Modélisation et analyse formelle de modèles système pour les menaces persistantes avancées

Tithnara Nicolas Sun

DOCTORAT BRETAGNE LOIRE / MATHSTIC

ENSTA BRETAGNE

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES BRETAGNE

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

Par
Tithnara Nicolas SUN

Modélisation et Analyse Formelle de Modèles Système pour les Menaces Persistantes Avancées

Thèse présentée et soutenue à Brest, le 10 mai 2022
Unité de recherche : Laboratoire Lab-STICC, UMR, 6285

Rapporteurs avant soutenance :

Bruno MONSUEZ Professeur, ENSTA ParisTech
Brahim HAMID Professeur, IRIT, Université de Toulouse 2- Jean Jaurès

Composition du Jury :

Président :	Régine LALEAU	Professeur, Université Paris-Est Créteil
Examinateur :	Joël CHAMPEAU	Enseignant-chercheur, ENSTA Bretagne
Dir. de thèse :	Raul MAZO	Professeur, ENSTA Bretagne
Encadrant :	Ciprian TEODOROV	Enseignant-chercheur, ENSTA Bretagne

Invité(s) :

Lionel VAN AERTRYCK DGA-MI

Remerciements

"*Nani gigantum humeris insidentes*" - Bernard de Chartres

"Nous sommes des nains sur des épaules de géants" disent les intellectuels dont les avancées s'appuient sur les travaux de leurs illustres prédécesseurs. En tant que doctorant, je tenais à exprimer ma reconnaissance envers les géants qui m'ont permis d'accomplir cette thèse.

Je souhaite tout d'abord remercier mon encadrant de thèse, Ciprian TEODOROV pour ses conseils avisés. Son exigence, sa bienveillance et sa bonne humeur contagieuse ont créé l'environnement de travail dont j'avais besoin pour réaliser cette thèse.

J'adresse toute ma reconnaissance à mon directeur de thèse, Philippe DHAUSSY pour son écoute. Son expérience, ses remarques pertinentes et sa vision d'ensemble ont su guider mes premiers pas dans le monde de la recherche.

J'aimerais également remercier chaleureusement Joël CHAMPEAU, d'abord mon professeur m'invitant vers la thèse dans ma formation d'ingénieur puis mon collègue et mentor honoraire dans le laboratoire. C'est grâce à son aide inestimable pendant la rédaction que je vous présente ce manuscrit aujourd'hui.

Mes remerciements vont également à Joaquin GARCIA-ALFARO et Alain PLANTEC, les membres de mon comité de suivi individuel de thèse. Leurs retours constructifs et bienveillants m'ont permis de prendre confiance dans ma recherche.

Je remercie également Bruno MONSUEZ et Brahim HAMID, les rapporteurs de mes travaux pour avoir accepté de relire consciencieusement mon manuscrit.

Enfin je tenais à témoigner ma gratitude envers le Pôle d'Excellence Cyber, la Direction Générale de l'Armement et la Région Bretagne qui ont financé mes travaux durant ces longues années. Je remercie en particulier Lionel VAN AERTRYCK mon référent DGA-MI qui m'a accompagné dans la découverte et le développement de cette thèse.

Je voudrais également adresser mes remerciements à tous mes collègues. Merci Luka LE ROUX, Pascale GAUTRON, Sylvain GUÉRIN, Fahad GOLRA, Vincent RIBAUD, Annick BILLON-COAT et Michelle HOFFMAN. Je salue mes aînés doctorants qui m'ont montré la voie à suivre. Merci Cyrielle FERON, Théotime BOLLENGIER, Vincent LEILDE, Bastien DROUOT et Hannah BOENNING. Je souhaite bien du courage à mes collègues doctorants Louis MORGE-ROLLET, Grégoire DE BROGLIE, Émilien FOURNIER, Quentin DUCASSE, Nasreddine OULDEI TEBINA et Hiba HNAINI ! Et je remercie tous les autres que je n'ai pas cités, tous ceux que j'ai oubliés pendant l'écriture de ces lignes. Les pauses cafés passées en votre compagnie ont su enjoliver mes journées de travail à l'ENSTA.

Je souhaite pour finir remercier ma famille. Merci à mes parents Céline et François qui m'ont permis de poursuivre mes rêves depuis tout petit et qui me soutiennent encore maintenant. Merci à mon grand frère Anthony dont la réussite tant scolaire que professionnelle m'impressionne chaque jour. Ton exemple m'encourage toujours à devenir meilleur. Merci à ma petite sœur Amélie dont les prouesses artistiques, le parcours tout aussi exemplaire et la gentillesse font de moi le plus fier de tous les grands frères. Enfin merci à Stéphanie

qui partage ma vie depuis déjà une décennie. Ton soutien indéfectible, ton sourire et ta bonne humeur me permettent de traverser toutes les épreuves. Malgré la distance qui me sépare de la région parisienne, vous restez tous les jours dans mon cœur et c'est à vous que je dois cette réussite.

Enfin je tenais à dédier ces travaux à ma grand-mère Kuy Eng, décédée en 2020. Du fait des conditions sanitaires récentes, je n'ai pas pu honorer sa mémoire en personne. Aussi j'espère lui rendre hommage à travers ce manuscrit.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Introduction générale

1.1.1 Cyber-sécurité des systèmes industriels

Les systèmes industriels prennent une part grandissante dans la société actuelle. De plus, ces systèmes évoluent rapidement au rythme des avancées technologiques. Cette évolution s'accompagne d'une complexité croissante dans le fonctionnement des systèmes. La dépendance de la société sur ces systèmes et leur complexité font de l'industrie une cible majeure pour les attaquants du monde cyber. C'est pourquoi la cyber-sécurité des systèmes industriels est un enjeu capital dont traite le présent manuscrit.

Le domaine de la cyber-sécurité des systèmes industriels est un domaine très vaste et en continuelle expansion. Aussi nous limitons-nous à l'étude d'un type spécifique de menace, les menaces persistantes avancées ou *Advanced Persistent Threat* (APT). Ces menaces se définissent comme des attaquants ayant recours à des stratégies développées et d'énormes moyens. Leur mode opératoire complexe est un challenge pour les acteurs de la cyber-sécurité. Bien qu'il existe de nombreuses solutions pour se défendre contre ce type de menaces, celles-ci ne répondent que partiellement au problème.

Dans ce manuscrit, nous proposons de prendre du recul vis-à-vis des APT en étudiant spécifiquement comment ces dernières conçoivent leur stratégie. Ce faisant, nous espérons porter un nouveau regard pour mieux comprendre ces menaces et ainsi proposer de nouveaux moyens de défense.

1.1.2 Ingénierie des modèles

À cause de la grande complexité des systèmes industriels, il devient difficile voire souvent impossible de garantir leur sécurité en conduisant des analyses sur les systèmes réels dans leur intégralité. C'est là qu'intervient l'ingénierie dirigée par les modèles ou génie logiciel. L'ingénierie dirigée par les modèles (IDM) est utilisée dans les différentes phases du cycle de vie des systèmes industriels [47]. Elle permet notamment de conduire des analyses structurelles et comportementales sur les systèmes ainsi que des évaluations de performance [77]. Elle se base sur l'utilisation de modèles conceptuels définis comme "la description formelle de certains aspects du monde physique et social qui nous entoure à des fins de compréhension et de communication" [68].

La construction d'un ou plusieurs modèles du système permet de conduire des analyses formelles impossibles sur le système réel. On parle de *techniques de vérification formelle,* ou *méthodes formelles.* Elles permettent d'établir l'exactitude d'un système suivant une rigueur mathématique.

Dans ce manuscrit, nous proposons de mettre en application l'utilisation des modèles couplée à des analyses formelles dans le contexte de la cyber-sécurité contre les APT pour parvenir à des résultats de sécurité mathématiquement vérifiés. Ce faisant, nous espérons démontrer l'utilité de l'ingénierie dirigée par les modèles pour la cyber-sécurité.

1.2 Problématiques et contributions

La question posée dans ce manuscrit est la suivante : "Comment l'APT établit-elle sa stratégie ?" En effet, la cyber-sécurité des systèmes industriels est un sujet d'importance grandissante et critique de nos jours. D'une part, il est de plus en plus difficile de défendre des systèmes de plus en plus complexes. D'autre part les menaces persistantes avancées ont recours à des moyens de plus en plus sophistiqués pour nuire aux systèmes industriels. Aussi, il est intéressant de chercher à comprendre le processus de conception de stratégie de telles menaces pour mieux s'en défendre.

Pour résoudre ce problème, nous considérons que les menaces persistantes avancées agissent comme des stratèges militaires pour établir leur plan d'action. Nous conduisons notre approche pour répondre à deux problématiques intimement liées

1. Quel est le point de vue de l'attaquant de type APT sur le système ?
2. Comment à partir de ce point de vue l'APT conçoit-elle sa stratégie ?

Les contributions que nous proposons permettent, *in fine,* à un concepteur de conduire le processus d'élaboration de stratégie pour une mission donnée. Ce faisant, le concepteur produit une stratégie opérationnelle. La méthodologie proposée est entièrement outillée et validée par un cas d'étude réelLes contributions présentées dans ce manuscrit sont les suivantes :

1. Tout d'abord, nous adaptons la méthodologie militaire de l'Operational Design au contexte des APT. En nous basant sur la littérature, nous proposons une méthodologie pour construire la stratégie d'une APT sur les systèmes industriels.
2. Ensuite nous proposons une implémentation concrète de cette méthodologie au moyen d'un framework de fédération. Chaque phase de la méthodologie est implémentée selon un paradigme de modélisation entièrement supporté par le framework.
3. En particulier, nous concevons un nouveau langage de modélisation de système adapté à notre contexte, le langage DyPimca. Ce langage étend le langage de système Pimca par des aspects comportementaux.
4. Enfin, notre méthodologie de modélisation de stratégie de l'APT est validée sur un cas d'étude d'attaque de station de pompage d'eau.

1.3 Plan du manuscrit

Ce manuscrit comporte cinq chapitres présentant l'état de l'art du domaine considéré, les contributions théoriques et la validation de notre approche.

Le Chapitre 2 fait l'état de l'art du domaine. Il introduit le contexte de la cyber-sécurité des systèmes industriels. Ensuite, nous identifions un type de menace spécifique, les APT, dont le mode opératoire nécessite une étude approfondie. L'élaboration de stratégies de ces attaquants constitue le cœur de nos travaux. La troisième section aborde les méthodologies de planification militaire comme prisme de réflexion à travers lequel nous étudions les stratégies des APT. Enfin, une quatrième section s'attarde sur les problématiques techniques d'ingénierie dirigée par les modèles de nos travaux.

Le Chapitre 3 définit notre méthodologie d'élaboration de stratégies. La première section présente la méthodologie de manière abstraite. Les sections suivantes décrivent l'implémentation des différents processus de la méthodologie concrète.

Le Chapitre 4 valide la méthodologie que nous proposons. La première section traite un cas d'étude réel d'attaque de station de pompage d'eau. La seconde section dresse le bilan du cas d'étude, de la méthodologie et des outils.

Le Chapitre 5 conclut le manuscrit. Il rappelle le contexte puis récapitule les contributions apportées pour répondre aux problématiques de la thèse.

Chapitre 2

État de l'art

Sommaire

La cyber-sécurité des systèmes industriels est l'enjeu que nous traitons dans ce manuscrit. Aussi il est primordial de bien définir le cadre de notre étude. C'est pourquoi nous définissons un **système** de la manière suivante :

Définition 2.0.1 (Système)**.** Un ensemble de composants informatiques et/ou de communication ainsi que d'autres ressources qui supporte un ou plusieurs objectifs fonctionnels d'une organisation. Les ressources d'un système incluent tous les composants informatiques ainsi que les machines et les procédés physiques impliqués dans l'acquisition, le stockage, la manipulation, l'affichage et/ou le transfert d'informations ou encore le contrôle ou la surveillance de procédés opérationnels. Un système consiste en un ensemble de taille variable d'un ou plusieurs ordinateurs et leur ressources. Les composants d'un système ne sont pas nécessairement physiquement connectés. [75]

Dans une première partie 2.1, nous présentons le contexte global de l'étude, c'est-à dire les menaces cyber qui pèsent sur le monde industriel. Dans une deuxième partie 2.2, nous définissons et illustrerons ce qui constitue une menace persistante avancée, nous détaillerons également les différentes techniques existantes pour se défendre contre le problème des menaces persistantes avancées. Dans une troisième partie 2.3, nous étudions les méthodologies de stratégies militaires et nous argumentons en faveur de leur apport pour la cyber-sécurité contre les menaces persistantes avancées. Enfin, dans une quatrième partie 2.4, nous nous attardons particulièrement sur les problématiques de l'ingénierie dirigée par les modèles afin de présenter et justifier les besoins de modélisation dans le contexte des menaces persistantes avancées.

2.1 Contexte

Dans le contexte de la cyber-sécurité, les menaces sont définies par plusieurs organismes comme le National Institute of Standards and Technology américain (NIST) [79] et l'Organisation Internationale de Normalisation (ISO) [2] :

— "Toute circonstance ou événement ayant le potentiel d'impacter négativement les opérations de l'organisation (ceci inclut mission, fonctions, image, ou réputation), les biens de l'organisation ou les individus dans le système d'informations à travers l'accès non-autorisé, la destruction, la divulgation, la modification d'informations et/ou le déni de service. Ceci désigne également le potentiel d'un attaquant d'exploiter une faille dans le système d'informations." [79]

— "Cause potentielle d'un incident qui peut résulter en des dommages sur un système ou une organisation. " [2]

Dans le contexte des systèmes industriels, la menace cyber est un problème de plus en plus vital à adresser. La Figure 2.1 montre un exemple d'architecture classique d'un système de contrôle et d'acquisition de données (SCADA [48]) industriel inspirée de Ginter [32]. Le système est composé de trois niveaux séparés par des pare-feux : un niveau de système physique, un niveau de système de contrôle industriel (*Industrial Control System* ou ICS) et un niveau de système d'information (*Information Technologies* ou IT). Les pare-feux constituent les points d'échange entre les différents niveaux, ils filtrent les communications afin de garantir la sécurité du système.

Système physique : Ce système correspond au niveau industriel physique. Il est composé de plusieurs automates programmables industriels (*Programmable Logic Controller* ou *PLC*) contrôlant divers actionneurs physiques en fonction de données relevées par des capteurs et d'une commande, qui est un programme stipulant leur comportement. Les PLC communiquent avec un centre de contrôle SCADA au travers d'un large réseau.

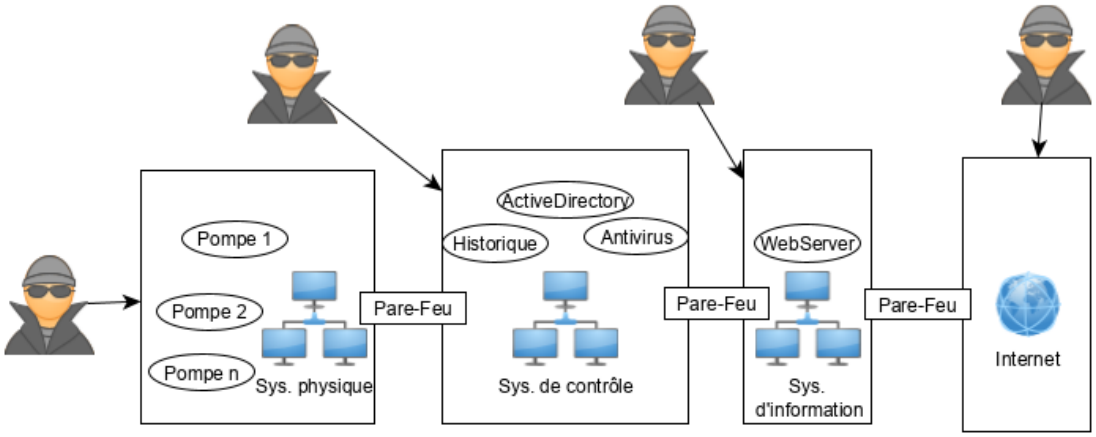


FIGURE 2.1 – Architecture classique d'un système de contrôle industriel

Système de contrôle industriel : Ce système supervise le niveau physique. Il est principalement composé d'ordinateurs et de quelques opérateurs. Il a pour but de surveiller le niveau physique, de relever différentes informations issues des différents PLC et de permettre le diagnostic et le contrôle-commande du site de l'usine. Ce contrôle est rendu possible grâce à une interface utilisateur pour les opérateurs du niveau ICS.

