identifier l'attaque la plus efficace pour surpresser un réservoir (état dangereux pouvant conduire à une explosion). Le système simulé comprend 3 capteurs, 3 actionneurs et 4 régulateurs PID.

Song et al. [64] (2012) présentent une méthode générique d'appréciation du risque basée sur les normes du domaine des centrales nucléaires. La méthode définit un flux de travail en six étapes. Tout d'abord, ils identifient le système et modélisent la cybersécurité. Ensuite, les actifs du système sont classés en niveaux de sécurité en fonction de leur criticité pour la sûreté de la centrale nucléaire. Chaque niveau de sûreté nécessite des contraintes de cybersécurité spécifiques. Troisièmement, une analyse des menaces est réalisée pour déterminer les scénarios d'attaque potentiels contre l'architecture du réseau de la centrale, suivie d'une analyse des vulnérabilités (collecte des CVE pertinents). Ensuite, des contrôles de sécurité sont conçus pour éliminer ou atténuer les vulnérabilités

relevées dans l'analyse des vulnérabilités. Enfin, la conception de la cybersécurité est validée lors d'une phase de pentesting (test de pénétration de la cybersécurité).

Young et Leveson [71] (2013) appliquent l'analyse du processus théorique du système (STPA) à des fins de sécurité nommée STPA-sec. La première étape consiste à identifier les principaux événements, c'est-à-dire les vulnérabilités et leur événement de perte correspondant, comme le rejet de matières radioactives (vulnérabilité) qui peut entraîner de graves blessures humaines ou la perte de vies humaines et la contamination de l'environnement (événement de perte). Ensuite, une spécification graphique des contrôles fonctionnels du système, appelée structure de contrôle de haut niveau (High Level Control Structure - HLCS), est élaborée afin de mettre en évidence l'absence de contraintes (sécurité). La troisième étape vise à identifier les actions de contrôle non sûres/non sécurisées qui peuvent entraîner les vulnérabilités du système identifiées lors de la première étape (libération de matières radioactives, par exemple). Les actions de contrôle non sûres/non sécurisées sont identifiées en fonction de quatre comportements.

- L'absence d'action de contrôle entraîne un risque.
- Fournir l'action de contrôle conduit à un danger.
- Fournir une action de contrôle potentiellement sûre, mais trop tôt, trop tard ou dans le mauvais ordre.
- L'action de contrôle dure trop longtemps ou est arrêtée trop tôt.

Ensuite, des exigences et des contraintes de sécurité de haut niveau sont définies à partir des actions de contrôle non sûres, telles que *La commande de fermeture de la vanne principale d'isolation de la vapeur ne doit jamais être fournie lorsqu'il n'y a pas de rupture ou de fuite.* Enfin, des scénarios de causalité sont élaborés pour violer les exigences et les contraintes de sécurité et de sûreté.

Rocchetto et Tippenhauer [60] (2017) présentent une méthodologie qui étend le modèle bien connu de l'attaquant Dolev-Yao pour prendre en compte les interactions physiques avec le procédé. Ensuite, ils modélisent l'état global du système par des propriétés LTL écrites en langage ASLAN++ vérifiées par l'outil CL-Atse [67] (2006). Enfin, ils évaluent leur méthodologie sur une station d'épuration appelée SWaT, contenant 18 capteurs et 27 actionneurs dont la modélisation a nécessité environs mille de lignes de code.

Mesli-kesraoui et al. [49] (2016) utilisent des automates temporisés pour modéliser la chaîne de contrôle-commande, c'est-à-dire les programmes de contrôle, les interfaces de supervision, les équipements physiques et les tâches humaines. Ils vérifient ensuite avec le vérificateur de modèle (model checker) UPPAAL des propriétés d'usage et de sûreté du modèle. Ils appliquent la méthodologie à un cas d'utilisation simple d'une tâche d'ouverture de vanne.

Zhu et al. [72] (2018) introduisent un modèle dynamique d'appréciation du risque, comprenant une modélisation du système par un modèle de flux multi-niveaux (multilevel flow model - MFM) qui décrit la structure et l'objectif du système industriel de manière graphique. Un réseau bayésien multicouche est construit à partir de ce modèle pour

modéliser la propagation des attaques, tandis qu'un calcul de perte de processus évalue les conséquences en termes d'argent. Enfin, le risque est calculé à partir du produit de la probabilité et de la perte. La méthode est appliquée à un procédé chimique comprenant trois sous-procédés : le procédé de réaction (4 vannes et 1 capteur de pression), le procédé de séparation (2 vannes et 1 capteur de niveau de liquide) et le procédé d'extraction (2 vannes et 1 capteur de niveau de liquide).

Puys et al. [59] (2018) présentent une méthodologie qui utilise des automates temporisés pour modéliser à la fois le profil de l'attaquant et le système. Le vérificateur de modèle UPPAAL vérifie ensuite les propriétés de sûreté sur la composition des automates temporisés du profil de l'attaquant et du système. Khaled et al. [42] (2020) proposent un moyen d'améliorer cette méthode en modélisant chaque élément du système, c'est-à-dire les entités physiques, les logiciels et les personnes qui sont définis par leurs capacités et leurs comportements. Un analyseur vérifie le modèle de graphe de l'interaction et de la communication entre les composants pour générer des attaques basées sur des propriétés de sécurité. La méthode crée un automate avec 100 000 états lorsqu'elle est appliquée à un processus physique, comprenant 3 actionneurs et 5 capteurs. Cheh et al. [14] (2018) suggèrent d'améliorer [59] en incorporant l'interaction avec la couche physique dans le modèle de l'attaquant.

Eckhart et al. [21] (2022) présentent une identification automatisée des risques de cybersécurité à l'aide du format d'échange AutomationML (AML). Les auteurs utilisent l'AML pour rassembler des artefacts d'ingénierie qu'ils complètent avec une bibliothèque de cybersécurité (AMLsec). Ils transforment ensuite l'AML en OWL (Web Ontology Language) afin de créer une base de connaissances comprenant les artefacts techniques (transformés en OWL à partir de l'AML), une ontologie de sécurité des ICS et des lien vers données de sécurité (CWE, CVE, etc.). À partir de cette base de connaissances, ils automatise (1) l'identification des menaces et des vulnérabilités, (2) les conséquences des attaques et (3) la génération de graphes d'attaques cyber-physiques. Enfin, ils appliquent leur méthode à l'exemple AML officiel d'une cellule robotisée d'un procédé de soudage par points.

Bhosale et al. [8] (2023) introduisent un modèle graphique qui montre le risque de cybersécurité pour la sûreté humaine et fonctionnelle. Ils utilisent un réseau de croyance bayésien (Bayesian Belief Network - BBN), c'est-à-dire un graphe acyclique direct avec des probabilités conditionnelles, pour calculer la probabilité de propagation du risque. Le BBN est composé de nœuds qui modélisent les fonctions du système et d'arêtes qui modélisent l'interdépendance entre la sûreté fonctionnelle, la sûreté humaine et les problèmes de sécurité dans le système. Le modèle leur permet de calculer la propagation du risque de cybersécurité en fonction des propriétés conditionnelles définies pour chaque relation causale (arête dirigée). Les auteurs appliquent leur méthode à la station Distributing Pro 5 qui assure les fonctions de maintien, de tri et d'alimentation des pièces (3 actionneurs et 4 capteurs). La modélisation nécessite un BBN, comprenant 74 nœuds et 77 arêtes.

Son et al. [63] (2023) présentent une appréciation globale du risque de cybersécurité

pour la sûreté des ICS nucléaires. La méthode utilise la célèbre analyse théorique des processus du système (STPA) pour identifier les scénarios de menace qui pourraient violer les contraintes de sûreté. Tout d'abord, les auteurs définissent les accidents et les risques pour le système (pertes du système, dangers du système et contraintes de sûreté). Ils modélisent ensuite un réseau de menaces pour chaque bien numérique critique (Critical Digital Asset - CDA) en fonction d'une liste d'événements de menace (comme STRIDE) et de vecteurs d'attaque (tirés de la norme NEI 10-09 du Nuclear Energy Institute). Troisièmement, ils modélisent les actions de contrôle dangereuses en définissant différents comportements de contrôle et leurs conséquences (cf. Young et Leveson [71], STPA-sec). Ensuite, les auteurs analysent les événements du réseau de menaces qui peuvent conduire à une action de contrôle dangereuses à partir de quatre cybermenaces : dans le contrôleur, dans les entrées contrôlées, dans le chemin de contrôle et dans le procédé contrôlé. Enfin, ils classent les risques de cybersécurité.

Le tableau 3.7 compare notre travail à l'état de l'art en termes de globalité de la méthode et de scalabilité à des systèmes larges.

Nous avons choisi d'évaluer la globalité des méthodes conformément au processus d'appréciation du risque de la norme ISO 31000. La norme ISO 31000 définit le processus d'appréciation du risque comme un processus en trois étapes permettant d'identifier, d'analyser et d'évaluer le risque. Selon cette définition, nous n'avons trouvé que trois appréciation globales et intégrées des risques sûreté-sécurité dans l'état de l'art [11, 72, 63].

Nous évaluons la scalabilité des méthodes en fonction du rapport entre la taille du système et de sa modélisation. Ce rapport permet de mettre en exergue la granularité de la modélisation. Plus le rapport est grand, plus le modèle comporte de détail et, par conséquent, devient difficile à appliquer sur des systèmes larges.

Le tableau 3.7 peut être divisé en deux groupes en fonction de la taille du système. Le premier groupe ([70, 64, 71, 49, 72, 59, 42, 8, 72]) applique sa méthodologie à des cas d'utilisation simple (moins d'une dizaine d'actionneurs et de capteurs) pour démontrer la granularité fine de leur modélisation sûreté-sécurité. Cependant, lorsque la taille de la modélisation est fournie, nous observons que ces méthodes ne s'adaptent pas aux systèmes complexes. D'autre part, le deuxième groupe, comprenant uniquement le travail de Rocchetto et Tippenhauer [60], applique sa méthode à des systèmes complexes (plus d'une dizaine de capteurs et d'actionneurs), ce qui montre la scalabilité de leur modélisation. Cependant, cette méthode nécessite beaucoup de temps et de ressources pour créer le modèle. Notre méthode offre un équilibre entre la granularité de la modélisation pour s'adapter aux systèmes complexes et le coût de la modélisation en termes de temps et de ressources nécessaires pour réaliser l'appréciation du risque. La principale limite de notre modélisation tient au faite que nous incorporons pas de connaissances du procédé en dehors des fonctions de sûreté. Il devient donc difficile avec notre méthode d'analyser les effets cascades et les attaques multi-étapes.

Méthode	**Taille du système**	**Taille de la modélisation**	**Globalité**
Winther et al. [70]	Système téléphonique pour les chefs de train (système comprenant 7 composants)	Non fourni	Id
Cárdenas et al. [11]	3 capteurs et 3 actionneurs	Non fourni (codes Fortran et MATLAB)	Id, An, Ev
Song et al. [64]	Système de protection d'un réacteur basé sur PLC	Pas de modèle du système	Id
Young et Leveson [71]	Vanne principale d'isolation de la vapeur d'un réacteur à eau pressurisée	5 sommets et 6 arêtes	Id
Rocchetto et Tippenhauer [60]	18 capteurs et 27 actionneurs	1000 lignes de code (ASLAN++)	Id
Mesli-kesraoui et al. [49]	Vanne	107 états (automate temporisé)	Pas d'appréciation du risque
Zhu et al. [72]	8 actionneurs et 3 capteurs	30 sommets et 32 arêtes (Modèle de flux multiniveaux étendu)	Id, An, Ev
Puys et al. [59]	2 capteurs et 2 actionneurs	Non fourni (automate temporisé)	Id
Cheh et al. [14]	Jonction en diamant de lignes ferroviaires	10, 11, 3, 2 états (4 automates temporisés)	Id
Khaled et al. [42]	5 capteurs et 3 actionneurs	99,220 états (Processus de décision Markovien)	Id
Eckhart et al. [21]	Cellule robotisée d'un processus de soudage par points	Non fourni	Id
Bhosale et al. [8]	3 actionneurs et 4 capteurs	74 nœuds et 77 arêtes (BBN)	Id
Son et al. [63]	Système de protection d'un réacteur basé sur PLC	Pas de modèle du système	Id, An, Ev
Ce travail	26 capteurs et 23 actionneurs	52, 30, 36 et 12 états (4 automates finis)	Id, An, Ev

Id = Identification, An = Analyse, Ev = Évaluation

TABLE 3.7 – Synthèse de la comparaison

3.5 Conclusion du chapitre

Dans ce chapitre, nous avons présenté en détail notre méthode d'identification des risques de cybersécurité pour la sûreté du système. Notre méthode commence par une modélisation du système (cf. section 3.1). Ce modèle du système est construit à partir des programmes des PLC que nous nous traduisons en SLA (automate à état finis) pour expliciter le comportement du système de contrôle. Le programme des PLC fournissent une modélisation minimale du système centrée sur une représentation des états du procédé nécessaires au contrôle du système (incluant sa sûreté). Cependant, nous avons montré que malgré cette représentation minimal, l'espace d'état du modèle diverge. Nous avons présenté que la principale limite de notre modélisation était liée à l'évolution exponentielle de l'espace d'état pour les programmes exécutés en parallèles. A partir de ce constat, nous avons introduit un ensemble d'optimisation qui limitent les effets de la parallélisation pour modéliser de systèmes larges. Pour chacune de ces optimisations, nous avons présenté la limite de complexité qu'elles réduisaient et les raisons de cette amélioration.