Système d'information : Ce système correspond au réseau informatique de l'entreprise. Il est relativement indépendant du niveau SCADA. Il est composé d'ordinateurs et d'employés de services indépendants de la production en usine.

D'une part, la surface d'attaque des systèmes, c'est-à-dire l'ensemble des points d'interactions possibles pour un attaquant (Manadhata et Wing [69]), s'étend à mesure qu'augmentent la complexité des systèmes et l'automatisation des procédés. Ceci entraîne une augmentation du nombre de failles exploitables par un attaquant dans le système. De plus, à mesure que les systèmes se complexifient, les algorithmes sur lesquels ils reposent et dont il faut assurer la sûreté et la sécurité de fonctionnement sont un réel enjeu pour la menace cyber.

D'autre part, les instances de cyber-attaques deviennent de plus en plus sophistiquées. En effet comme le montre le rapport technique de Ginter [32], les systèmes industriels sont les cibles de cyber-attaques de degré de complexité variable en employant différents vecteurs dont voici trois exemples :

Insider dans le système de contrôle industriel : Un employé du département IT travaille en réalité pour l'attaquant (*insider*). Il obtient les identifiants de ses collègues du département ICS lors d'une rencontre professionnelle usuelle. À l'aide de ces identifiants, l'*insider* se connecte sur la machine du réseau ICS à distance. Il explore le réseau ICS et commence à développer une copie de l'interface de l'usine. Malgré sa compréhension limitée du système industriel, l'*insider* envoie les commandes qui lui paraissent pouvoir causer le maximum de dommages à l'usine. Cette attaque a un niveau de sophistication faible car l'*insider* IT manque généralement de connaissance en architecture de système industriel. Il a cependant l'occasion d'utiliser l'ingénierie sociale pour obtenir facilement les identifiants requis pour mener à bien cette attaque. L'attaque a pour conséquences des incidents pouvant causer des pannes ou simplement de la confusion. Elle peut entraîner une inspection de l'usine touchée pour rétablir des configurations d'équipements correctes afin d'éviter tout dysfonctionnement futur.

Zero-Day Ransomware : Une agence de renseignement laisse une liste de vulnérabilités *zero-day* sur des systèmes d'exploitation ou des applications dans un centre de commande et contrôle en ligne. Ces vulnérabilités *zero-day* sont des failles inconnues des

constructeurs. Un groupe de hackers attaque le centre en ligne et obtient la liste de vulnérabilités. Cette liste est revendue à une organisation criminelle. Elle met au point un rançongiciel (*ransomware*) autonome qui se propage en utilisant des vulnérabilités *zero-day* d'applications de transfert de fichiers sur le système d'exploitation Windows. Le *ransomware* est simultanément distribué sur un ensemble de sites web compromis et se propage rapidement. Au niveau ICS, un transfert de fichier avec le réseau IT provoque la propagation du *ransomware* jusque dans le réseau ICS. Le *ransomware* chiffre l'ensemble du système ICS, causant un arrêt d'urgence de l'usine et des dommages physiques sur les équipements. Cette attaque a un niveau de sophistication très élevé en termes cyber et un niveau de sophistication faible en termes de système industriel. En effet, les recherches en sécurité découvrent régulièrement de nouvelles vulnérabilités *zero-day*. Ces informations sont parfois achetées par des agences de renseignement ou des groupes d'attaquants à des fins de cyber-attaque. Les conséquences de cette attaque sont l'arrêt brutal et inattendu du système industriel. Sa remise en marche à partir de sauvegardes prend en moyenne entre cinq et dix jours. De plus, il est possible que l'arrêt ait causé des dommages irréparables sur les équipements de l'usine, nécessitant un remplacement de pièces coûteuses et pouvant prendre plus de temps.

Stuxnet [59] : Des attaquants sophistiqués ciblent un site industriel spécifique et hautement protégé. Ils compromettent tout d'abord un service tiers moins défendu. Ce faisant, ils exfiltrent des données sur l'architecture du site industriel et son système de protection. Les attaquants développent ensuite un malware autonome pour atteindre les équipements du site industriel et causer des dommages physiques. Ce malware autonome exploite des vulnérabilités *zero-day*. Le fournisseur de service tiers implante le malware dans le système industriel à l'aide d'une clé USB. Les anti-virus du système ne détectent pas le malware car il exploite des vulnérabilités inconnues à ce jour. Cette attaque a un niveau de sophistication très élevé au niveau cyber avec l'exploitation de vulnérabilités *zero-day* et le développement d'un malware autonome. Elle a également un haut niveau de sophistication au niveau ingénierie des systèmes industriels pour comprendre et cibler précisément des procédés physiques spécifiques et éviter toute détection. Le site d'enrichissement d'uranium de Natanz ciblé par Stuxnet a connu de nombreux mois de production réduite voire nulle à cause des interférences du malware. On estime également que le site a souffert d'une usure prématurée et de la destruction de plusieurs centrifugeuses d'uranium. Plus généralement, cette classe d'attaque peut passer au travers de la plupart des systèmes de sécurité et peut causer de grands dommages.

Cette dernière décennie a vu naître des instances de cyber-attaques employant simultanément des vecteurs multiples comme l'ingénierie sociale et les failles logicielles. Ces cyber-attaques d'une complexité nouvelle font partie d'une classe qu'on nomme **menace persistante avancée** (APT, Advanced Persistent Threat).

De nos jours, les menaces sont de plus en plus difficiles à anticiper en raison de la complexité grandissante des systèmes et la sophistication croissante des attaques. Les menaces persistantes avancées en particulier posent un véritable défi pour les experts en cyber sécurité. Ces nouvelles menaces sont la cible du travail bibliographique du chapitre suivant.

2.2 Menaces Persistantes Avancées

Dans cette section, nous définirons la notion de menace persistante avancée. Puis, nous détaillons leur mode opératoire et nous discutons des différentes solutions proposées dans la littérature pour lutter contre les APT. Enfin nous concluons sur les limites constatées dans la littérature pour répondre au problème des APT.

2.2.1 Caractéristiques

Menace persistante avancée ou plus communément *Advanced Persistent Threat* dans la littérature internationale est un terme désignant une nouvelle classe de menace en cybersécurité. Le National Institute of Standards and Technology (NIST) [73] définit Advanced Persistent Threat (APT) comme suit :

Définition 2.2.1 (Menace persistante avancée)**.** Adversaire possédant un niveau d'expertise sophistiquée et une quantité significative de ressources qui lui permettent de créer des opportunités pour accomplir ses objectifs en utilisant de multiples vecteurs d'attaque (e.g., cyber, physique, dissimulation). Ces objectifs incluent typiquement d'établir et d'étendre des points d'ancrage dans l'infrastructure du système d'information de l'organisation visée à des fins d'ex-filtration d'informations, d'affaiblissement ou de sabotage de biens critiques pour une mission, un programme ou une organisation, ou simplement pour se positionner dans le système afin d'exécuter une opération d'attaque future. La menace persistante avancée : (i) poursuit ses objectifs de manière répétée sur une période de temps étendue ; (ii) s'adapte aux efforts de défenseurs du système ; et (iii) est déterminée à maintenir le niveau d'interactions requis pour atteindre ses objectifs.

Tankard [95] explique que le terme *avancée* indique le haut degré de sophistication d'une APT que ce soit en termes de compétences techniques qu'en termes de ressources disponibles. Le terme *persistant* quant à lui fait référence à l'approche lente et furtive typique des APT pour s'introduire dans le système et pour y établir puis y maintenir un point d'ancrage.

Contrairement à des menaces classiques, les APT présentent quatre caractéristiques distinctives identifiées par Chen *et al.* [13] : (i) des cibles spécifiques et des objectifs clairement définis ; (ii) des attaquants hautement organisés et ayant accès à de nombreuses ressources ; (iii) des campagnes sur le long terme avec des tentatives répétées ; et (iv) des techniques de dissimulation et d'évasion.

Cibles spécifiques et objectifs clairement définis : Contrairement à des menaces classiques, les APT sont connues pour se concentrer sur des cibles spécifiques. Cette focalisation implique un potentiel de dommages causés plus important pour la cible que dans des cas de cyber-attaques classiques qui, elles, visent un grand nombre de cibles pour espérer toucher un maximum de victimes. Les cibles d'APT sont variées et concernent de nombreux domaines tels que le service, le commerce, la finance, le divertissement, l'ingénierie, les gouvernements, la justice, les hautes technologies, la santé, les transports et l'aérospatial, pour citer les premiers secteurs d'activités concernés dans le rapport de Mandiant [70]. Quant aux objectifs définis, ceux-ci visent typiquement les biens critiques d'une entreprise ou d'une organisation afin de prodiguer un avantage concurrentiel ou stratégique à l'APT, comme par exemple le vol de propriété intellectuelle ou le vol de secrets militaires.

Attaquants hautement organisés et ayant accès a de nombreuses ressources : Les acteurs d'une APT sont multiples, coordonnés et compétents en piratage informatique. Ils font parfois partie de groupe gouvernementaux ou militaires [64]. D'autres sont simplement des mercenaires engagés par des entités étatiques ou privées [53]. Ce sont donc des menaces qui possèdent un haut niveau de compétences techniques et qui ont accès à

des ressources financières importantes. Ceci leur permet de mettre en place des stratégies d'attaque mettant en jeu une voire parfois même plusieurs vulnérabilités *zero-day* [59], c'est-à-dire des failles n'ayant fait l'objet d'aucune publication jusqu'alors. Ces failles sont notoirement difficiles à défendre puisqu'elles sont inconnues de la défense.

Campagnes sur le long terme avec des tentatives répétées : Une attaque APT est une campagne de longue durée. L'APT peut rester plusieurs mois ou plusieurs années dans le système avant d'être détectée. Les attaquants persistent et s'adaptent aux difficultés éventuelles qu'ils rencontrent dans la défense de la cible.

Techniques de dissimulation et d'évasion : Les APT sont capables de contourner les défenses du système pour rester furtives dans le réseau du système. Les actions de l'APT sont discrètes et maintiennent le niveau d'interaction au minimum pour rester dissimulé.

D'autres classifications de types d'attaquants en cyber-sécurité existent dans la littérature. C'est le cas de Choo [14] qui propose une classification en trois types de groupes de crime organisé en cyber-sécurité :

— L'organisation criminelle traditionnelle cherche à faire prospérer ses activités illicites réelles par le biais des technologies d'information et de communication. Elle est essentiellement motivée par le profit.

— Le groupe de pirates informatiques agit exclusivement par des moyens informatiques. Le haut niveau de sophistication de certains de ses membres s'apparente à celui des menaces persistantes avancées. Cependant, il s'agit généralement d'un groupe désorganisé sinon un ensemble d'individus aux compétences et aux motivations variables.

— L'organisation criminelle politique ou idéologique est un groupe particulièrement motivé pour nuire à une cible désignée. De fait, elle est prête à mener une campagne de longue durée pour parvenir à ses fins.

Dans cette classification l'APT est une organisation criminelle politique ou idéologique. Le mode opératoire d'une APT sera discuté dans la Section 2.2.2.

2.2.2 Mode opératoire et contre-mesures

Le mode opératoire d'une APT est celui d'une campagne qui s'étend sur le long terme. La campagne est minutieusement planifiée pour atteindre des objectifs précis, ce qui n'est pas sans rappeler les méthodologies opérationnelles militaires.

2.2.2.1 Cyber kill chain

Chen *et al.* [13] spécifient un modèle d'attaque ou *cyber kill chain* en six phases pour une attaque d'APT typique : (i) reconnaissance et armement ; (ii) acheminement ; (iii) intrusion initiale ; (iv) commandement et contrôle ; (v) mouvement latéral ; et (vi) ex-filtration de données. Ce modèle est basé sur les travaux de Hutchins *et al.* [46] sur la *cyber kill chain*, un framework de modélisation d'attaques inspiré des frameworks d'opérations militaires.

La cyber-sécurité contre les APT est un problème difficile à résoudre, notamment à cause des méthodes complexes employées par ce type d'attaquant. De part leur multiples vecteurs d'attaques, il n'y a pas de solution universelle pour la défense contre les APT. Brewer [10] suggère de faire face à ce problème en orientant la défense séparément sur chaque étape du mode opératoire des APT. Il est vrai que les outils de sécurité traditionnels peinent à combattre la menace persistante avancée dans son ensemble. En revanche, ils constituent un premier rempart qui peut être utilisé à bon escient pour contrer certaines phases des APT. Selon cet axe de réflexion, cette partie détaillera les différentes techniques de défense contre les APT.

Reconnaissance et armement : Cette phase correspond à la phase militaire du renseignement et de l'intelligence. C'est une étape de préparation cruciale dans laquelle des informations sur la cible sont organisées afin de mettre au point une stratégie, c'est-à-dire un plan d'attaque. Les informations rassemblées sur la cible portent sur la configuration du système ciblé, l'existence de potentielles failles, les personnalités influentes et leur coordonnées. Les sources de ces informations peuvent varier :

— Ingénierie sociale : manipulation psychologique de personnes liées à la cible afin d'obtenir des informations critiques ou de créer des opportunités d'actions dans le système.

— Open-Source Intelligence (OSINT) : ressources librement disponibles en ligne comme les coordonnées des employés d'une entreprise ou la configuration d'un site web [33].

— Intelligence étatique/militaire : renseignements confidentiels issus d'opération d'intelligence étatique ou militaire.

L'APT construit ensuite un plan d'attaque basé sur les connaissances rassemblées. La stratégie se base notamment sur l'exploitation de failles identifiées dans le système que l'attaquant est capable d'exploiter. Pour ce faire, l'APT se dote d'outils nécessaires à l'exploitation de failles comme Metasploit [54]. La stratégie élaborée en première instance prend en compte de multiples vecteurs d'attaques différents afin de pallier un éventuel imprévu.

Lors de cette phase, l'APT cherche à obtenir des informations sur le système de plusieurs manières pour s'armer en conséquence. Pour déjouer ce processus, il est tout d'abord important de sensibiliser le système et les différentes personnes qui y travaillent. En effet, le rapport de ISACA [50] stipule que 53.4 % des professionnels de la sécurité interrogés ignorent la différence entre une menace classique et une APT. Il est primordial de faire la distinction car les activités de reconnaissance d'une APT sont répétées et peuvent donc laisser des traces de leur passage [10]. Si, par exemple, un port est scanné à de multiples reprises depuis la même IP extérieure, il peut être utile de surveiller cette IP en particulier. Dans le cas d'une attaque interne, c'est-à-dire d'une opération de reconnaissance d'un agent du système ciblé par ingénierie sociale par exemple, il est crucial de surveiller les accès à des informations sensibles en temps réel. Pour ce qui est de l'armement, il est important de limiter les vulnérabilités du système au maximum. Cela passe par de la sensibilisation auprès des employés dans le système et des applications de correctifs réguliers pour limiter les failles non-humaines dans le système.

Acheminement : Cette phase correspond à l'acheminement jusqu'à la cible des différents malwares mis au point au préalable. L'acheminement peut être direct ou indirect. Dans le premier cas l'attaquant envoie directement sa charge utile à la cible en utilisant diverses techniques telles que le *spear phishing*, c'est-à-dire l'envoi de mail frauduleux à une cible choisie. À l'inverse du *phishing* classique, le mail est spécifiquement fabriqué pour toucher sa victime en particulier. Les informations obtenues lors de la reconnaissance augmentent significativement les chances de réussite de cette technique [31]. Dans le second cas, l'attaquant vise un tiers de confiance pour le système comme par exemple un site web fréquemment consulté par les employés de la cible.

Lors de cette phase, l'APT accède au périmètre du système. Bien qu'il existe de nombreuses méthodes pour y parvenir [10], le *spear-phishing* est la méthode principale utilisée par les instances d'APT observées et reconnues à ce jour [13, 20]. La première défense contre cette technique réside encore une fois dans la sensibilisation aux problématiques cyber pour prévenir l'ouverture d'un mail malveillant. Toutefois, étant donné les ressources de l'APT, il peut être difficile de distinguer un mail malveillant spécifiquement fabriqué pour un cible choisie. C'est pourquoi il est également nécessaire de spécifier une politique de sécurité classique (pare-feu [89], antivirus [106], système de détection d'intrusion [66]) au sein du système pour filtrer les mails ainsi que leurs pièces jointes. En effet, la charge

utile est attachée au mail frauduleux et requiert une action de la cible (ouverture de pièce jointe ou liens Internet) pour prendre effet. Ceci peut être prévenu en analysant les contenus des mails dans ce qu'on appelle une *sandbox* [83, 107, 108] c'est-à-dire une machine virtuelle isolée permettant d'expérimenter sur le contenu sans risquer de contaminer le réseau du système.