Dans un second temps, nous avons proposé un modèle de menace sûreté-sécurité qui corrèle la manipulation de variables de contrôle du système avec leurs conséquences sur la sûreté du système (cf. section 3.2). Ce modèle de menace exploite le fait que les variables de contrôles, que ce soit les variables venant du procédé ou les commandes émises par les PLC, transitent par une infrastructure de communication informatisée vulnérables aux cyberattaques. Nous avons donc une interface mixte qui est à la fois critique pour la sûreté du système et exposée aux cyberattaque. Notre modèle de menace déterminer les variables qu'un attaquant doit manipuler afin de compromettre le déclenchement ou le maintien d'une fonction de sûreté par l'intermédiaire d'attaques dites par blocage ou par forçage.

Dans un troisième temps, nous avons expliqué comment nous intégrons ce modèle de menace dans notre modélisation du procédé afin de générer des scénario d'attaque qui compromette la sûreté du système à l'étude (cf. section 3.3. Nous avons ensuite montrer comment nous filtrons les scénarios d'attaque en fonction des vulnérabilités du système pour ne conserver uniquement ceux réalisables pour le système modélisé. La recherche de ces scénarios d'attaque dans le modèle du procédé (SLA) consiste à identifier des chemins du SLA et évolue donc linéairement en fonction du nombre de transitions du SLA.

Enfin, nous avons présenté l'état de l'art des méthodes intégrés d'appréciations du risque de cybersécurité pour la sûreté du système avec lequel nous avons comparé notre travail (cf. section 3.4. Cette comparaison montre que les travaux de l'état de l'art soit proposent une description plus fine des cyberattaques mais ne sont pas capable de passer à l'échelle sur des systèmes larges, soit leur processus passe à l'échelle mais nécessitent un coût en terme de temps et de ressource non négligeable comparativement à une méthode automatisée comme la notre.

Dans nos travaux futurs, nous aimerions améliorer les scénarios d'attaque en analysant les attaques multi-étapes et les effets en cascade sur le système, tout en conservant la scalabilité de notre méthode actuelle.

4 Conclusion

Dans ce travail de thèse, nous avons présenté notre méthode d'appréciation du risque sûreté-sécurité. Cette méthode adresse une réponse à chacune des trois questions que nous nous étions posées dans l'introduction de ce manuscrit, à savoir :

- Comment **identifier** les menaces et vulnérabilités de cybersécurité spécifiques aux systèmes de contrôle industriel ?
- Comment **analyser** leurs conséquences sur la sûreté du système ?
- Comment **évaluer** le risque de cybersécurité pour la sûreté du système ?

En effet, notre méthode comprend 3 étapes qui, chacune, répond à une des questions posées. Dans la première étape, nous avons développé un processus automatisable d'identification des vulnérabilités spécifiques aux systèmes de contrôle industriel basé sur un outil de modélisation de la menace (cf. Chapitre 2). Dans la seconde étape, nous avons proposée une analyse des conséquences des vulnérabilités sur la sûreté du système selon une méthode d'identification du risque de cybersécurité impactant la sûreté du système à partir de la logique des PLC (cf. Chapitre 3). Enfin, notre troisième étape, évalue le risque de cybersécurité pour la sûreté du système selon une matrice des risques qui détermine si le niveau du risque est tolérable par l'organisation détentrice du système (cf. section 1.1.3).

Pour conclure ce manuscrit, nous proposons un rappel des contributions et perspectives globales de notre méthode et spécifiques de chacune des étapes

4.1 Contributions et perspectives globales

Du point de vue global, nous avons contribué à développer une méthode d'appréciation du risque sûreté-sécurité complète comprenant un haut niveau d'automatisation et qui est applicable à des systèmes complexes :

Niveau d'automatisation : Dans ce travail, nous avons évalué le niveau d'automatisation de notre processus selon trois critères qualitatifs :

1. *Manuel :* l'étape du processus nécessite une intervention externe
2. *Automatisable :* nous avons apporté les informations nécessaires pour qu'une personne initiée à la programmation informatique puisse créer un code d'automatisation.
3. *Automatisé :* un code d'automatisation a été développé dans le cadre de cette thèse.

Selon ces critères, notre méthode est majoritairement automatisable ou automatisée. Cependant, certaines étapes restent tout de même manuelles. Dans un futur travail, nous envisageons d'étudier comment nous pourrions rendre ces étapes automatisables.

Couverture : Nous qualifions notre méthode de complète, car elle couvre l'ensemble des étapes du processus d'appréciation du risque ISO 31000, à savoir l'identification, l'analyse et l'évaluation du risque.

Passage à l'échelle : Nous évaluons le passage à l'échelle des méthodes en fonction du rapport entre la taille du système (nombre de capteurs et d'actionneurs) et de sa modélisation. Ce critère permet d'apprécier la capacité d'un modèle à représenter des systèmes relativement complexes (plus d'une dizaine de capteurs et d'actionneurs). Dans le Chapitre 3, nous avons montré que notre processus de modélisation était capable de modéliser un système comprenant 26 capteurs et 23 actionneurs. L'Annexe A, montre que notre méthode est capable de modéliser des systèmes complexes composés de sous-procédés comprenant jusqu'à 12 960 états avec un ordinateur de bureautique standard. Dans ce manuscrit, nous avons présenté un premier processus de modélisation qui comprenait une complexité de calcul qui ne permettait pas de modéliser des systèmes complexes dans un temps raisonnable (cf. section 3.1.2). Pour pallier cela, nous avons présenté plusieurs optimisations permettant de réduire de manière importante le temps de modélisation rendant possible la modélisation de systèmes complexes. Cependant, notre processus actuel comprend tout de même une limite physique (espace mémoire) pour modéliser de plus grands systèmes. Dans un futur travail, nous souhaiterions continuer notre étude de complexité afin : (1) d'identifier l'évolution temporelle (temps de modélisation) et physique (espace mémoire) de notre processus en fonction de l'espace d'état du système à modéliser ; (2) d'identifier des heuristiques pour réduire l'espace d'état du modèle ; (3) d'identifier des algorithmes plus performants et des représentations du modèle moins volumineuses pour réduire l'empreinte mémoire de notre programme.

Actuellement, dans notre programme, le modèle est représenté par un tableau comprenant l'ensemble des transitions du graphe sous la forme *[état source, état destination, condition de transition, sortie]*. Nous pourrions améliorer cette représentation (3) en modélisant le système, au niveau du programme, par une matrice d'adjacences ou une liste d'adjacences. Cette représentation pourrait minimiser la taille du graphe en mémoire et permettre d'utiliser les fonctions matricielles pour réduire la complexité algorithmique du produit fort de graphes. Pour l'identification des heuristiques (2), nous pourrions construire, à la place d'un produit fort de graphes, un graphe hybride pour lequel nous fusionnerions uniquement les états et les transitions des graphes initiaux qui possèdent une variable en commun (entrée ou sortie). Ce graphe hybride permettrait ainsi de restreindre l'évolution exponentielle de l'espace produit d'état du modèle final.

4.2 Contributions et perspectives spécifiques

Notre méthode comprend 3 étapes qui relèvent des challenges spécifiques. Dans la suite, nous rappelons, pour chacune des étapes de notre méthode, nos contributions pour

relever ces défis ainsi que nos perspectives.

Étape 1 - La première étape de notre méthode contribue à identifier les menaces et vulnérabilités spécifiques aux ICS. Pour cela, nous avons :

1. Proposé une méthode de modélisation des composants ICS et d'intégration de leurs vulnérabilités dans *Microsoft Threat Modeling Tool*.
2. Développé un processus d'automatisation du flux de travail de *Microsoft Threat Modeling Tool* afin d'automatiser l'identification des vulnérabilités spécifiques aux systèmes de contrôle industriels.

Nous avons développé une méthode de modélisation des composants ICS dans *Microsoft Threat Modeling Tool* à partir des profils de protection de l'ANSSI. Ces profils de protection servent initialement à la préparation d'une certification de sécurité de premier niveau (CSPN) reconnue, à ce jour, uniquement par la France (ANSSI) et l'Allemagne (BSI). Dans un travail futur, nous voulons éclaircir l'état actuel et les perspectives du schéma de certification européen (ICCS) afin que notre modélisation des composants ICS soit extraite d'un schéma de certification de sécurité reconnue à l'échelle européenne.

Afin de pleinement automatiser le flux de travail de l'outil *Microsoft Threat Modeling Tool*, nous avons élaboré une méthode automatisable de construction d'un modèle du système (DFD) utilisable par *Microsoft Threat Modeling Tool*. L'intégration de ces deux méthodes (modélisation des composants ICS, création du modèle du système) au flux de travail standard de *Microsoft Threat Modeling Tool* permet de créer un processus automatisable d'identification des vulnérabilités spécifiques aux ICS.

Dans un travail ultérieur, nous souhaiterions proposer un filtrage automatique des vulnérabilités identifiées par l'outil *Microsoft Threat Modeling Tool* en fonction d'un modèle d'attaquant. Ce modèle d'attaquant serait construit à partir des critères de description des menaces définis par la norme CEI 62443-3-2, à savoir : la source de la menace, la capacité ou niveau de qualification de la menace, le vecteur de menace et les actifs potentiellement affectés par la menace. Ces informations nous permettraient de filtrer les vulnérabilités pour ne conserver que les vulnérabilités exploitables par l'attaquant modélisé. Une autre perspective de cette première étape serait d'étudier la possibilité de définir automatiquement le niveau de sécurité atteint par le système tel que défini dans la norme CEI 62443-3-3 afin d'établir les contrôles techniques de cybersécurité à mettre en œuvre pour parvenir au niveau de sécurité souhaité par l'organisation détentrice du système.

Étape 2 - La seconde étape de notre appréciation du risque contribue à identifier les conséquences des vulnérabilités sur la sûreté du système. Pour cela, nous avons développé un modèle de menace sûreté-sécurité qui met en relation les vulnérabilités du système avec leurs impacts sur la sûreté par l'intermédiaire de scénarios d'attaque. Ces scénarios d'attaque identifient, à partir d'un modèle du comportement des PLC, les vulnérabilités à exploiter par un attaquant ainsi que les valeurs de variables à fixer pour compromettre

la sûreté du système. À ce jour, nos scénarios d'attaque comprennent uniquement des attaques directes. Dans nos travaux futurs, nous souhaiterions étudier les attaques multi-étapes et intégrer dans le modèle du comportement des PLC les effets en cascade d'une cyberattaque, tout en conservant la possibilité d'analyser des systèmes complexes.

Étape 3 - Enfin, dans la dernière étape de notre méthode, nous construisons une matrice des risques qui classe les scénarios d'attaque en fonction de leur vraisemblance et de leur impact. La vraisemblance est déterminée selon le score d'exploitabilité des vulnérabilités qui composent le scénario et l'impact en fonction du danger engendré par la fonction de sûreté compromise par le scénario d'attaque. De plus, nous déterminons les scénarios d'attaque qui nécessitent des remédiations complémentaires conformément au risque tolérable par l'organisation détentrice du système. Dans un travail à posteriori de cette thèse, nous souhaiterions étudier la possibilité de définir des métriques quantitatives d'évaluation de la vraisemblance d'un scénario d'attaque basées sur des données fiables et quantifiables.

Pour conclure ce manuscrit, je souhaitais replacer les contributions de cette thèse dans un cadre plus global de la convergence de la sûreté et de la sécurité des systèmes de contrôle industriel plus précisément par rapport au fonctions définies par le cadre de cybersécurité du NIST et la classification des dépendances entre sûreté et sécurité. Tout d'abord, selon le cadre de cybersécurité (cybersecurity framework) du NIST [51], la résilience d'un système est conditionnée par 6 fonctions :

1. **Gouverner :** La stratégie de gestion des risques de cybersécurité de l'organisation, les attentes et la politique de l'organisation sont établies, communiquées et contrôlées.
2. **Identifier :** les risques actuels de l'organisation en matière de cybersécurité sont compris. Cette fonction inclut parmi deux autres la réalisation d'une appréciation du risque.
3. **Protéger :** Des mesures de protections sont utilisées pour gérer les risques de cybersécurité de l'organisation.
4. **Détecter :** Les attaques et compromissions éventuelles de cybersécurité sont détectées et analysées.
5. **Répondre :** Les mesures relatives à un incident de cybersécurité détecté sont prises
6. **Restaurer :** Les actifs et les opérations touchés par un incident de cybersécurité sont rétablis.