Intrusion initiale : Cette phase correspond au moment où l'APT parvient à faire son premier accès non-autorisé dans le système. C'est notamment lors de cette étape que l'APT établit son point d'ancrage dans le système à partir duquel elle poursuivra ses opérations dans le futur. L'APT a parfois recours à une exploitation de faille de type zéro-day pour y parvenir, mais il arrive également que la faille vise une application qui ne serait simplement plus à jour.

Lors de cette phase, l'APT exploite une vulnérabilité du système pour y établir un point d'ancrage. L'application régulière de correctifs peut empêcher l'APT de réussir à cette étape. Les systèmes de détection d'intrusion [66] peuvent également aider à cette tâche. Toutefois, l'APT peut avoir recours à l'exploitation de faille *zero-day* ce qui est impossible à détecter pour un système de détection d'intrusion basé sur les signatures de malwares connus. Pour pallier cela, des systèmes de détection d'intrusion avancés se basent sur le système de *sandbox* [83] pour détecter des comportements malveillants.

Commandement et contrôle : Cette phase consiste à établir un mécanisme de *Command and Control*, c'est-à-dire établir une liaison entre le point d'ancrage et l'extérieur du système. À partir de cette liaison, l'APT prend le contrôle de la machine infectée lors de l'intrusion initiale sans avoir à opérer directement dans le système. Pour éviter la détection, la liaison peut être établie en utilisant des services utilisés dans le système de manière légitime.

Lors de cette phase, l'APT installe un mécanisme de contrôle à distance de son malware à partir de l'extérieur du système. Ceci peut être prévenu par différents procédés comme le blocage de certaines requêtes vers des IP suspectes détectées lors de l'étape de reconnaissance. Cependant, les méthodes utilisées par les APT sont discrètes et évasives, il n'existe donc pas de pattern à détecter pour lutter systématiquement contre elles. Il faut donc allier les systèmes de détection d'intrusion avec des systèmes de gestion de l'information et des événements de sécurité (*Security information and event management* ou *SIEM*) [30]. Ceux-ci surveillent les événements dans le système et peuvent les corréler pour déterminer des comportements anormaux. Ils déterminent qu'un événement est anormal lorsqu'il sort de la limite acceptable d'un comportement d'un utilisateur normal du système.

Mouvement latéral : Cette phase, aussi appelée *Pivoting*, permet à l'APT de se déplacer dans le système pour étendre son contrôle sur d'autres machines et collecter/modifier/détruire des biens critiques. Elle correspond à une phase d'exploration du système à partir du point d'ancrage. Cette étape peut prendre du temps car l'APT cherche à rester dissimulée le plus longtemps possible dans le système.

Lors de cette phase, l'APT cherche à élever ses privilèges et à étendre son influence dans le système. Pour cela, elle scanne son environnement depuis son point d'ancrage et cherche à déterminer de nouvelles vulnérabilités à exploiter. Les techniques classiques de système de détection d'intrusion ainsi que les systèmes de gestion de l'information sont utilisés pour se défendre contre cela. De plus l'implémentation d'une politique de sécurité efficace, c'est-à-dire une politique de contrôle d'accès à l'information solide au sein du système [44, 57], permet de limiter l'expansion de l'APT dans le système.

Ex-filtration de données : Cette phase permet d'accomplir l'objectif principal de l'APT qui est, comme expliqué précédemment, le vol de données critiques. Divers moyens de compression et de cryptage peuvent être utilisés lors de cette étape pour minimiser le risque de détection [10].

Phase	Techniques	Contre-mesures
Reconnaissances et armement	Ingénierie sociale, OSINT, Intelligence, Metasploit	Sensibilisation, Correctifs, Pare-feu
Acheminement	*Spear phishing*	Pare-feu, Antivirus, Système de détection d'intrusion
Intrusion initiale	Exploitation de faille, Faille *zero-day*	Correctifs, Système de détection d'intrusion avancé
Commandement et contrôle	Outil d'accès à distance, Chiffrement	Système de détection d'intrusion, SIEM
Mouvement latéral	Élévation de privilège, Collecte de données	Système de détection d'intrusion, SIEM, Politique de sécurité
Ex-filtration de données	Compression, Chiffrement	Antivirus, Pare-feu, Solution DLP, Système de détection d'intrusion

Table 2.1 – Contre-mesures contre les APT

Lors de cette phase, l'APT cherche à extraire les données qu'elle a recueillies dans le système jusqu'à elle. Pour prévenir cette étape, l'utilisation de techniques de *Data Loss Prevention* (*DLP*) [67] est préconisée. Ce terme regroupe toutes les techniques qui visent à minimiser la fuite de données sensibles. Cela regroupe les techniques de sécurité classiques comme les antivirus, les pare-feu, les systèmes de détection d'intrusion par signature mais aussi les systèmes de détection avancés. De plus, il existe des solutions de DLP comme celle proposée par Schmid *et al.* [87], qui surveille les flux de données dans le système et agit en temps réel pour prévenir les fuites de données. La contre-mesure contre cette étape, bien que très fournie, requiert une compréhension poussée des biens critiques du système afin de définir une politique de sécurité adaptée au système.

La Table 2.1 résume les différentes techniques employées par les APT lors de chaque étape de leur mode opératoire ainsi que les différentes contre-mesures de la littérature contre elles. Faire face à un nombre restreint d'étapes à la fois est possible au moyen d'approches existant dans la littérature.

Certaines approches produisent des analyses d'impacts [80, 91]. Celles-ci considèrent la dynamicité du modèle et permettent de capturer le comportement du système pendant une attaque. Néanmoins, la modélisation de l'attaquant est insuffisante dans l'approche de Peterson [80] pour capturer des scénarios d'attaques complexes. Dans l'approche de Sultan *et al.* [91], le formalisme d'automates permet de représenter des impacts d'attaques plus complexes au prix d'une modélisation intrusive aux composants (ajouts d'états et transitions) ce qui demande une compréhension poussée du fonctionnement du système et pose des problèmes de pérennité. Il est donc difficile d'appliquer ces approches dans le contexte *ad hoc* des menaces persistantes avancées, où les attaquants n'ont qu'une connaissance partielle du système ciblé.

D'autres approches sont basées sur le formalisme d'arbres d'attaque [58, 82]. Les arbres d'attaques permettent de modéliser des scénarios d'attaque dans un formalisme expressif. Ces outils mettent en jeu de multiples niveaux d'abstraction, ce qui requiert une compréhension poussée du fonctionnement système et de ses vulnérabilités. Ceci ne convient pas non plus au contexte de modélisation *ad hoc* des menaces persistantes avancées.

Toutefois, la décomposition de la *cyber kill chain* peut être discutée. En effet, si le modèle de Chen *et al.* [13] comporte six étapes, le modèle de Hutchins *et al.* [46] sur lequel il est basé en comporte sept, puisqu'il sépare la phase de **reconnaissance** et celle d'**armement**. De plus, la dernière phase d'**ex-filtration de données** n'est pas toujours applicable dans le cas où l'objectif de l'APT n'est pas de récupérer des données. Par exemple, dans le cas

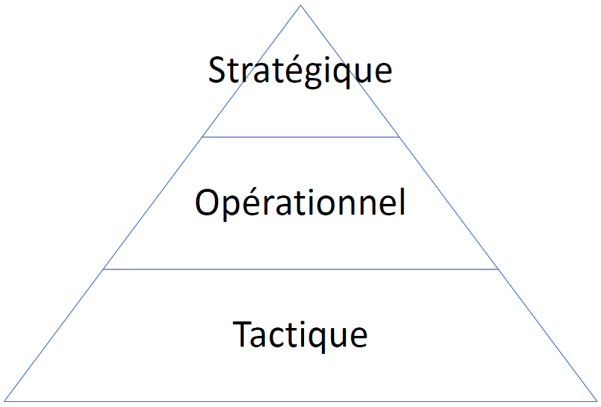


FIGURE 2.2 – Trois niveaux de guerre

de la célèbre attaque Stuxnet [59], classifiée comme APT, l'objectif premier était d'endommager prématurément des centrifugeuses. Ceci met en évidence certaines faiblesses du modèle de la *cyber kill chain* pour comprendre le fonctionnement des APT.

2.2.2.2 Vision holistique

Comprendre la méthodologie des APT implique de comprendre comment elles élaborent leur stratégie, c'est pourquoi il peut être utile de prendre du recul sur les techniques concrètes employées par les APT pour se focaliser sur un plus haut niveau d'abstraction. Certains travaux cherchent à répondre à ce problème en considérant les APT de manière holistique. C'est le cas des travaux de Rass *et al.* [84] qui définissent une défense basée sur la théorie des jeux. L'APT est considérée comme un joueur dont le gain peut être évalué à chaque action et les contre mesures suggérées pour la défense sont des choix optimaux en termes de retour sur investissement. D'autres travaux comme ceux de Hutchins *et al.* [46] étudient le comportement des APT de cas concrets passées et dérivent un plan d'action pour se prémunir d'attaques futures provenant d'une même source.

Le mode opératoire hautement organisé des APT n'est pas sans rappeler celui d'une organisation militaire. De plus, le concept même de *kill chain* provient du monde militaire Ceci peut également offrir une compréhension holistique des APT. Les forces armées divisent la guerre en trois niveaux : stratégique, opérationnel et tactique [104]. Cette division, historiquement issue des guerres napoléoniennes, permet une meilleure compréhension des causes et des conséquences du conflit à différentes échelles intrinsèquement liées. Bien que chaque niveau permette d'élaborer des stratégies en estimant les forces alliées et ennemis, le niveau de granularité diffère dans chacun d'eux.

Niveau stratégique de guerre : Ce niveau se concentre sur les politiques nationales ou internationales, il s'échelonne sur une durée longue d'une voire plusieurs années. Il considère le conflit dans son ensemble et établit des stratégies dans les grandes lignes. En des termes plus généraux, le niveau stratégique explique **pourquoi** les forces armées doivent intervenir, **avec quoi** elles doivent combattre et **pourquoi** l'adversaire se bat.

Niveau opérationnel de guerre : Ce niveau se concentre sur les forces armées sur le théâtre d'opérations dans leur ensemble, il s'échelonne sur une durée moyenne allant d'un à plusieurs mois. Il se situe au niveau de la campagne militaire dans son ensemble. Une campagne stipule comment atteindre un ou plusieurs objectifs de niveau stratégique en organisant les forces armées et les opérations militaires majeures. Le commandant doit coordonner les opérations en identifiant les forces et les ressources mises en jeu. La stratégie intègre plusieurs missions de niveau tactique, c'est pourquoi le niveau opérationnel fait

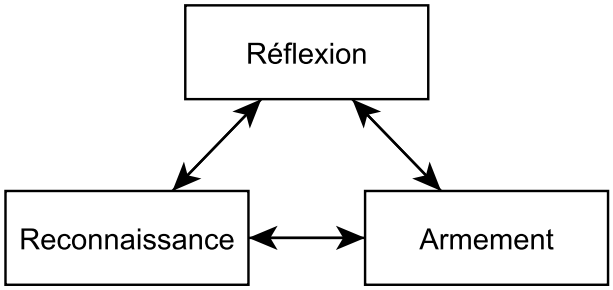


FIGURE 2.3 – Trois processus itératifs de la reconnaissance à l'armement

le pont entre le niveau stratégique et le niveau tactique. En des termes plus généraux, le niveau opérationnel détermine **quoi** faire, dans **quel ordre**, pendant **combien de temps** et avec **quelles ressources**.

Niveau tactique de guerre : Ce niveau se concentre sur les opérations individuelles, il s'échelonne sur une durée courte allant jusqu'à quelque jours. L'ensemble des actions militaires concrètes se situe à ce niveau, bien qu'il puisse avoir des effets à un niveau opérationnel ou tactique. En des termes plus généraux, le niveau tactique explique **comment** combattre.

Il est intéressant de noter le parallèle qui existe entre les niveaux de guerre et le mode opératoire des APT décrit par la *cyber kill chain*. En effet, la plupart des phases décrivent un niveau tactique d'opérations militaires puisqu'elles stipulent les moyens concrets qui permettent aux APT de passer à l'action. Cependant, on peut noter que la phase **reconnaissance et armement** est plus difficile à classer dans le niveau tactique, car elle comprend un processus de réflexion de plus haut niveau, à savoir le niveau opérationnel. Afin d'avoir une vue d'ensemble sur la méthodologie des APT, il est crucial de se placer au niveau opérationnel qui englobe l'ensemble de la mission d'une APT.

2.2.3 Limites

De nombreuses approches de sécurité répondent déjà à la plupart des phases de la *cyber kill chain* des APT. Toutefois, il apparaît que l'étude et les contre-mesures contre l'étape de **reconnaissance et armement** soient encore peu développées. La seule solution proposée est la prévention via correctifs et la sensibilisation aux contraintes cyber qui permettent de pallier l'ingénierie sociale. Toutefois, cette solution ne considère pas la **reconnaissance et armement** dans son entièreté. Cela s'explique par le fait que cette phase fait intervenir des compétences relativement étrangères à la cyber sécurité : celles de la méthodologie opérationnelle, notamment militaire. Il semble néanmoins crucial de mieux comprendre cette phase de la *cyber kill chain* car elle précède toutes les autres et se situe en partie à un plus haut niveau de réflexion stratégique.

Il est d'ailleurs notable que l'étape de **reconnaissance et armement** fasse en réalité intervenir des processus issus de domaines de différentes natures, à savoir la reconnaissance, la réflexion et l'armement. Il paraît donc judicieux de l'étudier selon ces différents axes pour proposer une contre-mesure plus complète et efficace. La Figure 2.3 représente les trois procédés itératifs qui interviennent à cette étape.

Reconnaissance : lors de ce processus l'APT recueille des informations sur le système ciblé par différents moyens : l'ingénierie sociale, l'OSINT, les techniques du renseignement auquel elle a accès. Ces informations lui permettent de modéliser le système.

Réflexion : lors de ce processus l'APT construit une stratégie, c'est-à-dire un plan d'action. Elle détermine une ou plusieurs cibles dans le système qu'elle imagine vulnérables en fonction de diverses informations techniques (configurations de différents éléments du

système accessibles depuis l'extérieur, suite logicielle utilisée) et d'informations sociales (adresses mail, profils du personnel). Ce faisant, l'APT peut déterminer un manque d'informations, ce qui peut entraîner une nouvelle phase de reconnaissance. Sinon, l'APT peut poursuivre jusqu'à la concrétisation.

Concrétisation, armement : lors de ce processus l'APT détermine le faisabilité de la stratégie qu'elle prévoit dans le processus précédent. En fonction des failles connues, l'APT détermine concrètement si son plan d'action lui semble réalisable. Elle concrétise la stratégie avec des exploitations de failles issues d'outils comme Metasploit [54]. Si la stratégie ne semble pas réalisable, la réflexion reprend.

La phase de **reconnaissance et armement** peut être scindée en trois processus itératifs : l'observation du système, la formulation d'une stratégie et la concrétisation de la stratégie. Il est intéressant de remarquer que la littérature ne s'occupe que de deux de ces processus itératifs, à savoir la reconnaissance à travers la sensibilisation et les pare-feux, et la concrétisation avec l'application régulière de correctifs. Les stratégies des APT sont considérées comme des "boîtes noires" prenant en entrée des données sur le système et produisant en sortie un scénario d'attaque. L'étape de réflexion et de formulation de stratégie manque de contre-mesures. Pourtant, il semble intéressant de mieux la comprendre car c'est justement le processus de *réflexion* qui organise l'ensemble de la mission et qui permettrait donc d'avoir un point de vue holistique sur les APT. Or il semble indispensable de mieux comprendre l'étape de réflexion et de formulation de la stratégie car c'est justement le processus de *réflexion* qui organise l'ensemble de la mission. Un point de vue holistique sur les APT au niveau stratégique est nécessaire pour mieux comprendre et analyser leur fonctionnement et ainsi en déduire les contre-mesures. Ce constat plaide pour la prise en compte d'abord d'un niveau système des APT avant d'envisager toute concrétisation de la mission.