Par rapport à ce cadre nous avons contribué à répondre à l'une des exigences de la fonction *Identifier*.

Deuxièmement, afin de pleinement faire converger la sûreté et la sécurité, les systèmes industriels devraient intégrer pour chacune de ces fonctions l'ensemble des interdépendances entre la sûreté et la sécurité que nous avons mentionnés dans l'introduction de

manuscrit, à savoir la dépendance conditionnelle, le renforcement, l'antagoniste et l'indépendance. Dans cette thèse, nous avons étudié l'une des dépendances conditionnelles où la sûreté du système dépend de sa cybersécurité.

Cette mise en perspective de notre travail me permet de donner au lecteur l'intuition de l'étendu et de la richesse de ce champ de recherche pour lequel j'espère pouvoir contribuer dans les années à venir.

Annexes

Annexe A : Cas d'usage Tennessee-Eastman

Annexe A.1 : Présentation du cas d'usage Tennessee-Eastman

Le procédé chimique de Tennessee-Eastman est un système qui simule de nombreuses caractéristiques typiques des procédés industriels complexes de la vie courante. De ce fait, il est largement utilisé comme exemple de simulation dans le contrôle, l'optimisation, la surveillance et les défauts des systèmes industriels. La spécification de ce processus a été publiée pour la première fois en 1993 [19]. Dans notre cas, nous utilisons une version modifiée du processus d'origine.

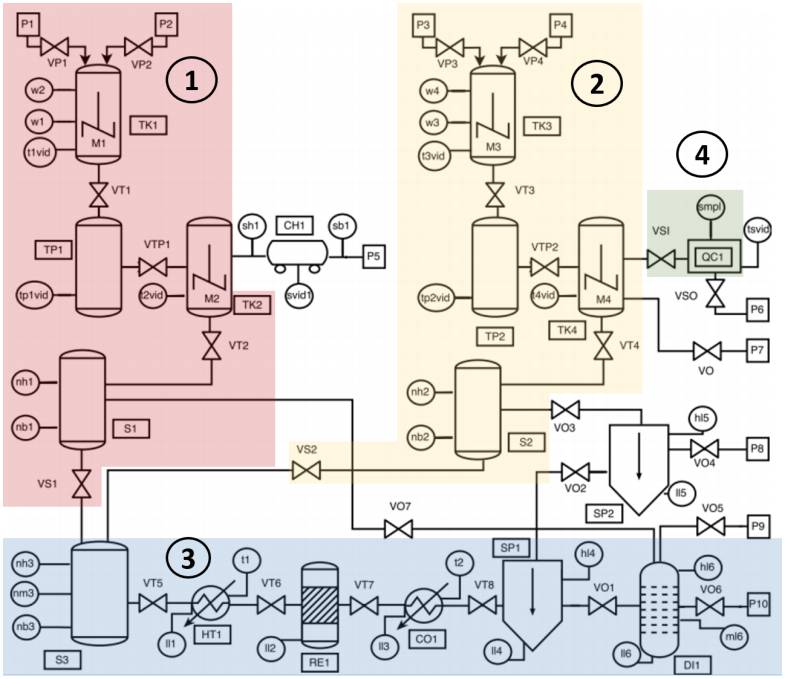


FIGURE 4.1 – Procédé chimique Tennessee-Eastman

L'objectif du procédé chimique, présenté dans la Figure 4.1, est de synthétiser deux produits par l'intermédiaire d'une réaction chimique impliquant plusieurs réactifs. Dans la Figure 4.1, nous avons mis en évidence les quatre parties que nous utilisons dans notre cas d'usage. Sachant que les parties 1 et 2 ont un fonctionnement similaire, nous présenterons uniquement la partie 1.

Partie 1 et 2 - Les réactifs initiaux, P1 et P2, sont injectés dans le réacteur TK1 (situé en haut à gauche de la Figure 4.1) respectivement par les vannes VP1 et VP2. Le produit P1 est injecté jusqu'à que le niveau de réactif atteint le capteur W1. Puis le produit P2 est injectée jusqu'à que le niveau du mélange atteigne le capteur W2. Dans le réacteur TK1, l'agitation des réactifs est contrôlé par le moteur M1. Le mélange est ensuite vidé dans la trémie tampon TP1 par la vanne VT1 puis évacué dans le réacteur TK2. À ce moment-là, le moteur du réacteur TK2 est mis en marche, de sorte que tous les réactifs du réservoir soit entièrement mélangés [1]. Si la trémie tampon TP1 n'est pas vide, le mélange est vidé dans le silo S1 avant de démarrer une nouvelle phase. Toutes les étapes précédentes sont synchronisées de sorte que le silo S1 ne soit jamais vide (donnée récupérée grâce au capteur NB1). Dans la partie 2 du procédé, les réactifs P3 et P4 seront mélangés et évacués jusqu'au silo S2 de façon similaire à ce qui est fait pour les produits P1 et P2. La principale différence réside dans le prélèvement d'échantillons de réactifs dans le réacteur TK4 réalisé par la vanne VSI (partie 4).

Partie 3 - Lorsque les réactifs de S1 et S2 ont fini de réagir, une quantité importante de produits entre dans S3 par l'intermédiaire des vannes VS1 (S1, réaction de P1 et P2) et VS2 (S2, réaction de P3 et P4). La quantité de réactifs dans S3 est surveillée par les capteurs de niveau de liquide NH3 et NM3. Après cela, les réactifs sont chauffés par le four HT1 pour répondre aux conditions de température requises pour les réactions ultérieures, puis transférés dans le réacteur RE1. Les produits de réaction dans RE1 seront refroidis dans CO1, puis séparés à l'aide des séparateurs SP1 et SP2 et de la colonne de distillation DI1 en fonction de la densité, du point de fusion et du point d'ébullition de la substance. Le produit final P10 est récupéré de la colonne de distillation par la vanne VO6.

1. Dans le processus original, P5 est transporté vers TK2 via le chariot CH1, ce qui explique pourquoi P1 et P2 sont de nouveau mélangés

Annexe A.2 : Temps, espaces d'états et limites de la modélisation du cas d'usage Tennessee-Eastman

Les résultats introduits dans le Tableau 4.2 s'appuient sur le procédé Tennessee-Eastman présenté dans la Figure 4.2 selon la nomenclature défini dans le Tableau 4.1. Les modèles présentés dans le Tableau 4.2 ont été générées avec un ordinateur portable équipé d'un processeur Intel(R) Core(TM)i5-8365U @1,60GHz-1,90GHz et de 16 Go de RAM.