Bien qu'elles puissent manquer dans la littérature de cyber-sécurité, les théories de stratégies abondent dans la littérature militaire. Dans la Section 2.3, nous détaillerons la méthodologie militaire de l'*Operational Design* pour mieux comprendre la méthodologie adoptée par une APT lors de l'étape de **réflexion** de la phase de **reconnaissance et armement**.

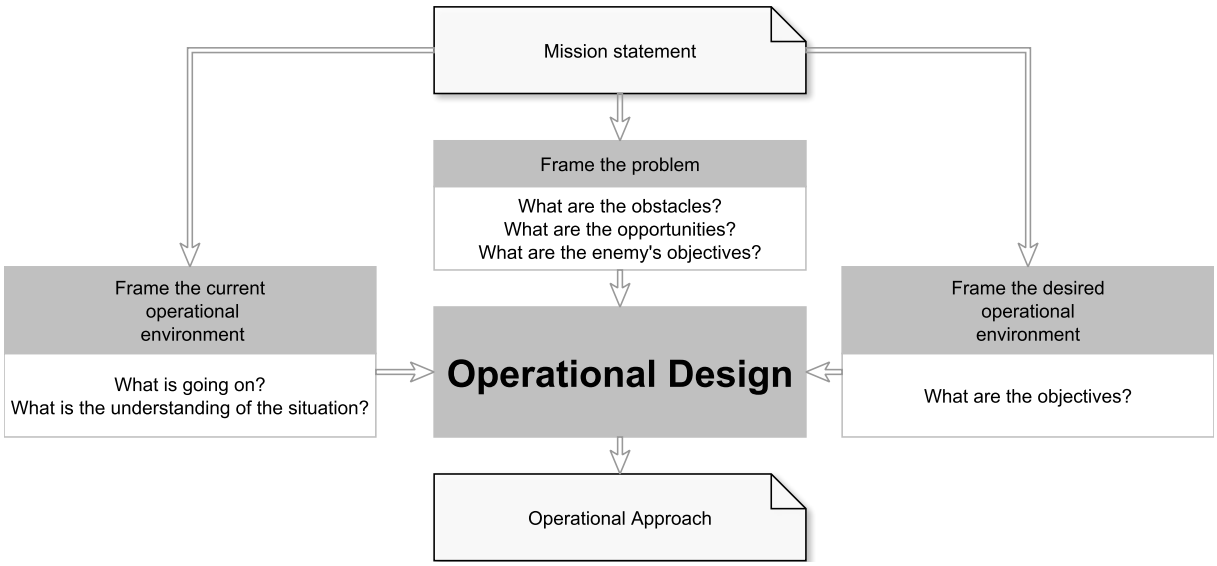


FIGURE 2.4 – Méthodologie de l'Operational Design, inspirée de JP 5-0

2.3 Cyber Operational Design

Puisque la phase de **reconnaissance et armement** consiste à élaborer une stratégie d'attaque complexe, il est intéressant d'étudier les méthodologies militaires qui permettent de répondre à cette problèmatique.

2.3.1 Operational Design

La conception opérationnelle, plus communément appelée **Operational Design**, est définie dans *Joint Publication (JP 5-0), Joint Operation Planning* comme "la conception et la construction d'un cadre de travail qui soutient la campagne ou l'opération et son exécution." [102]. Lorsqu'un commandant implémente la stratégie d'une opération militaire, l'Operational Design en est son plus haut niveau d'implémentation. L'Operational Design repose sur l'Operational Art défini dans JP 5-0 :

Définition 2.3.1 (Operational Design)**.** L'approche cognitive des commandants et de leurs équipes, soutenue par leurs compétences, leurs connaissances, leurs expériences, leur créativité et leur jugement, pour développer des stratégies, des campagnes et des opérations de forces militaires en intégrant les objectifs, les méthodes et les moyens.

La méthodologie de l'Operational Design est détaillée dans plusieurs publications militaires [12, 26, 41, 101, 102, 105]. Cette méthodologie d'un haut niveau d'abstraction vise à développer une approche opérationnelle, *i.e.* une stratégie qui permet d'accomplir un objectif à partir d'un environnement et d'obstacles définis. L'Operational Design gravite autour de trois processus continus, simultanés et récursifs présentés en Figure 2.4.

— Définir l'environnement opérationnel courant (**C**urrent **O**perational **E**nvironment ou **COE**).

— Définir le problème.

— Définir l'environnement opérationnel souhaité (**D**esired **O**perational **E**nvironment ou **DOE**).

Définir le COE : ce processus est détaillé dans *Joint Intelligence Preparation of the Operational Environment (JIPOE) process* [103]. Son objectif est d'identifier et de caractériser l'environnement opérationnel, (*Operational Environment ou OE* défini selon sept

aspects distincts : (1) identifier la zone d'opération des forces alliées ; (2) analyser la mission et les intentions du commandant ; (3) déterminer les caractéristiques pertinentes de l'environnement ; (4) identifier les limites des zones d'intérêt des forces armées ; (5) déterminer le niveau de granularité requis et possible dans le temps imparti ; (6) déterminer le renseignement et les informations prioritaires, manquants et/ou incomplets ; et (7) collecter les données et soumettre des requêtes pour obtenir des informations complémentaires nécessaires aux analyses.

Définir le problème : ce processus a pour objectif de comprendre et de passer en revue les tendances et potentielles actions des différents acteurs de l'environnement. Ce faisant, il est possible de déterminer la cause première qui empêche l'OE d'atteindre l'état désiré. La compréhension du problème requiert de multiples niveau de réflexion puisqu'il faut : déterminer le contexte stratégique et la nature systémique du problème, synthétiser les directives stratégiques, identifier les tendances stratégiques, identifier les informations manquantes et les suppositions faites sur le problème et identifier le problème à un niveau opérationnel [99].

Définir le DOE : ce processus a pour but de décrire comment l'OE doit être modifié pour atteindre l'état désiré. Pour cela, il faut décrire les différents objectifs de la mission étape par étape, c'est-à-dire expliciter pour chaque tâche : qui, quoi, quand, où et pourquoi [99]. Ainsi, le commandant formule une approche opérationnelle (*Operational Approach*) de la mission. Celle-ci dépend largement du cadre de travail choisi. C'est pourquoi elle peut prendre bien des formes que ce soit textuelles ou graphiques.

L'annexe B de *Army Design Methodology* [101] montre un exemple de la méthodologie d'Operational Design dans le contexte d'un réseau de trafic d'opiacés lié à une faction rebelle. Les acteurs identifiés dans cet exemple ne sont proposés qu'à titre d'illustration. L'objectif du concepteur est de développer un plan d'action, ou approche opérationnelle, afin de perturber les relations entre les différents acteurs du réseau. L'environnement opérationnel est modélisé sous la forme d'un schéma présenté en Figure 2.5, accompagné d'explications narratives textuelles. Le concepteur formule son approche opérationnelle à partir d'informations rassemblées au préalable.

La Figure 2.5 représente les différents acteurs de l'environnement sous la forme de nœuds numérotés. Ces acteurs sont dans l'ordre : (1) producteur d'opium, (2) courtier d'opium/marché local, (3) laboratoire, (4) réseau de contrebande, (5) chimiste, (6) marchés étrangers (Europe), (7) barons de la drogue, (8) banques, (9) Hawaladeer (personne), (10) dirigeant politique et (11) Taliban. Dans cet exemple, le concepteur modélise le réseau de trafic de drogue à travers ces différents acteurs, dont il identifie les relations.

Plus précisément, les producteurs fournissent l'opium aux marchés locaux et des fonds pour les Taliban. Les marchés locaux fournissent des fonds aux producteurs et aux Taliban. Ils fournissent également de l'opium aux laboratoires, ainsi que des précurseurs de l'héroïne.

Les laboratoires fournissent de l'héroïne aux réseaux de contrebande. Les réseaux échangent des armes et du personnel avec les Taliban, fournissent les précurseurs aux marchés locaux et fournissent les marchés étrangers en héroïne. Les chimistes travaillent dans les laboratoires à transformer l'opium en héroïne à l'aide de précurseurs.

Les marchés étrangers fournissent des fonds aux banques et à Hawaladeer. Ces trois entités fournissent également des fonds aux barons de la drogue. Les barons de la drogue gèrent les laboratoires, leurs chimistes et les réseaux de contrebande. Ils fournissent des fonds aux dirigeants politiques en échange d'une protection politique. Enfin, ils fournissent des fonds aux Taliban.

À partir de cet environnement, le concepteur élabore différentes contre-mesures pour perturber le réseau de trafic de drogues. Elles sont de quatre types différents :

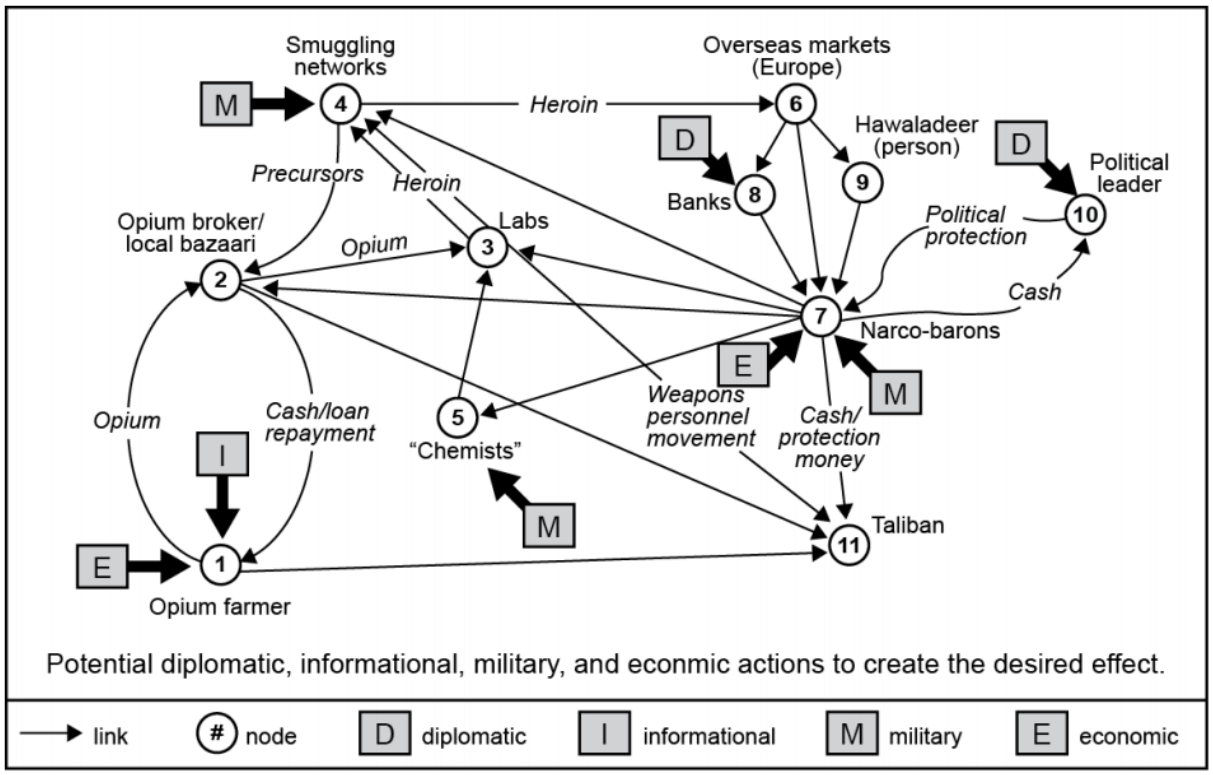


FIGURE 2.5 – Environnement opérationnel issu de *Army Design Methodology* [101]

— L'action diplomatique consiste à appliquer une pression sur les dirigeants politiques pour interrompre la protection qu'ils offrent aux barons de la drogue, et à partager des informations avec les banques internationales pour retracer les flux de capitaux.

— L'action informationnelle consiste à cibler les producteurs d'opium par des approches de communication pour les pousser à accepter une alternative à la production d'opium.

— L'action militaire consiste à capturer les barons de la drogue ou les chimistes, à détruire les laboratoires ou à démanteler les réseaux de contrebande pour empêcher l'accès aux précurseurs.

— L'action économique consiste à geler les actifs des barons de la drogue auprès des banques internationales et à travailler avec les pays de la région pour proposer des alternatives économiques à la culture d'opium.

Cette analyse montre comment la méthodologie d'Operational Design peut aider à produire une stratégie opérationnelle pour un problème donné. Dans le contexte des APT, l'Operational Design et la planification militaire sont pertinents pour mieux comprendre la méthodologie des APT lors de l'étape de **reconnaissance et d'armement**. En effet, leur sophistication leur permet de créer des stratégies complexes pour atteindre leur cibles. En d'autres mots, la méthodologie d'une APT est semblable à une méthodologie d'opérations militaires. Étudier l'Operational Design offre donc de nouvelles perspectives à la modélisation de sécurité dans le contexte des APT.

2.3.2 Planification d'opérations cyber

Certains efforts ont d'ores et déjà été entrepris pour adapter l'Operational Design au monde de la cyber-sécurité : on parle de Cyber Operational Design [52]. En effet, l'armée américaine [100] souligne l'émergence d'un nouveau front d'opérations après la terre, la

mer, l'air et l'espace : l'espace cyber. Ce nouveau front interagit avec les concepts plus classiques manipulés en Operational Design, comme les aspects politiques, militaires, économiques, sociaux ou les infrastructures [39, 40]. Il faut donc considérer l'espace cyber comme étant une partie intégrante de l'environnement opérationnel.

Al-Ahmad [4] désigne comme "cyber guerre" le conflit qui prend place dans l'espace cyber et qui a des répercussions dans la vie réelle. Ceci étend le théâtre des opérations militaires au réseau et non plus simplement sur le champ de bataille [65].

Afin d'adapter l'Operational Design, Karaman *et al.* [52] reprennent des processus de la méthodologie comme l'analyse de centre de gravité, c'est-à-dire l'identification des points critiques dans le système ciblé. Cependant, cette méthodologie reste abstraite dans le sens où elle n'est pas appliquée sur un exemple concret présent ou passé. La méthodologie proposée repose essentiellement sur les compétences du concepteur. Ceci est problématique car la planification d'opérations militaires requiert des compétences notoirement éloignées des connaissances en technologie de l'information et de la communication. L'applicabilité du Cyber Operational Design reste donc à prouver sur un cas concret.

À l'inverse, Williams [110] défend une approche basée sur l'Operational Design classique. Il suggère que le terme "cyber guerre" et le terme "cyber" en général ajoutent de la confusion dans l'approche. Il préconise une méthodologie s'appuyant sur les compétences stratégiques du commandant des opérations pour s'adapter aux contraintes cyber comme il s'adapte aux autres environnements (aérien, terrestre, maritime et spatial).

Ce faisant, quatre axiomes sur lesquels il faut développer la méthodologie sont préconisés :

1. Axiome 1 : il faut proscrire le terme "cyber guerre" car il est contre-productif. La guerre ne change pas de nature dans l'espace cyber, seules quelques spécificités liées au domaine s'ajoutent à la conception de stratégie. C'est pourquoi il n'est pas nécessaire de distinguer la guerre dans ce domaine en particulier.
2. Axiome 2 : la doctrine actuelle s'accommode avec aisance des opérations dans l'espace cyber. Il n'est pas nécessaire d'inventer une nouvelle méthodologie.
3. Axiome 3 : il faut former des opérateurs de l'espace cyber pour pouvoir traiter ce nouveau front comme les autres fronts. Comme dans les autres espaces de conflit, Williams [110] propose de former les officiers pendant au moins 10 ans à résoudre des problèmes tactiques dans tous les aspects de l'espace cyber.
4. Axiome 4 : le mot "cyber" doit être utilisé avec parcimonie. Seul le terme "espace cyber" est clairement défini, les autres combinaisons tendent à compliquer le discours.

Cette approche repose essentiellement sur les compétences du commandant des opérations. Cependant, l'axiome 3 souligne un manque d'effectifs qualifiés pour mettre en place les opérations dans l'espace cyber. Cette approche semble donc trop immature à l'heure actuelle et dans un futur proche pour combattre les menaces persistantes avancées par la méthodologie de l'Operational Design.

2.3.3 Pimca

D'autres initiatives posent les fondations d'une approche concrète basée sur la planification militaire dans un contexte de cyber-sécurité. C'est le cas du langage Pimca introduit par la Direction Générale de l'Armement (DGA) en France.