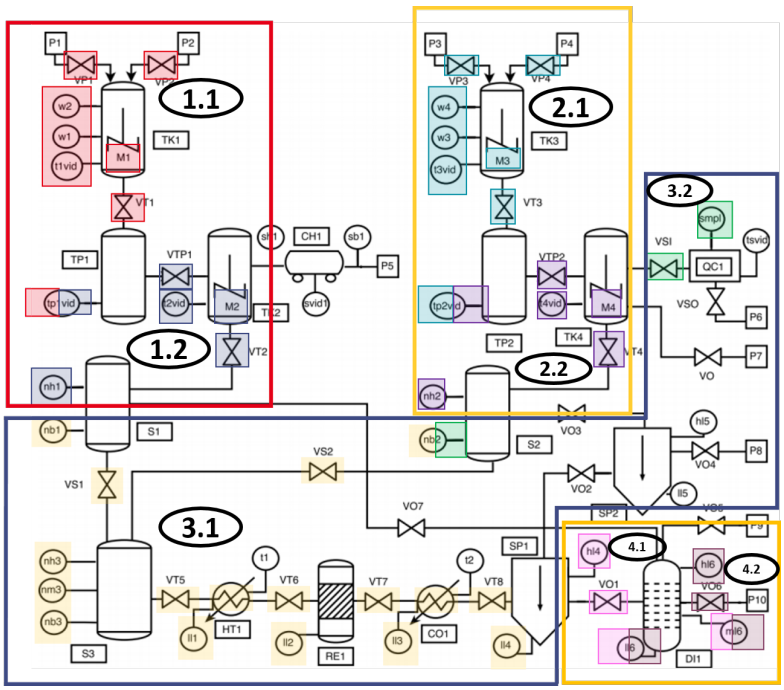


FIGURE 4.2 – SFCs du procédé chimique Tennessee-Eastman

Sous-procédé	**Représentation**	**SFC**	**Représentation**
Sous-procédé 1	encadrer Rouge	SFC 1.1 SFC 1.2	Rouge Gris
Sous-procédé 2	encadrer Jaune	SFC 2.1 SFC 2.2	Bleu Violet
Sous-procédé 3	encadrer Bleu	SFC 3.1 SFC 3.2	Jaune Vert
Sous-procédé 4	encadrer Orange	SFC 4.1 SFC 4.2	Rose Marron

TABLE 4.1 – Nomenclature de la Figure 4.2

Le Tableau 4.2 présente les résultats du temps de modélisation et de l'espace d'état modélisé pour le cas d'usage Tennessee-Eastman. Les quatre premiers blocs présentent le temps de modélisation et l'espace d'états pour chacun des sous-procédés. De plus, pour chaque sous-procédé, nous avons précisé le temps de modélisation et l'espace d'état des réseaux SFC qui le composent. Enfin, le dernier bloc présente les limites de modélisation que nous avons atteintes avec notre machine. Les résultats montrent que la limitation principale de notre processus de modélisation est liée à l'espace mémoire de la machine, car le temps de modélisation avant le crash de la mémoire, est tout à fait raisonnable.

Modèle	**Temps *PF SLA**	**Temps SLA**	**États**	**Transitions**
SFC 1.1	N/A	0min 0s 0ms	6	29
SFC 1.2	N/A	0min 0s 0ms	5	17
Sous-procédé 1	0min 0s 156ms	0min 0s 230ms	30	488
SFC 2.1	N/A	0min 0s 3ms	6	29
SFC 2.2	N/A	0min 0s 0ms	6	23
Sous-procédé 2	0min 0s 142ms	0min 0s 289ms	36	580
SFC 3.1	N/A	0min 0s 42ms	13	146
SFC 3.2	N/A	0min 0s 0ms	4	16
Sous-procédé 3	0min 0s 319ms	0min 1s 264ms	52	2000
SFC 4.1	N/A	0min 0s 0ms	4	16
SFC 4.2	N/A	0min 0s 1ms	3	9
Sous-procédé 4	0min 0s 133ms	0min 0s 34ms	12	87
Sous-procédé 1 + 2	0min 2s 636ms	18min 11s 838ms	1050	306 309
Sous-procédé 1 + 2 + SFC 3.2	0min 42s 420ms	Crash mémoire	4320	6 128 640
Sous-procédé 1 + 2 + SFC 3.2 + 4.1	**37 min	Crash mémoire	12 960	?

*PF = Produit fort de graphes
**Nous avons réussi à exécuter le programme une seule fois en 37 min. Le reste du temps, nous avons un crash mémoire.

TABLE 4.2 – Résultats

Bibliographie

[1] Houssein Abdo, Mohamad Kaouk, J-M Flaus, and François Masse. A safety/security risk analysis approach of industrial control systems : A cyber bowtie–combining new version of attack tree with bowtie analysis. *Computers & security*, 72 :175–195, 2018.

[2] MS Abrams and J Weiss. Malicious control system cyber security attack case study - maroochy water services. `https://www.mitre.org/sites/default/files/pdf/08_1145.pdf`, 2008. [Online ; accessed 22-July-2024].

[3] Eman A AbuEmera, Hesham A ElZouka, and Amani A Saad. Security framework for identifying threats in smart manufacturing systems using stride approach. In *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pages 605–612. IEEE, 2022.

[4] Md Rashid Al Asif, Khondokar Fida Hasan, Md Zahidul Islam, and Rahamatullah Khondoker. Stride-based cyber security threat modeling for iot-enabled precision agriculture systems. In *2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pages 1–6. IEEE, 2021.

[5] Thiago Alves. OpenPLC_Editor. `https://github.com/thiagoralves/OpenPLC_Editor`, 2024. [Online ; accessed 10-April-2024].

[6] and European Union Agency for Cybersecurity. *Good practices for security of Internet of things in the context of smart manufacturing*. Publications Office, 2018.

[7] Étienne André, Didier Lime, Mathias Ramparison, and Mariëlle Stoelinga. Parametric analyses of attack-fault trees. *Fundamenta Informaticae*, 182(1) :69–94, 2021.

[8] Pushparaj Bhosale, Wolfgang Kastner, and Thilo Sauter. Integrated safety-security risk assessment for production systems : A use case using bayesian belief networks. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–6, 2023.