La présente section introduit Pimca [92], un langage de modélisation de systèmes pour la sécurité de l'espace cyber dont l'objectif principal est de souligner la surface d'attaque du système. Prenons l'exemple simple, présenté en Figure 2.6, d'un technicien utilisant une imprimante pour imprimer un document. Ce sous-système est situé dans le réseau

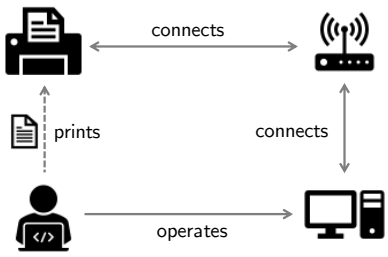


FIGURE 2.6 – Exemple fil rouge : Système d'impression de documents

d'information de l'architecture globale d'un système SCADA. La topologie du système est la suivante : un technicien, un poste de travail et une imprimante connectés à un réseau local (*Local Area Network ou LAN*). Le technicien opère sur son poste de travail et envoie des données depuis celui-ci jusque dans le réseau. Ces données sont ensuite transmises à l'imprimante qui exécute l'impression. Cet exemple met en évidence plusieurs aspects de l'architecture des systèmes : d'une part l'interaction entre les composants du monde *cyber* comme le réseau LAN et les composants du monde *physique* comme le technicien ; d'autre part les différentes natures des relations du langage Pimca, *concrète* dans le cas d'"opérer" ou d'"être connecté" et *conceptuelle* dans le cas d'"imprimer".

Le modèle Pimca capture la structure du système étudié. Un système est considéré comme la composition de divers macro structures dans le contexte des systèmes cyber-physiques, des systèmes de contrôle industriel ou plus généralement des systèmes de systèmes [62]. Pimca modélise ces divers composants communicants et interdépendants par l'abstraction de *knowledgeComponent*.

Un *knowledgeComponent* est défini et identifiable de manière unique, il possède un attribut *adresse* qui peut être soit physique, soit virtuelle ou soit générique. Un *knowledgeComponent* est une méta-classe abstraite possédant trois sous-classes concrètes : *machinery*, *resource* et *relation*. Ces trois types de *knowledgeComponents* seront détaillés dans ce qui suit.

2.3.3.1 Machinery

Une **machinerie** (***Machinery***) définit un élément actif du système. Les instances de *machinery* ont la capacité d'intéragir avec les autres éléments du systèmes au travers des relations ; *i.e.*, ils peuvent déclencher des actions et réagir à d'autres actions. Ce concept est utilisé pour capturer les éléments pourvus de comportement dans le système. Il est important de considérer les actions et les effets déclenchés par les comportements de ces éléments comme des étapes d'attaques potentielles pour évaluer les opportunités d'attaque.

Une machinerie possède un attribut *dimensionType i.e.*, physique, virtuel ou composite. La dimension définit l'espace dans lequel les machineries peuvent interagir, c'est-à-dire qu'une machinerie avec une dimension physique a une réalité tangible et ne peut interagir directement qu'avec d'autres machineries physiques. À l'inverse, une machinerie virtuelle n'a pas de réalité tangible et ne peut interagir qu'avec d'autres machineries virtuelles. Une machinerie de dimension composite fait le lien entre les dimensions physiques et virtuelles. Cette distinction est utile pour une APT car ses opérations dans le système commencent par un nœud physique ou virtuel. Pour progresser dans une autre dimension du système, l'attaquant est contraint de passer à travers une machinerie composite.

La machinerie est spécialisée pour représenter : des exécutants (*performers*), des interfaces (*interfaces*), des points de contrôle (*checkpoints*) et des réseaux (*networks*)

— L'**exécutant** (*Performer*) représente un élément humain. Ce concept permet de capturer le comportement particulier des êtres humains par opposition avec les éléments automatisés. L'exécutant expose une faiblesse particulière du système à l'ingénierie sociale notamment. L'exécutant a par nature une existence physique, il est donc modélisé comme une *machinerie physique*. Dans l'exemple illustratif, le technicien qui travaille dans le système est un *performer*. Le concept d'exécutant peut également être spécialisé davantage en attaquant (*Attacker*) qui représente un acteur humain malveillant.

— L'**interface** (*Interface*) représente le concept qui fait le lien entre deux dimensions, physique ou virtuelle. Ce concept est vital pour l'élaboration de stratégies de haut niveau car il caractérise la surface d'attaque définie par [98] et [69]. Les interfaces sont les seuls concepts de Pimca qui peuvent être de dimension composite *i.e.*, elles peuvent interagir à la fois avec l'espace physique et l'espace virtuel. Toutefois, ce n'est pas nécessaire dans Pimca qui autorise les interfaces entre deux dimensions du même type. Pour imposer un sens, le concept d'interface peut être spécialisé en **points d'entrée** ou **de sortie** (*EntryPoint* ou *ExitPoint* respectivement). Par exemple, un clavier peut être modélisé par un point d'entrée du monde physique au monde virtuel alors qu'un écran peut être modélisé comme un point de sortie du monde virtuel au monde physique. Le poste de travail de l'exemple est modélisé par une *interface* entre l'espace physique du technicien et l'espace virtuel du réseau LAN.

— Le **point de contrôle** (*Checkpoint*) représente un élément qui requiert une *ressource* spécifique pour être franchi. Dans le contexte des APT, ce concept permet de modéliser les éléments qui ralentissent/empêchent la progression de l'attaque. Les points de contrôle peuvent être soit physiques soit virtuels. Par exemple, un passage gardé à l'entrée d'un site industriel est un point de contrôle physique alors qu'un pare-feu réseau est un point de contrôle virtuel.

— Le **réseau** (*Network*) est un concept qui représente les points d'échange dans le système. C'est un concept essentiel à modéliser dans le contexte des APT car il donne accès à de nombreux *knowledgeComponents*. Accéder à un réseau en particulier est une étape-pivot pour les attaquants qui leur permet d'enclencher un mouvement latéral. Dans un réseau physique, les entités interagissent à un niveau physique et ne peuvent être atteintes que par des moyens physiques. Dans un réseau virtuel, les entités interagissent à un niveau virtuel et ne peuvent être atteintes que par des moyens virtuels. La nature des échanges est définie par l'attribut *networkType*, *i.e.* énergie, matière ou données. Par exemple un réseau électrique est un réseau physique qui transfert de l'énergie, alors qu'un réseau LAN est virtuel et transfère des données.

2.3.3.2 Resource

Une **ressource** (***Resource***) définit un élément passif du système. Par opposition aux machineries, les ressources ne déclenchent pas d'actions. Elles peuvent simplement être les cibles d'actions déclenchées par des machineries. Ce concept représente les biens critiques du système qui peuvent constituer des cibles pour l'APT. Par exemple, l'eau d'une station de pompage constitue une ressource.

Si une ressource peut être définie telle quelle, il existe deux spécialisations de ressources *i.e.* les passeports (*passports*) et les instructions (*instructions*). Ces ressources sont définies de la manière suivante :

— Le **passeport** (*Passport*) représente la ressource directement liée à un point de contrôle spécifique. L'accès au passeport est nécessaire pour passer le point de contrôle. Ce

concept est utile dans le contexte des APT car il capture les clés de passage de point de contrôle qui constituent des objectifs secondaires pour l'attaquant.

— L'**instruction** (*Instruction*) représente la ressource utilisée par une machinerie pour paramétrer son comportement. Ce concept symbolise le fait que le comportement d'une machinerie soit potentiellement une cible pour l'attaquant. Si une instruction est modélisée dans le système, elle peut être modifiée par l'attaquant. Par exemple, la politique de sécurité du réseau LAN de l'exemple est une instance d'instruction.

2.3.3.3 Relation

Une des spécificités du langage Pimca réside dans la définition riche du concept de relation afin de prendre en compte différentes sémantiques. Une **relation** (***Relation***) modélise les interactions complexes entre différents *knowledgeComponents*. Un relation possède une unique *source* et une unique *cible*, toutes deux étant de type *knowledgeComponents*. Une relation utilise différents attributs pour caractériser sa nature complexe. Une relation peut utiliser (*employing*) des machineries intermédiaires dont elle dépend. Dans l'exemple, c'est le cas de la relation d'impression entre le technicien et l'imprimante qui dépend du poste de travail et du réseau LAN. Une relation peut transporter (*conveying*) une ressource. Dans l'exemple, c'est le cas de la ressource "document" dans la relation d'impression. Une relation peut passer au travers (*passingThrough*) d'autres relations concrètes. Dans l'exemple, la relation d'impression est abstraite et s'exprime au travers des différentes relations concrètes entre le technicien et son poste de travail (*operates*), entre le poste de travail et le réseau (*connected to* et entre le réseau et l'imprimante (*connects*). Une relation peut donc posséder en attribut des *knowledgeComponents* de tout type : des machineries via *employing*, des ressources via *conveying* et des relations via *passingThrough*. Une relation est un type abstrait qui doit être spécialisé dans un de ses multiples sous-types concrets : échange (*exchange*), contrôle (*control*, utilisation (*use*), production (*produce*) et vérification (*check*).

— L'**échange** (*Exchange*) est défini comme une relation bi-directionnelle entre deux machineries.

```
ExchangeSrcTgtIsMachinery -> self
   .source.oclIsTypeOf(self.target
   .oclType(Machinery))
```

Dans l'exemple, le poste de travail échange des données avec le réseau LAN.

— Le **contrôle** (*Control*) est défini entre une machinerie source qui influence le comportement d'une machinerie cible. Il représente un lien à travers lequel la machinerie peut diriger la machinerie cible et déclencher ses différentes actions. Cette relation peut être vue comme un accès en écriture de la source sur la cible. Cette relation expose l'aspect critique de la machinerie source, dans le sens où une corruption de la source entraîne une corruption de la cible par la même occasion. Dans l'exemple, le technicien contrôle son poste de travail.

— L'**utilisation** (*Use*) représente le fait qu'une machinerie source ait besoin d'un *knowledgeComponent* cible pour fonctionner. Cette relation expose la dépendance de la source vis-à-vis de la cible, dans le sens où le blocage de l'accès à la cible perturbe le comportement de la source. Dans l'exemple, le poste de travail utilise une alimentation électrique pour fonctionner.

— La **production** (*Produce*) est la relation opposée de l'utilisation, la source machinerie produit le *knowledgeComponent* cible. Cette relation expose la dépendance de la cible sur la source pour exister. De ce fait, mettre hors service la source affecte la disponibilité de la cible.

— La **vérification** (*Check*), définie entre une machinerie source et un *knowledgeComponent* cible, représente une relation "sensorielle". La source observe la cible et possède donc une connaissance partielle de l'état de la cible, ce qui peut être vu comme un accès en lecture de la source sur la cible. Une attaque contre la cible pourrait donc être détectée par la source. Par exemple, si un capteur mesure un certain niveau d'eau d'un réservoir, toute modification du niveau d'eau est détectée par le capteur. Cette relation peut être spécialisée encore davantage :

  — La **maintenance** (*Maintain*) est une relation de vérification entre un exécutant source et un *knowledgeComponent* qui définit que la source humaine est responsable du bon fonctionnement de la cible. L'exécutant peut déclencher certaines actions sur la cible s'il s'agit d'une machinerie. Cette relation peut être vue comme un accès conditionnel en écriture de la source sur la cible en plus d'un accès en lecture. Cette relation est cruciale pour un APT car elle modélise la résilience de la cible tant que la source est opérationnelle. Si la cible n'est plus opérationnelle, l'exécutant est capable de la rétablir.

  — La **validation** (*Match*) est une relation de vérification entre un point de contrôle source et son passeport cible correspondant. Elle capture le fait que la ressource passeport soit nécessaire pour franchir le point de contrôle. Cette relation est particulièrement importante pour un APT car les points de contrôles constituent des obstacles pour l'attaquant jusqu'à ce que les passeports correspondants soient obtenus.

2.3.3.4 Application

En considérant ces concepts, plaçons nous dans la phase de **reconnaissance et armement** d'une APT désirant s'attaquer au système d'impression de documents de la Figure 2.6. D'après des opérations de renseignement antérieures, l'APT sait caractériser l'**environnement opérationnel courant**. Le système est composé d'un technicien modélisé par un **exécutant**, d'un poste de travail modélisé par une **interface**, d'un réseau LAN modélisé par un **réseau** et d'une imprimante modélisée par un **point de sortie**. Ces **machineries** sont reliées entre elles par des relations d'**échange** entre l'imprimante et le réseau et entre le réseau et le poste de travail. De plus, il existe un lien d'**utilisation** entre le technicien et son poste de travail. Enfin le lien conceptuel d'impression est représentée par une relation d'**utilisation** passant à travers toutes les autres relations et utilisant le poste de travail et le réseau LAN.

À partir de ce point de vue, supposons que l'objectif stratégique de l'APT soit d'empêcher l'impression du document par l'imprimante. L'environnement opérationnel souhaité, représenté en Figure 2.7, est donc privé de la relation d'impression. Chaque élément du système constitue une cible potentielle car la relation d'impression fait intervenir tous les éléments du système. Par exemple, si l'APT parvient à corrompre le technicien, l'impression n'est plus assurée. C'est également le cas pour chacun des éléments du système.

Cependant, le technicien peut être difficile à corrompre. Il faut obtenir plus d'informations sur lui afin de pouvoir mieux construire une attaque de *spear phishing* contre lui. Les spécificités du réseau LAN ne sont peut être pas connues, idem pour le poste de travail ou l'imprimante. Dans ce cas, l'APT poursuit ses opérations de reconnaissance jusqu'à

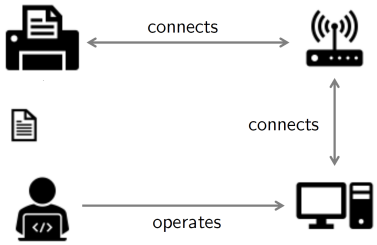


FIGURE 2.7 – Exemple fil rouge : état souhaité

obtenir suffisamment d'informations pour poursuivre. Dans le cas contraire, l'APT détermine sa stratégie. Par exemple, elle décide de cibler le technicien car elle a étudié son comportement en ligne via ingénierie sociale et peut donc fabriquer un mail frauduleux crédible. L'APT peut ensuite concrétiser sa stratégie en mettant en place les différents moyens qu'elle utilisera.

Le langage Pimca permet donc de faire de l'Operational Design dans un contexte de cyber-sécurité. Il permet de visualiser la phase de réflexion qui permet à l'APT de formuler son plan d'action. Cependant, à l'heure actuelle, Pimca n'est pas entièrement outillé pour faciliter ce processus. En effet, les processus itératifs de l'Operational Design demandent une réflexion itérative ce qui implique de pouvoir faire évoluer le modèle au fil de la réflexion.

Par exemple pour illustrer les trois exemples issus de Ginter [32] présentés au début de ce chapitre, Pimca atteint ses limites. En effet, chaque attaque révèle un raisonnement itératif sur l'architecture du système ciblé. La connaissance que l'attaquant a du système évolue avec le temps. Dans le cas de l'*insider*, il serait intéressant de modéliser le système du point de vue de l'attaquant. En première phase, sa connaissance se limite au réseau IT, puis elle progresse à une partie du réseau ICS. Enfin le réseau ICS est exploré jusqu'à ce que l'attaquant obtienne suffisamment d'informations pour causer des dommages. Ceci requiert de construire un modèle de système dynamique qui évoluerait donc à mesure que l'attaquant progresse. Dans le cas du *ransomware*, le mode opératoire n'est pas transposable à celui d'une APT. Il ne sera donc pas traité par le langage Pimca. Il est néanmoins possible de représenter le scénario de propagation du ransomware du réseau IT au réseau ICS à travers Pimca. Enfin pour le cas de Stuxnet [59], il est clair que les concepteurs de l'attaque ont utilisé une méthodologie similaire à l'Operational Design. En effet, la cible principale était le site industriel de raffinement d'uranium de Natanz. Pourtant l'APT a choisi une cible différente pour pénétrer dans le système. Le fournisseur de service externe au site industriel a été déterminé comme cible première spécifiquement aux termes d'une réflexion opérationnelle. Ceci a permis d'enclencher une seconde phase de reconnaissance sur le système ciblé. Ces nouvelles informations ont déclenché une nouvelle phase de réflexion opérationnelle. L'APT a ainsi pu mettre au point un malware complexe qui évite la détection des systèmes de sécurité et parvient à ses objectifs de dysfonctionnement. La mise au point de cette attaque a demandé une réflexion itérative et opérationnelle, elle semble difficile à représenter dans le langage Pimca à l'heure actuelle.

Pour traiter ces problématiques d'évolution opérationnelles temporelles, un aspect dynamique à Pimca pourrait être intégrer au langage. De plus si Pimca est un support pour l'Operational Design, il ne prodigue pas d'outil d'aide à la décision pour orienter la réflexion du concepteur.

2.4 Ingénierie Dirigée par les Modèles

Afin de raisonner sur nos systèmes et de fournir un support à l'élaboration des stratégies, il est indispensable de proposer une abstraction des systèmes ciblés et un outillage associé. Les concepteurs doivent en effet être aidés dans leur réflexion pour favoriser l'aide à la décision des stratégies.

L'approche ingénierie dirigée par les modèles (IDM) [16] fournit cette double possibilité de proposer la définition d'abstractions spécifiques à un domaine et d'obtenir un outillage ad'hoc pour les langages de modélisation spécifiés.

2.4.1 Principe

L'ingénierie dirigée par les modèles (IDM), ou *Model Driven Engineering* (MDE) en anglais est une sous-branche de l'ingénierie logicielle. Elle se pose en successeur du paradigme de la programmation orientée objet (POO) [90] dans le domaine logiciel pour augmenter le niveau d'abstraction. Mais l'IDM est aussi une formidable opportunité pour créer un lien entre le logiciel et le niveau système. La POO suit le principe du "tout est objet", c'est-à-dire qu'elle repose sur le concept de classe. Une classe représente une catégorie d'objets possédant des caractéristiques communes (méthodes et attributs). De ce concept découlent les deux relations fondamentales de la POO : l'instanciation et l'héritage. L'instanciation est la relation qui lie un objet à sa classe. L'héritage est la relation qui lie une classe fille (ou sous-classe) à une classe mère (ou super-classe). La classe fille "hérite" de toutes les caractéristiques de la classe mère, elle permet de spécifier de nouvelles caractéristiques en conservant celles de la classe mère.

En IDM, le concept principal est la notion de modèle : "Un modèle est une abstraction d'un système, modélisé sous la forme d'un ensemble de faits construits dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé" [16]. L'IDM permet de résoudre le problème de la complexité grandissante des systèmes d'information. Le recours à une modélisation du système est une simplification de la réalité pour répondre à une préoccupation précise. Le modèle est décrit dans un langage de modélisation spécifique, ou Domain Specific Modeling Language (DSML), conçu spécifiquement pour répondre au besoin. Par exemple, le langage Pimca est un langage de modélisation de systèmes pour conduire des analyses de sécurité. Ce faisant, il capture les caractéristiques pertinentes du système pour ce type d'analyse en suivant le principe de substituabilité [71]. Le principe de substituabilité stipule qu'un modèle "doit être suffisant et nécessaire pour permettre de répondre à certaines questions en lieu et place du système qu'il est censé représenter, exactement de la même façon que le système aurait répondu lui-même" [16]. Au regard d'une problématique précise, le modèle répond de la même manière que le système : on dit que le modèle est une représentation du système. Le modèle fait ensuite l'objet d'analyses partiellement voire entièrement automatisées comme la validation, la vérification ou l'exécution pour répondre au besoin.

Le langage de modélisation est lui-même défini par ce qu'on appelle un *méta-modèle* [78]. Le méta-modèle est, comme son nom l'indique, un modèle du DSML. La méta-modélisation est l'activité qui consiste à modéliser le langage qui sera utilisé pour répondre à un besoin précis. Le langage est défini par deux concepts : la syntaxe et la sémantique.

La syntaxe décrit la forme du langage. Elle spécifie la manière d'écrire ou de représenter le DSML en termes de symboles, mots-clés et formes. La syntaxe décrit les différentes règles de construction du langage. En informatique, on distingue généralement deux aspects différents de syntaxe : la syntaxe concrète et la syntaxe abstraite. La syntaxe concrète est le langage manipulé par l'utilisateur, elle peut être textuelle ou graphique. La syntaxe abstraite est la représentation interne manipulée par la machine, elle prend la forme d'un

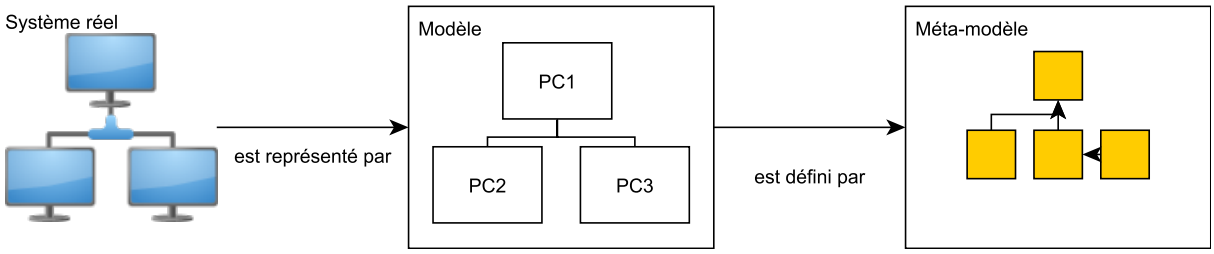


FIGURE 2.8 – Modélisation informatique

arbre abstrait. Ces deux aspects sont liés par une relation d'abstraction définie dans le langage.

La sémantique stipule la signification du langage. En d'autres termes, elle donne un sens aux concepts définis par la syntaxe. La sémantique permet de traduire un système en modèle de syntaxe concrète. Le domaine sémantique représente l'ensemble des états possibles du système dans le DSML.

La méta-modélisation revient donc à décrire la syntaxe abstraite, la syntaxe concrète et la sémantique du langage. La Figure 2.8 illustre les liens entre le système réel, le modèle et le méta-modèle. Plus particulièrement en ingénierie dirigée par les modèles, la syntaxe abstraite prend un rôle prépondérant. Pour répondre à des questions d'un domaine spécifique, l'ingénieur conçoit tout d'abord une syntaxe abstraite. Il fait ensuite le lien entre le domaine sémantique et les concepts de la syntaxe abstraite pour définir la sémantique du langage. C'est ce qu'on appelle le *mapping*. Le concepteur peut ensuite définir une ou plusieurs syntaxes concrètes, c'est-à-dire qu'un même modèle abstrait peut avoir plusieurs représentations concrètes (textuelle et graphique par exemple). Cette opération permet de faire le lien entre le langage et l'utilisateur.

Le principe de l'ingénierie dirigée par les modèles est de résoudre les problématiques sur le système en concevant un DSML pour chaque préoccupation. Le système est représenté par des modèles sur lesquels peuvent s'appliquer des analyses de vérification et de validation formelles [82]. C'est dans ce contexte que le langage Pimca a été conçu [92]. Il s'attache à identifier la surface d'attaque du système modélisé et permet de conduire d'autres analyses de sécurité. Usuellement en IDM, les analyses de modèles sont conduites à la main par le concepteur. L'analyse du concepteur est conduite sur l'espace sémantique du langage. Cependant, dans un environnement où le fonctionnement du système est amené à évoluer avec le temps et à cause des interactions de l'attaquant, il devient difficile de conduire les analyses à la main. Aussi il devient nécessaire de prodiguer au concepteur des techniques d'analyse assistée par la machine. Cependant, l'état de l'art montre que ce langage n'est pas suffisant pour représenter le raisonnement des menaces persistantes avancées sur un système ciblé.

Il se pose en effet plusieurs problématiques avec en premier lieu, comment modéliser les objectifs des APTs pour pouvoir les prendre en compte dans notre analyse ? Comment également enrichir le langage Pimca pour décrire l'évolution dynamique des entités Pimca ? Comment modéliser et capitaliser les modèles du système pour prendre en compte la découverte progressive du système ? Et enfin, comment modéliser les relations entre ces différents modèles pour assurer une cohérence durant tout le processus d'établissement de la stratégie de l'APT ?

Ces différentes questions soulèvent que l'expression de ces différents modèles requière une modélisation hétérogène par préoccupation et de manière sous-jacente introduit le problème de l'interopérabilité entre les modèles, que nous analysons dans le section suivante.

2.4.2 Interopérabilité

Dans le cadre de cette thèse, il est nécessaire de manipuler des modèles de systèmes décrits dans le langage préexistant Pimca [92]. Étant donné que le langage Pimca est spécifié par la DGA selon un cahier des charges spécifique, nos travaux ne peuvent pas modifier le méta-modèle Pimca en lui-même. Cependant, notre étude bibliographique montre que le langage Pimca n'a pas vocation à être un langage dynamique, ce qui est pourtant une nécessité pour modéliser le mode de fonctionnement des menaces persistantes avancées par la méthodologie d'Operational Design. Il est donc nécessaire de proposer un nouveau langage dynamique pour modéliser les aspects dynamiques du système. De plus, il faut également adapter la méthodologie d'Operational Design pour les besoins des menaces persistantes avancées. Bien qu'il soit conçu pour le langage Pimca, il est crucial de ne pas simplement étendre le langage Pimca par des concepts dynamiques. En effet, ceci limiterait la réutilisabilité de l'approche de l'Operational Design conjointement avec d'autres approches de modélisation de systèmes cyber. Aussi, il est clair que le problème de l'interopérabilité se pose, à minima, entre les deux aspects de modélisation statique et dynamique. Nous pouvons également étendre cette problématique aux différents modèles qui seront mis en jeu dans notre approche et décrits dans le chapitre 3.

L'interopérabilité est une problématique identifiée par Wegner [109] en 1996. Elle décrit la capacité de plusieurs composants logiciels à fonctionner ensemble malgré les différences de langage, d'interface et de plate-forme d'exécution. L'Association Francophone des Utilisateurs de Logiciels Libres (AFUL) [42] donne la définition suivante :

Définition 2.4.1 (Interopérabilité)**.** L'interopérabilité est la capacité que possède un produit ou un système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre [42].

Résoudre cette problématique est complexe car celle-ci possède plusieurs niveaux [21] :

— L'interopérabilité politique porte sur le partage de visions, orientations et stratégies globales des différentes parties prenantes.

— L'interopérabilité juridique repose sur un cadre légal commun entre les différents systèmes.

— L'interopérabilité organisationnelle porte sur les organisations et les processus mis en œuvre pour favoriser les échanges entre les différents systèmes.

— L'interopérabilité sémantique porte sur la signification des mots pour chacun des composants logiciels. En effet, certains termes en commun peuvent être utilisés et compris de manière différentes par les composants. L'interopérabilité sémantique fixe le sens des données échangées pour qu'il n'y ait qu'une interprétation possible.

— L'interopérabilité syntaxique assure que le format des données échangées entre les différents composants est le même. Elle assure que la communication entre les composants suive un format commun bien défini.

— L'interopérabilité technique assure que les interfaces des données échangées entre les différents composants sont les mêmes. Elle rend possible l'échange des informations définies au niveau sémantique selon le format défini au niveau syntaxique entre les composants.

Dans le cadre de la modélisation de la stratégie des menaces persistantes avancées, seuls les niveaux sémantique, syntaxique et technique sont pertinents. On peut décrire plus précisément ces trois aspects pour l'ensemble de nos travaux de la manière suivante :

— Le niveau sémantique consiste à résoudre le problème de "savoir se comprendre". Notre approche doit mettre en commun les différents concepts de la modélisation de systèmes cyber pour la sécurité et le vocabulaire du domaine de la planification militaire. Les termes choisis pour le standard doivent être définis sans ambiguïté afin de permettre l'échange entre ces deux domaines.

— Le niveau syntaxique consiste à résoudre le problème de "savoir communiquer". Notre approche doit définir un ensemble de concepts standard issus de la modélisation de systèmes pour la sécurité. Ces concepts utilisés dans notre nouveau langage permettent de mettre en place un format commun c'est-à-dire l'interopérabilité syntaxique.

— Le niveau technique consiste à résoudre le problème de "pouvoir communiquer". Notre approche doit prodiguer les interfaces nécessaires à l'interopérabilité avec d'autres langages de modélisation de systèmes pour la sécurité.

Les enjeux de l'interopérabilité sont nombreux, notamment en ingénierie dirigée par les modèles [5]. En effet, il est possible de résoudre des problèmes différents sur un système avec plusieurs DSML. Cependant, les différents modèles représentent le même système et ils sont donc interdépendants. Si le système évolue, chaque modèle doit évoluer conjointement. Les analyses de modèle peuvent induire des changements dans la configuration du système. On peut donc dire que les différents modèles peuvent induire des changements dans les autres modèles. De plus, si le méta-modèle d'un DSML évolue, cela peut engendrer des incohérences avec les autres langages utilisés pour d'autres diagnostics. Autrement dit, la question de l'interopérabilité des différents DSML est cruciale en IDM. Dans le cadre de nos travaux, le problème se pose notamment au niveau de l'interopérabilité entre le langage Pimca et le langage dynamique que nous proposons pour analyser les stratégies des menaces persistantes avancées.

Il existe plusieurs approches pour résoudre le problème de l'interopérabilité en IDM. Elles sont généralement classées en trois groupes : l'intégration, l'unification et la fédération de différents formalismes [25, 43, 74, 85].

L'intégration est une approche qui consiste à créer un langage de modélisation global qui inclut l'union de tous les concepts de chaque DSML. C'est une approche relativement simple à mettre en place car elle est systématique. Toutefois, cette approche est applicable lorsque le nombre de DSML est relativement restreint et que les problématiques ne nécessitent pas de faire intervenir de nouveaux DSML dans le futur. En effet, l'ajout d'un nouveau langage requiert de redéfinir le langage global d'intégration.

L'unification est une approche qui consiste à identifier les concepts communs entre les différents formalismes des langages. Dans cette approche, le concepteur définit la correspondance entre les différents concepts communs de chaque langage à travers un DSML. On parle d'une approche par langage pivot. Cette stratégie est la plus utilisée pour concevoir et implémenter l'interopérabilité entre plusieurs DSML. L'unification offre plusieurs avantages. Si le nombre de concepts communs est restreint, le langage pivot est un moyen compact pour maintenir les liens entre les différents langages. Une fois que les correspondances sont définies, la transformation de modèles d'un langage à un autre est directe entre les DSML et le langage pivot ou dans le sens inverse. Toutefois, la définition précise d'un langage pivot est une opération difficile pour plusieurs raisons. D'une part si la définition de la correspondance est trop abstraite, les concepts endémiques des différents langages sont perdus lors de la transformation en modèle pivot. D'autre part si la définition de la correspondance se fait à travers un langage pivot très riche pour conserver un maximum de propriétés des différents DSML, cela revient à adopter une approche d'intégration. Cette approche est donc rigide et lourde. L'approche d'unification prodigue donc une meilleure

flexibilité que l'intégration de manière générale, mais au prix d'un compromis entre une définition trop abstraite ou trop précise du langage pivot.

La fédération est une approche qui consiste à modéliser la sémantique des liens entre les concepts des différents DSML [24, 43, 74, 88]. Ces liens définissent ensuite la correspondance entre les différents concepts des langages sans l'usage d'un langage pivot. L'objectif principal de la fédération est de spécifier l'interopérabilité sans modifier les concepts des DSML et en s'axant plutôt sur les définitions sémantiques des relations entre les concepts. Ce faisant, les transformations ne se font plus de modèle à modèle mais simplement de concept à concept lorsque c'est nécessaire. Les approches de fédérations ont l'avantage de marquer une séparation entre les éléments sources issus des DSML et la modélisation de la sémantique de fédération. Ceci permet notamment d'assigner plusieurs sémantiques différentes à un même concept de DSML ou bien une seule sémantique à plusieurs concepts de DSML différents.

L'état de l'art montre que le domaine de la recherche sur les menaces persistantes avancées connaît une expansion graduellement croissante. Du point de vue de l'ingénierie dirigée par les modèles, il semble donc raisonnable de prévoir la création de nombreux DSML pour résoudre différents aspects de la problématique des APT. Afin de proposer une approche pérenne et interopérable avec ces futurs DSML, nous adoptons une approche de fédération dans la suite de nos travaux.

2.4.3 Analyse basée sur les modèles

En ingénierie dirigée par les modèles, il existe différentes techniques pour analyser et évaluer les systèmes modélisés [9]. On parle de techniques de Vérification et de Validation (V&V) définies par la norme ISO 9001 [49] concernant les principes de management de la qualité.