[9] Zoe Braiterman, Adam Shostack, Jonathan Marcil, Stephen de Vries, Irene Michlin, Kim Wuyts, Robert Hurlbut, Brook S.E Schoenfield, Fraser Scott, Matthew Coles, Chris Romeo, Alyssa Miller, Izar Tarandach, Avi Douglen, and Marc French. Threat modeling manifesto. `https://www.threatmodelingmanifesto.org/`, 2020. [Online ; accessed 05-March-2024].

[10] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. Request for comments, Internet Engineering Task Force (IETF), Wilmington, USA, July 2017.

[11] Alvaro A. Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems : risk assessment,

detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, page 355–366, New York, NY, USA, 2011. Association for Computing Machinery.

[12] Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388(1-29) :3, 2016.

[13] Timothy Casey. Threat agent library helps identify information security risks. *Intel White Paper*, 2, 2007.

[14] Carmen Cheh, Ahmed Fawaz, Mohammad A. Noureddine, Binbin Chen, William G. Temple, and William H. Sanders. Determining tolerable attack surfaces that preserves safety of cyber-physical systems. *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing*, 2018.

[15] Cybersecurity & Infrastructure Security Agnecy (CISA). Cybersecurity and physical security convergence. `https://www.cisa.gov/sites/default/files/publications/Cybersecurity%2520and%2520Physical%2520Security%2520Convergence_508_01.05.2021.pdf`, 2021. [Online ; accessed 22-July-2024].

[16] Clusif. Fiches incidents cyber si industriel. `https://clusif.fr/publications/fiches-incidents-cyber-si-industriels/`, 2022. [Online ; accessed 22-July-2024].

[17] Worl Wide Web Consortium. Extensible markup language (xml). `https://www.w3.org/XML/`, 2016. [Online ; accessed 29-February-2024].

[18] Siddhartha Shankar Das, Edoardo Serra, Mahantesh Halappanavar, Alex Pothen, and Ehab Al-Shaer. V2w-bert : A framework for effective hierarchical multiclass classification of software vulnerabilities. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–12. IEEE, 2021.

[19] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3) :245–255, 1993.

[20] Victoria Drake. Threat Modeling. `https://owasp.org/www-community/Threat_Modeling`, 2024. [Online ; accessed 05-March-2024].

[21] Matthias Eckhart, Andreas Ekelhart, and Edgar Weippl. Automated Security Risk Identification Using AutomationML-Based Engineering Data. *IEEE Transactions on Dependable and Secure Computing*, 19(3) :1655–1672, May 2022.

[22] Shannon Eggers and Katya Le Blanc. Survey of cyber risk analysis techniques for use in the nuclear industry. *Progress in Nuclear Energy*, 140 :103908, 2021.

[23] Marco Ehrlich, Andre Bröring, Christian Diedrich, Jürgen Jasperneite, Wolfgang Kastner, and Henning Trsek. Determining the target security level for Automated Security Risk Assessments. *2023 IEEE 21st International Conference on Industrial Informatics*, 2023.

[24] Marco Ehrlich, Andre Bröring, Henning Trsek, Jürgen Jasperneite, and Christian Diedrich. Evaluation concept for prototypical implementation towards Automated Security Risk Assessments. *IEEE 28th International Conference on Emerging Technologies and Factory Automation*, 2023.

[25] European Reference Network for Critical Infrastructure Protection (ERNCIP). IACS Components Cybersecurity Certification Scheme. `https://erncip-project.jrc.ec.europa.eu/networks/tgs/european-iacs`, 2014. [Online ; accessed 28-June-2024].

[26] Nicolas Falliere, Liam O Murchu, Eric Chien, et al. W32. stuxnet dossier. *White paper, symantec corp., security response*, 5(6) :29, 2011.

[27] First. Common vulnerability scoring system version 3.1 : Specification document revision 1. `https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf`, 2019. [Online ; accessed 14-December-2023].

[28] Lars Halvdan Flå, Ravishankar Borgaonkar, Inger Anne Tøndel, and Martin Gilje Jaatun. Tool-assisted threat modeling for smart grid cyber security. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–8. IEEE, 2021.

[29] Markus Fockel, Sven Merschjohann, Masud Fazal-Baqaie, Torsten Förder, Stefan Hausmann, and Boris Waldeck. Designing and integrating IEC 62443 compliant threat analysis. *Communications in Computer and Information Science*, 2019.

[30] French Cybersecurity Agency (ANSSI). Profils de protection pour les systèmes industriels. `https://cyber.gouv.fr/publications/profils-de-protection-pour-les-systemes-industriels`, 2015. [Online ; accessed 30-May-2024].

[31] Anne Honkaranta, Tiina Leppänen, and Andrei Costin. Towards practical cybersecurity mapping of stride and cwe — a multi-perspective approach. In *2021 29th Conference of Open Innovations Association (FRUCT)*, pages 150–159, 2021.

[32] IEC 60848. GRAFCET specification language for sequential function charts. International standard, International Electrotechnical Commission, Geneva, CH, February 2013.

[33] IEC 61131-10. Programmable controllers - Part 10 : PLC open XML exchange format. International standard, International Electrotechnical Commission, Geneva, CH, July 2019.

[34] IEC 61131-3. Programmable controllers - Part 3 : Programming languages. International standard, International Electrotechnical Commission, Geneva, CH, May 2013.

[35] IEC 61131-5. Programmable controllers - Part 5 : Communications. International standard, International Electrotechnical Commission, Geneva, CH, February 2000.

[36] IEC 61850-8-1. Réseaux et systèmes de communication pour l'automatisation des systèmes électriques – Partie 8-1 : Mise en correspondance des services de communication spécifiques (SCSM) – Mises en correspondance pour MMS (ISO 9506-1 et ISO 9506-2) et pour l'ISO/CEI 8802-3. International standard, International Electrotechnical Commission, Geneva, CH, June 2011.

[37] IEC 62443-1-1. Industrial communication networks – Network and system security – Part 1-1 : Terminology, concepts and models. International standard, International Electrotechnical Commission, Geneva, CH, July 2009.

[38] IEC 62443-3-2. Security for industrial automation and control systems – Part 3-2 : Security risk assessment for system design. International standard, International Electrotechnical Commission, Geneva, CH, June 2020.

[39] IEC 62443-3-3. Industrial communication networks – Network and system security – Part 3-3 : System security requirements and security levels. International standard, International Electrotechnical Commission, Geneva, CH, August 2013.

[40] ISO 31000. Risk management - Guidelines. International standard, International Organization for Standardization, Geneva, CH, June 2018.

[41] Konstantin Izrailov, Mikhail Buinevich, Igor Kotenko, and Alexander Yaroshenko. Identifying characteristics of software vulnerabilities by their textual description using machine learning. In *2021 World Automation Congress (WAC)*, pages 186–192. IEEE, 2021.