Définition 2.4.2 (Vérification)**.** Confirmation par des preuves tangibles que les exigences spécifiées ont été satisfaites [49].

Définition 2.4.3 (Validation)**.** Confirmation par des preuves tangibles que les exigences pour une utilisation spécifique ou une application prévue ont été satisfaites [49].

Ces techniques sont utilisées tout au long du cycle de développement d'un système ou d'un programme afin d'éviter une faute qui peut provoquer des erreurs engendrant des défaillances [60]. Concrètement, dans notre contexte, les techniques de V&V peuvent aider au diagnostic de la stratégie de l'attaquant une fois l'ensemble des modèles réalisés. Des techniques comme le **test logiciel** permettent de vérifier que pour un ensemble donné de cas de test, les modèles et donc le système modélisé se comportent selon les résultats attendus [111]. Cependant, ces techniques ne conviennent pas à notre contexte, où l'attaquant recherche des stratégies viables pour remplir sa mission. L'APT ne recherche pas à vérifier des résultats attendus.

Des techniques appelées **méthodes formelles** peuvent être utilisées pour la vérification. Ces techniques sont dites *formelles* car elles se basent sur des fondements mathématiques pour fournir leurs analyses. Principalement, on peut retenir deux grandes catégories de techniques de vérification formelle : la démonstration de théorèmes (*theorem proving*) [17] et la vérification de modèles [15].

Le *theorem proving* est une méthode formelle déductive qui consiste à vérifier des propriétés symboliques (théorèmes) via un système d'inférence. Ce processus peut être automatique, semi-automatique ou assisté afin d'obtenir une approche allant des hypothèses (spécifications) à la preuve (code source). Plusieurs environnements logiciels et cas industriels sont répertoriés relatifs à cette technique formelle comme la conception de systèmes critiques sûrs.

La vérification de modèles consiste à construire un modèle mathématique du système afin de pouvoir en vérifier les propriétés. Elles permettent d'avoir une garantie sur le résultat du diagnostic à condition que les modèles soient représentatifs du système [9]. On distingue généralement trois méthodes basées sur les modèles : le *model checking*, l'interprétation abstraite et le *monitoring*.

Le *model checking* consiste en la vérification d'un modèle d'automate fini pour qu'il soit conforme à une spécification. Il est notamment utilisé dans la vérification de systèmes logiciels critiques, par exemple en aéronautique [96]. Il étudie notamment le comportement d'un système au cours du temps et en fonction des perturbations.

L'interprétation abstraite consiste à utiliser une description abstraite d'un programme avec des concepts abstraits afin que le résultat de l'exécution abstraite aide au diagnostic de l'exécution concrète [19].

Le *monitoring* consiste à surveiller le comportement d'un programme lors de son exécution et à déclencher des contre-mesures en cas de comportement non désiré [56].

De toutes ces techniques, il semble préférable de s'orienter vers les techniques de vérification formelle basées sur les modèles afin de capitaliser sur tout un ensemble d'approches de modélisation. De plus, l'aspect formel fournit une garantie sur les résultats obtenus.

Le *model checking* en particulier étudie le comportement du système de manière exhaustive. Pendant l'analyse, on calcule une configuration du système qui capture l'état courant du système. L'état du système évolue dans le temps à l'aide d'une fonction de transition qui permet de calculer la ou les configurations suivantes du système. Une ou plusieurs propriétés exprimées en logique temporelle sont vérifiées dans l'ensemble des configurations possibles du système. Ces propriétés expriment des exigences critiques que le concepteur souhaite assigner au système. Classiquement le *model checking* est utilisé pour vérifier des propriétés de sûreté de fonctionnement, cependant on peut également utiliser cette technique pour vérifier des propriétés de sécurité [45].

Grâce à l'analyse exhaustive des configurations du système, le *model checking* garantit que la propriété est vérifiée dans tous les scénarios possibles ou elle expose un scénario dans lequel la propriété est violée. Une propriété de sécurité violée indique l'existence une faille de sécurité. Cette faille de sécurité pourrait être exploitée par l'attaquant. Donc du point de vue de l'attaquant, une propriété de sécurité violée indique qu'une attaque est possible [45]. Aussi il semble pertinent de retenir le *model checking* dans la recherche de scénarios d'attaques possibles.

Toutefois, la vérification formelle comporte plusieurs inconvénients en fonction de la technique utilisée. Dans le cas du *model checking*, du fait de l'analyse exhaustive de tous les scénarios possibles, la technique se heurte au problème de l'explosion combinatoire. Un modèle de système avec un nombre réduit d'entités peut engendrer un espace d'état très vaste, s'il existe un grand nombre d'interactions entre ces entités. Dès lors, chaque ajout dans le modèle peut entraîner une augmentation exponentielle de la taille de l'espace d'état. Si l'espace d'état est trop vaste, la technique n'est plus applicable car l'exploration de tous les scénarios prend trop de temps, ou pire, elle ne peut pas être menée à terme car la taille mémoire du calculateur est atteinte.

Cette technique sur les systèmes complexes comme les systèmes industriels reste donc difficilement applicable telle quelle. Diverses approches permettent de repousser l'explosion combinatoire comme l'optimisation du calcul ou du stockage des configurations lors de la construction de l'espace d'état [97]. Une autre approche possible consiste aussi à lever la contrainte de l'exhaustivité du parcours de l'espace d'état en guidant ce parcours selon différentes contraintes ou heuristiques.

Mais aussi d'un point de vue méthodologique, nous pouvons utiliser une conception de nos modèles comme le fait de contraindre le système par un comportement déterministe

de l'environnement ou le fait de procéder à des abstractions sur le système [61]. Mais dans ce dernier cas, la question du fossé sémantique entre le modèle et le système réel se pose [9] : le modèle est-il suffisamment fidèle au système réel pour que les résultats de *model checking* soient pertinents ?

Cependant, obtenir des scénarios d'attaque sur un système reste très intéressant dans les premières phases d'analyse de vulnérabilités ou de risques sur un système. Étant donné les prérogatives de l'APT, la recherche automatique de scénarios d'attaque est une problématique primordiale pour la suite de nos travaux.

2.4.4 Synthèse et besoins

Dans notre contexte, il semble capital d'identifier l'ensemble des besoins requis pour pouvoir modéliser l'élaboration de la stratégie des menaces persistantes avancées sur les systèmes qu'ils visent. On peut distinguer quatre types de besoins qu'il faudra remplir : la modélisation du système, la modélisation de la stratégie, l'interopérabilité et l'analyse des modèles. La Table 2.2 résume les différents besoins de nos travaux ainsi que leur avancement dans la littérature et elle est décrite dans les sections suivantes.

2.4.4.1 Modéliser le système ciblé

La modélisation du système pour l'élaboration de stratégie des menaces persistantes avancées consiste en la caractérisation du système d'un point de vue structurel et comportemental. Le point de vue structurel peut être modélisé par le langage Pimca proposé par la DGA [92]. Ce langage est d'ores et déjà pourvu d'une syntaxe abstraite sous la forme d'un diagramme de classe explicité dans le Chapitre 2.1. Sa sémantique est également déjà explicitée.

Néanmoins le langage Pimca est dépourvu de syntaxe concrète à l'heure actuelle, c'est-à-dire qu'il n'existe pas de moyen concret de manipuler les concepts du langage pour l'utilisateur. Une des exigences de ces travaux de thèse est donc de produire une syntaxe concrète pour le langage Pimca.

De plus, le langage Pimca ne peut modéliser le système que d'un point de vue statique. Il est donc nécessaire de proposer un DSML supplémentaire pour modéliser le système d'un point de vue comportemental. Ce langage sera utilisé conjointement avec Pimca pour modéliser le système dans son entièreté pour les analyses de sécurité. Ceci nécessite de créer un DSML complet. Du point de vue de l'ingénierie dirigée par les modèles, il s'agit de d'écrire la syntaxe abstraite du langage ainsi que sa sémantique. Il faudra également proposer une syntaxe concrète pour le langage dynamique afin qu'il puisse être manipulé par un utilisateur. La Table 2.2 indique cet avancement dans la littérature avec une méthodologie, une syntaxe abstraite et une sémantique présentes mais incomplètes. La syntaxe concrète qui correspond à nos besoins reste entièrement à réaliser.

2.4.4.2 Modéliser la stratégie

La modélisation de la stratégie des menaces persistantes avancées consiste en l'adaptation de la méthodologie militaire de l'Operational Design. Notre étude bibliographique montre que cette méthodologie pourrait fournir des résultats pertinents dans un nouveau contexte. Toutefois, les concepts du domaine militaire et ceux du domaine des menaces persistantes avancées ne coïncident pas parfaitement. Une des exigences des travaux de cette thèse est donc d'adapter cette méthodologie. La Table 2.2 indique que la méthodologie d'Operational Design semble utilisable mais elle reste à transposer à notre contexte.

		État de l'art
Modèle de systèmes	Méthodologie	~
	Syntaxe abstraite	~
	Syntaxe concrète	X
	Sémantique	~
Modèle de stratégie	Méthodologie	~
	Syntaxe abstraite	X
	Syntaxe concrète	X
	Sémantique	X
Interopérabilité	Approche	✓
	Outillage	X
Analyse	Approche	✓
	Outillage	~

Table 2.2 – Synthèse et besoins de l'approche

En d'autres termes, du point de vue de l'ingénierie dirigée par les modèles, il est nécessaire de créer un DSML complet également. Le DSML de stratégie doit avoir une syntaxe abstraite ainsi qu'une sémantique bien définie. Il faudra également proposer une syntaxe concrète pour le langage dynamique afin qu'il puisse être manipulé par un utilisateur. La Table 2.2 montre donc que ces trois éléments manquent dans la littérature.

2.4.4.3 Gérer l'interopérabilité

Les premières exigences introduisent naturellement le problème de l'interopérabilité entre les différents DSML que nous utiliserons. Comme expliqué précédemment, ce problème porte sur les aspects sémantique, syntaxique et technique de l'interopérabilité.

Pour pallier ces problèmes et permettre d'associer ces travaux à d'autres DSML dans le futur, nous faisons le choix de la fédération de modèles. Dans notre domaine en pleine expansion, l'approche par fédération semble la mieux indiquée. Il reste à déterminer puis à mettre en place l'outillage qui réalisera la fédération de modèles, ce qui constitue une nouvelle exigence pour ces travaux. La Table 2.2 indique que l'approche par fédération a déjà fait ses preuves dans la littérature. Il reste néanmoins à l'appliquer concrètement dans notre domaine.

2.4.4.4 Analyser les modèles

Enfin l'approche globale devra produire un diagnostic sur le système sous la forme d'une stratégie ou approche opérationnelle qu'adopterait une menace persistante avancée pour s'attaquer au système. Pour ce faire, il faut que l'ensemble de l'approche soit exécutable et le plus automatisable possible. Aussi le choix des techniques d'analyse formelle basées sur les modèles serait souhaitable pour tendre vers une sécurité garantie. En particulier, le *model checking* semble prometteur, dans la mesure où il permet d'exhiber des scénarios de défaillance du système, ce qui équivaut à des scénarios d'attaque pour l'APT. Cependant cette technique se heurte à l'explosion combinatoire qu'il sera nécessaire de gérer. La Table 2.2 indique donc que l'approche du *model checking* est d'ors et déjà utilisée mais il reste toutefois à adapter ces techniques formelles à notre problématique de sécurité.

2.5 Conclusion

Le contexte des systèmes industriels introduit dans la Section 2.1 est un domaine en pleine expansion. De fait, il devient de plus en plus vital de se prémunir contre les menaces qui pèsent sur des systèmes de plus en plus complexes et de plus en plus sensibles. Notre étude bibliographique montre que les APT sont une menace pour la sécurité des systèmes industriels dans la Section 2.2. Leur caractéristiques et leur mode opératoire en font des adversaires contre lesquels il est difficile de défendre un système. Malgré les efforts entrepris pour adapter la cyber-sécurité à cette nouvelle problématique, il semble que la phase de **reconnaissance et armement** de la *kill chain* des APT ne soit pas complètement comprise.

En effet, les techniques de sécurité ne cherchent pas à défendre contre cette phase dans son intégralité. Des solutions partielles existent mais l'APT est considérée comme une boîte noire dont on ne cherche pas à comprendre le fonctionnement ou la méthodologie. Pour mieux comprendre comment les APT développent leurs plans d'actions, il est pertinent d'étudier les travaux militaires qui théorisent déjà l'élaboration de stratégies dans la Section 2.3. La méthodologie de l'**Operational Design** en particulier est adaptée à cette problématique puisque des efforts ont déjà été entrepris pour l'adapter aux problématiques cyber. Par exemple le langage Pimca permet de faire de l'**Operational Design** dans ce contexte.

Toutefois, le langage Pimca connaît des limites. Il demande au concepteur, c'est-à-dire le commandant opérationnel de l'APT, de tout faire "à la main". C'est pourquoi il est important d'enrichir Pimca en proposant un **outillage autour du langage Pimca** d'une part et une **extension dynamique** d'autre part. Cet outillage doit par ailleurs répondre à un **ensemble d'exigences concernant les technologies d'ingénierie dirigée par les modèles** mises en lumière dans la Section 2.4.

Compte tenu de l'ensemble des problématiques soulevées par l'état de l'art, les travaux de thèse s'attarderont précisément sur **la modélisation de la phase de reconnaissance et armement des APT en appliquant la méthodologie de l'Operational Design**. La suite de ce manuscrit tente de répondre à ces problématiques à travers nos contributions dans le Chapitre 3.

Chapitre 3

Définition d'une méthodologie de stratégie

Sommaire

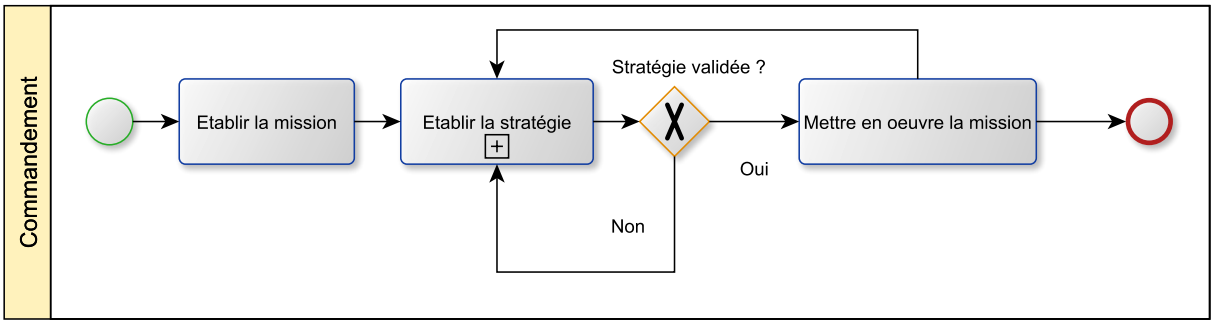


FIGURE 3.1 – Processus global d'attaque de l'APT

Le Chapitre 2 soulève différentes problématiques concernant les menaces persistantes avancées ou Advanced Persistent Threats(APT).

Dans ce chapitre, nous détaillons les différentes contributions de nos travaux pour répondre au problème posé par les APT. L'ensemble des travaux détaillés par la suite s'oriente sur la compréhension de la phase de reconnaissance et armement des APT, c'est-à-dire la phase de conception de stratégie. Nous répondons à cette problématique en utilisant l'ingénierie dirigée par les modèles (IDM) suivant différentes composantes : Définition de DSML, gestion de l'interopérabilité et analyse formelle des modèles produits.

Notre approche se base donc sur l'intégration de la méthodologie militaire de l'**Operational Design** dans le contexte de la cyber-sécurité. Pour ce faire, le langage Pimca représente la pierre angulaire pour construire une stratégie, en effet Pimca est un langage de modélisation système de haut niveau d'abstraction, conçu à des fins de cyber-sécurité.

Au cours de ce chapitre, nous employons la notation *Business Process Model and Notation* (BPMN) afin de modéliser les différents processus de notre méthodologie. La notation BPMN est standardisée par l'Object Management Group (OMG) [76] et permet de décrire les processus en termes de séquences de tâches à exécuter par différents participants. La Figure 3.1 représente, sous la forme d'un diagramme BPMN, le processus global dans lequel s'inscrit l'élaboration d'une attaque de l'APT. Dans ce processus, l'APT établit une mission dans un cadre d'opération et avec des objectifs bien précis.