[42] Abdelaziz Khaled, Samir Ouchani, Zahir Tari, and Khalil Drira. Assessing the severity of smart attacks in industrial cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 2020.

[43] Rafiullah Khan, Kieran McLaughlin, David Laverty, and Sakir Sezer. Stride-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017.

[44] Loren Kohnfelder and Garg Praerit. The threats to our products. `https://shostack.org/files/microsoft/The-Threats-To-Our-Products.docx`, 1999. [Online ; accessed 24-January-2024].

[45] S Kriaa, M Bouissou, and Y Laarouchi. A model based approach for scada safety and security joint modelling : S-cube. *IET Conference Proceedings*, 2015.

[46] Siwar Kriaa, Ludovic Pietre-Cambacedes, Marc Bouissou, and Yoran Halgand. A survey of approaches combining safety and security for industrial control systems. *Reliability Engineering & System Safety*, 2015.

[47] Rajesh Kumar. A model-based safety-security risk analysis framework for interconnected critical infrastructures. In *Critical Infrastructure Protection XIV : 14th IFIP WG 11.10 International Conference, ICCIP 2020, Arlington, VA, USA, March 16–17, 2020, Revised Selected Papers 14*, pages 283–306. Springer, 2020.

[48] Ralph Langner. Stuxnet : Dissecting a cyberwarfare weapon. *IEEE security & privacy*, 9(3) :49–51, 2011.

[49] S. Mesli-Kesraoui, A. Toguyeni, A. Bignon, F. Oquendo, D. Kesraoui, and P. Berruet. Formal and joint verification of control programs and supervision interfaces for socio-technical systems components. *(International Federation of Automatic Control*, 2016.

[50] MITRE. About CWE. `https://cwe.mitre.org/about/index.html`, 2024. [Online ; accessed 01-March-2024].

[51] National Institute of Standards and Technology. The NIST Cybersecurity Framework (CSF) 2.0. Technical Report NIST CSWP 29, National Institute of Standards and Technology, Gaithersburg, MD, February 2024.

[52] NIST. Official Common Platform Enumeration (CPE) Dictionary. `https://nvd.nist.gov/products/cpe`. [Online ; accessed 01-March-2024].

[53] Ocanis. open source version of the tool Teloco (Grafcet_to_Automaton). `https://github.com/ocanis/Grafcet_to_Automaton`, 2021. [Online ; accessed 22-April-2024].

[54] Ludovic Piètre-Cambacédès and Marc Bouissou. Modeling safety and security interdependencies with bdmp (boolean logic driven markov processes). In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2852–2861, 2010.

[55] PLCopen. XML Exchange. `https://plcopen.org/technical-activities/xml-exchange`, 2019. [Online ; accessed 10-April-2024].

[56] Tom Preston-Werner. Tom's Obvious Minimal Language. `https://toml.io/en/`, 2022. [Online ; accessed 29-February-2024].

[57] Julien Provost, Jean-Marc Roussel, and Jean-Marc Faure. Translating Grafcet specifications into Mealy machines for conformance test purposes. *Control Engineering Practice*, September 2011.

[58] Daniele Pugliesi. Functional levels of a distributed control system (dcs). `https://commons.wikimedia.org/wiki/File:Functional_levels_of_a_Distributed_Control_System.svg`, 2014. [Online ; accessed 22-July-2024 ; CC BY-SA 3.0, via Wikimedia Commons].

[59] Maxime Puys, Marie-Laure Potet, and Abdelaziz Khaled. Generation of applicative attacks scenarios against Industrial Systems. *Foundations and Practice of Security*, 2018.

[60] Marco Rocchetto and Nils Ole Tippenhauer. Towards formal security analysis of Industrial Control Systems. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

[61] Théo Serru. *Model-Based Security Assessment of Cyber-Physical Systems : Analyzing the Impact of Cyberattacks on Safety Using ALTARICA*. PhD thesis, EM2PSI, 2023.

[62] Adam Shostack. *Threat Modeling : Designing for Security*. Wiley Publishing, 1st edition, 2014.

[63] Kwang-Seop Son, Jae-Gu Song, and Jung-Woon Lee. Development of the framework for quantitative cyber risk assessment in nuclear facilities. *Nuclear Engineering and Technology*, 55(6) :2034–2046, 2023.

[64] Jae-Gu Song, Jung-Woon Lee, Cheol-Kwon Lee, Kee-Choon Kwon, and Dong-Young Lee. A cyber security risk assessment for the design of I&C systems in nuclear power plants. *Nuclear Engineering and Technology*, page 10, 2012.

[65] Keith Stouffer, Michael Pease, CheeYee Tang, Timothy Zimmerman, Victoria Pillitteri, Suzanne Lightman, Adam Hahn, Stephanie Saravia, Aslam Sherule, and Michael Thompson. Guide to Operational Technology (OT) Security. Technical Report NIST Special Publication (SP) 800-82, Rev.3, National Institute of Standards and Technology, Gaithersburg, MD, 2023.

[66] Jake Styczynski and Nate Beach-Westmoreland. When the lights went out : A comprehensive review of the 2015 attacks on ukrainian critical infrastructure. `https://www.boozallen.com/content/dam/boozallen/documents/2016/09/ukraine-report-when-the-lights-went-out.pdf`, 2019. [Online ; accessed 22-July-2024].

[67] Mathieu Turuani. The CL-ATSE protocol analyser. *Lecture Notes in Computer Science*, 2006.

[68] Tianyi Wang, Shengzhi Qin, and Kam Pui Chow. Towards vulnerability types classification using pure self-attention : A common weakness enumeration based approach. In *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*, pages 146–153. IEEE, 2021.

[69] Tao Wen, Yuqing Zhang, Ying Dong, and Gang Yang. A novel automatic severity vulnerability assessment framework. *J. Commun.*, 10(5) :320–329, 2015.

[70] Rune Winther, Ole-Arnt Johnsen, and Bjørn Axel Gran. Security assessments of safety critical systems using hazops. In Udo Voges, editor, *Computer Safety, Reliability and Security*, pages 14–24, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[71] William Young and Nancy Leveson. Systems thinking for safety and security. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, page 1–8, New York, NY, USA, 2013. Association for Computing Machinery.

[72] Qianxiang Zhu, Yuanqing Qin, Chunjie Zhou, and Weiwei Gao. Extended multilevel flow model-based dynamic risk assessment for cybersecurity protection in industrial production systems. *International Journal of Distributed Sensor Networks*, 14(6) :155014771877956, June 2018.