Définition 3.0.1 (Mission)**.** Le but à atteindre sur un système cible spécifique comprenant un ou plusieurs objectifs et rencontrant possiblement plusieurs obstacles.

Ensuite, l'APT élabore sa stratégie d'attaque pour parvenir à ses objectifs sur le système cible. Lorsqu'elle considère que sa stratégie est réalisable, elle peut passer à la mise-en-œuvre de la mission. La mise en œuvre de la mission peut aboutir à l'atteinte des objectifs ou à une remise en cause de la stratégie selon la découverte du système et des opportunités découvertes ou estimées. Dans ce cas l'APT ré-établit sa stratégie pour prendre en compte les changements constatés.

Dans le cadre de nos travaux, nous nous concentrons spécifiquement sur le processus d'élaboration de la stratégie, autrement dit : "Comment l'APT établit-elle sa stratégie ?"

Dans un premier temps, la Section 3.1 détaille la méthodologie globale de notre approche. Ensuite, dans la Section 3.2 nous passons en revue l'outillage qui a été mis en place pour prendre en charge le langage Pimca. Le framework que nous avons construit permet la création et l'analyse de modèle Pimca. Nous abordons notamment l'extension dynamique du langage Pimca. Celle-ci a pour but de modéliser le comportement des éléments du système au cours du temps d'une part du point de vue d'un comportement nominal du système et d'autre part du point de vue de l'attaquant APT. Enfin la Section 3.3 précise le point de vue de l'APT sur le système. Cette section explique comment l'APT est modélisée

puis connectée au modèle de système Pimca. Elle détaille les interactions entre les deux modèles ainsi que les différentes analyses qu'il est possible de mener.

3.1 Méthodologie

L'approche globale de nos travaux s'articule autour de la question suivante : "Comment une menace persistante avancée procède-t-elle pour attaquer le système ?".

Pour répondre à cette question, nous adoptons le point de vue d'Operational Design. Notre méthodologie consiste donc à modéliser le processus de conception dans lequel l'APT construit son approche opérationnelle pour attaquer le système.

Dans cette section nous détaillons comment nous avons adapté la méthodologie d'Operational Design au contexte de la cyber-sécurité. Dans un soucis de clarification, nous présentons notre méthotologie en deux temps, une méthodologie générale qui se veut réutilisable dans différents contextes et une méthodologie détaillée qui s'appuie sur une chaine d'outils utilisable par les APT.

3.1.1 Méthodologie d'Operational Design

Nous adaptons la méthodologie de l'Operational Design présentée dans le chapitre précédent en Figure 2.4 au domaine de la cyber-sécurité. Le framework que nous construisons permet au concepteur d'opérations cyber de capturer les caractéristiques de la mission, l'environnement opérationnel courant, l'environnement opérationnel souhaité et les obstacles posés par le système qui empêchent d'atteindre cet environnement souhaité. Grâce aux outils d'analyses que nous proposons, le concepteur peut modéliser des opportunités d'interactions avec le système et vérifier formellement si elles lui permettent d'atteindre ses objectifs.

La Figure 3.2 reprend la structure de la Figure 2.4 de l'Operational Design et présente l'approche globale de la thèse pour adapter cette méthodologie. L'approche se développe à partir de la mission de l'APT. Elle permet de concrétiser les objectifs de la mission en une approche opérationnelle, c'est-à-dire une stratégie concrète mettant en jeu les différentes capacités de l'APT. L'approche détaille la mission de haut niveau selon les trois axes proposés par l'Operational Design :

1. Identifier la cible de l'APT. La cible définit le périmètre de la mission, c'est-à-dire l'environnement opérationnel courant.
2. Identifier les objectifs. Les objectifs de la mission caractérisent l'environnement opérationnel souhaité.
3. Identifier les obstacles. Les obstacles sont les éléments du système ciblé qui empêchent l'APT d'accomplir sa mission.

Pour modéliser la cible (à gauche dans la Figure 3.2), c'est-a-dire l'environnement opérationnel courant, notre méthodologie utilise le langage Pimca. En effet, Pimca permet de modéliser la structure du système ciblé. Ce processus revient à identifier les différents acteurs du système pour les modéliser en composants Pimca. Il est également nécessaire de comprendre les différentes relations qui lient les acteurs dans le système afin de modéliser les relations Pimca correspondantes. Le premier modèle Pimca résultant de ce processus capture la compréhension de l'APT sur la structure du système ciblé. Toutefois, cette structure n'offre qu'un point de vue statique sur le système.

La littérature montre que dans de nombreux cas, les instances d'APT exploitent plusieurs phases de vie du système. Par exemple, dans le cas de l'attaque Stuxnet [59], l'APT

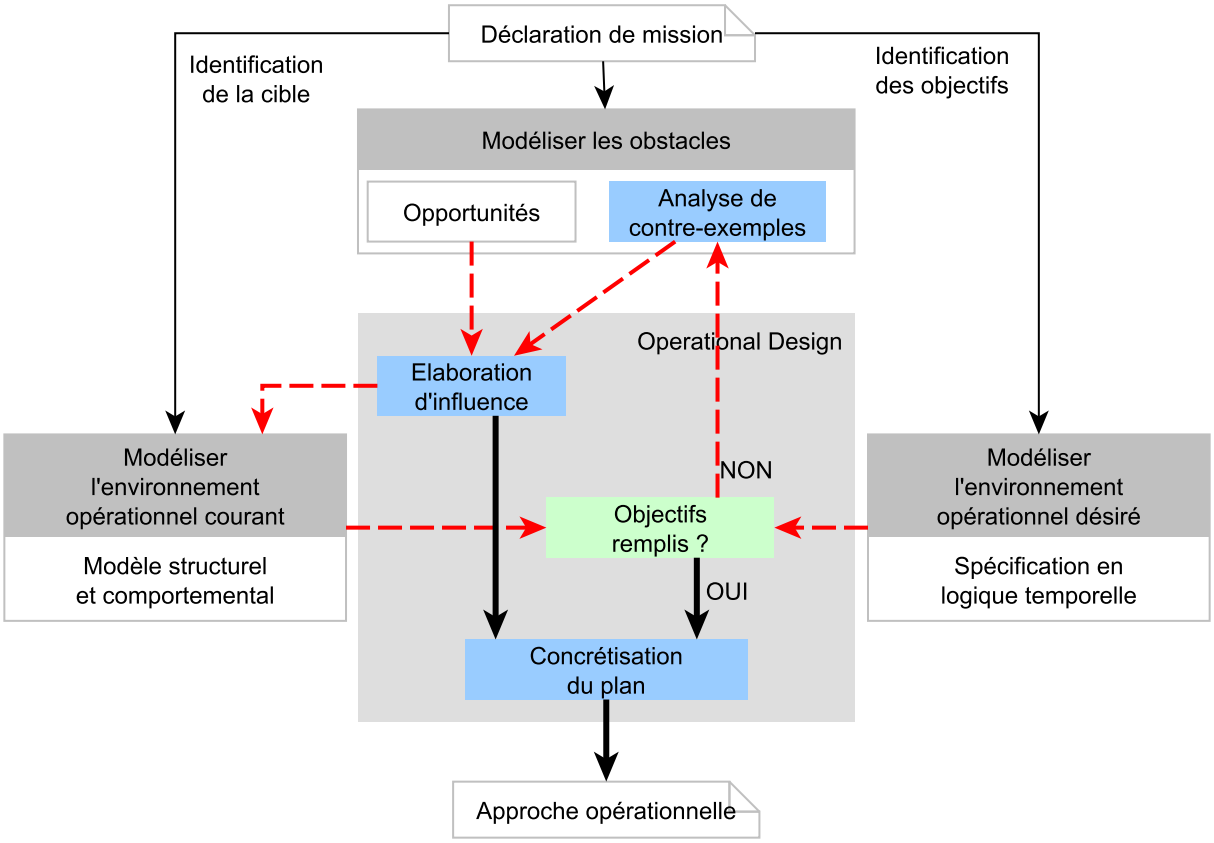


FIGURE 3.2 – Vue d'ensemble de l'approche

a supposément exploité la venue d'un fournisseur de service externe aux centrales de raffinement d'uranium pour accomplir sa mission. Il paraît donc nécessaire de pouvoir capturer le comportement du système d'un point de vue dynamique pour exposer des faiblesses liées à un comportement normal du système. Nous appelons cet aspect dynamique du système le **comportement nominal** du système. Pour capturer le **comportement nominal** du système, nous proposons une extension dynamique du langage Pimca. Les spécifications du langage Pimca précisent que les machineries sont des composants actifs. Ce sont des éléments pourvus de comportements et capables de modifier l'état du système. Cette extension attache aux machineries du modèle Pimca des comportements capturés dans un formalisme de commandes gardées. Ce formalisme sera détaillé dans la Section 3.3.

La modélisation de l'environnement courant construit un modèle Pimca capturant la structure et le comportement nominal de la cible. Ce faisant, le concepteur capture sa compréhension du système ciblé avec le langage Pimca. Notre framework, qui sera détaillé dans la Section 3.2, propose un aspect graphique à la conception de modèle Pimca. Ceci permet de faciliter la communication entre le concepteur et d'autres experts de la cybersécurité afin de perfectionner le modèle.

Pour modéliser les objectifs de la mission (à droite dans la Figure 3.2), c'est-à-dire l'environnement opérationnel souhaité, notre méthodologie s'appuie sur la modélisation système proposée lors de la modélisation de l'environnement opérationnel courant. Elle capture l'état dans lequel le système doit être pour valider la mission. Dans ce cadre de modélisation, l'APT cherche à décrire ou obtenir l'évolution temporelle du système qui va conduire à cet état du système recherché. Cette évolution temporelle va capturer tous les états du système, un ou plusieurs chemins dans un graphe d'états du système, qui vont conduire à l'état attendu. L'expression des objectifs sur le système cherche donc à exprimer

un avenir possible du système sur les états d'un ensemble ou sous-ensemble des entités du système modélisé avec le langage Pimca. Ce processus revient à spécifier les objectifs de la mission sous forme de propriétés de logique temporelle linéaire (LTL) à vérifier sur le modèle Pimca. À travers une modélisation adéquate du comportement nominal du système, le concepteur identifie les différentes entités impliquées dans les objectifs de la mission. Et ensuite, les objectifs conceptuels de la mission sont traduits en propriétés LTL en se basant sur ces mêmes entités et variables d'état de ces entités, ceci en se reposant sur les travaux de Abadi et Lamport [3]. Ces différents aspects du framework seront discutés en Section 3.3. De même que précédemment, le concepteur capture sa compréhension des objectifs de la mission vis-à-vis du système. Ces objectifs ainsi modélisés peuvent être présentés et discutés avec d'autres experts de la cyber-sécurité afin d'améliorer ce modèle.

Mis à part le cas très spécifique où le système est déjà dans son état d'environnement opérationnel désiré, le système ne vérifie pas les propriétés LTL spécifiées lors de la modélisation des objectifs de la mission. Un ou plusieurs obstacles empêchent l'accomplissement de la mission. Pour modéliser cet obstacle (en haut dans la Figure 3.2), le concepteur identifie les machineries du modèle Pimca dont le comportement entrave la mission. Dans notre framework, ce processus est guidé par le model-checking. Les propriétés LTL sont vérifiées par nos outils, qui seront détaillés en Section 3.3. Si une propriété est violée, notre framework retourne un contre-exemple de scénario d'évolution du système pouvant aider le concepteur à identifier les éléments du système qui posent problème de manière systématique. En se basant sur les connaissances, les ressources auxquelles l'APT a accès, les capacités de l'APT et l'expérience, le concepteur identifie alors des opportunités d'interaction avec des éléments problématiques qui peuvent lui permettre d'accomplir ses objectifs.

Les trois processus d'Operational Design sont itératifs et ils influent les uns sur les autres pour parvenir à une approche opérationnelle. Ce faisant, notre méthodologie s'appuie sur trois mécanismes qui apparaissent au centre de la Figure 3.2 :

1. La satisfaction d'objectifs représente l'étape de vérification formelle des objectifs dans le modèle de système courant grâce au model-checking. Ce mécanisme est automatisé dans notre framework. Si les objectifs ne sont pas satisfaits, notre outil produit un contre-exemple qui correspond à un scénario dans lequel le système résiste à l'attaque. Ce contre-exemple permet de guider le concepteur dans l'identification de l'obstacle. Si les objectifs sont satisfaits, le concepteur peut passer à l'étape de concrétisation du plan.

2. L'élaboration d'influence consiste à raffiner le modèle du système de l'environnement pour prendre en compte les différentes opportunités imaginées par le concepteur. Ces opportunités induisent des comportements différents pour les machineries du modèle Pimca ciblé par l'APT. Il est donc nécessaire d'altérer le modèle en conséquence. Cette altération utilise le formalisme de comportement que nous spécifions pour le modèle Pimca. Le concepteur peut ajouter, supprimer ou modifier des comportements de machineries dans ce même formalisme pour créer un nouveau modèle Pimca correspondant au système ciblé enrichi des opportunités d'interaction imaginées par le concepteur.

3. La concrétisation du plan est le mécanisme qui finalise l'approche opérationnelle. Une fois que les objectifs sont réalisables à partir de l'environnement courant, le concepteur analyse les différentes interactions mises en jeu pour satisfaire les objectifs. Ces interactions concrètes sont alors organisées en une stratégie, un plan d'action global qui constitue l'approche opérationnelle de l'APT.

La méthodologie que nous adoptons est directement adaptée de l'Operational Design pour le domaine de la cyber-sécurité. Les problématiques que nous traitons sont donc très

vastes. Bien que la thèse s'oriente tout particulièrement sur les menaces persistantes avancées, il nous semble judicieux de proposer deux aspects à notre méthodologie : une méthodologie générale et une méthodologie détaillée. La méthodologie générale présente le processus global indépendamment de la menace que nous souhaitons modéliser. Cela permet notamment sa réutilisation dans des contextes différents, avec d'autres types de menaces. La méthodologie détaillée, quant à elle, met en application notre méthodologie spécifiquement dans le contexte des APT. Elle s'appuie sur une chaîne d'outillage concrète et permet d'exécuter entièrement l'approche dans le cadre spécifique des menaces que nous avons traitées dans le Chapitre 4.

3.1.2 Méthodologie générale

La méthodologie que nous proposons se décompose en trois phases successives :

— Spécification

— Modélisation

— Analyse

Dans un premier temps, il est bon de préciser que la phase de spécification de la mission s'attarde à rassembler et à organiser l'information sur le système indépendamment de toute modélisation. Les sources d'information sont très diverses et peuvent provenir d'une phase d'ingénierie sociale (qui est hors du contexte de notre travail) mais nous pouvons établir que l'information vient principalement de sources documentaires. Mais ces différentes sources doivent être organisées et nous devons conserver les relations entre les objectifs généraux de la mission et ces sources documentaires. Pour spécifier la mission, nous allons donc chercher à créer et à conserver des relations entre les sources d'information et les concepts métiers nécessaires à la modélisation de la mission. La modélisation permettra ensuite de concrétiser les concepts métiers de la mission.

Dans un second temps durant cette phase de modélisation, le concepteur cherche à décrire le système cible selon les besoins de la mission. Ce faisant, il réalise un modèle du système selon ce point de vue particulier lié à la mission. Mais il modélise également les objectifs à atteindre et les opportunités disponibles dans l'environnement.

D'un point de vue abstrait, les trois processus que nous adaptons directement depuis l'Operational Design sont conduits de manière simultanée. La Figure 3.3 présente la méthodologie du point de vue du processus BPMN. Une fois que la mission est spécifiée, on retrouve les trois processus-clés de l'Operational Design que sont : "Modéliser l'environnement opérationnel désiré", "Modéliser l'environnement opérationnel courant" et "Modéliser les obstacles". Ces trois processus se conduisent en parallèle et interagissent entre eux à travers le modèle de système qu'ils manipulent. Il faut noter que le processus de modélisation de l'environnement opérationnel courant produit un modèle de système sur lequel se basent les deux autres processus. En effet, c'est sur ce modèle d'environnement que les réflexions en termes d'objectifs et d'obstacles sont menées. D'une itération à l'autre du modèle de système, les processus de modélisation d'environnement opérationnel désiré et d'obstacles peuvent exprimer un besoin de raffinement de l'environnement opérationnel courant. À terme, les trois processus fournissent trois modèles qui seront utilisés dans les analyses :

— le modèle de système,

— le modèle d'objectifs,

— le modèle d'obstacles.

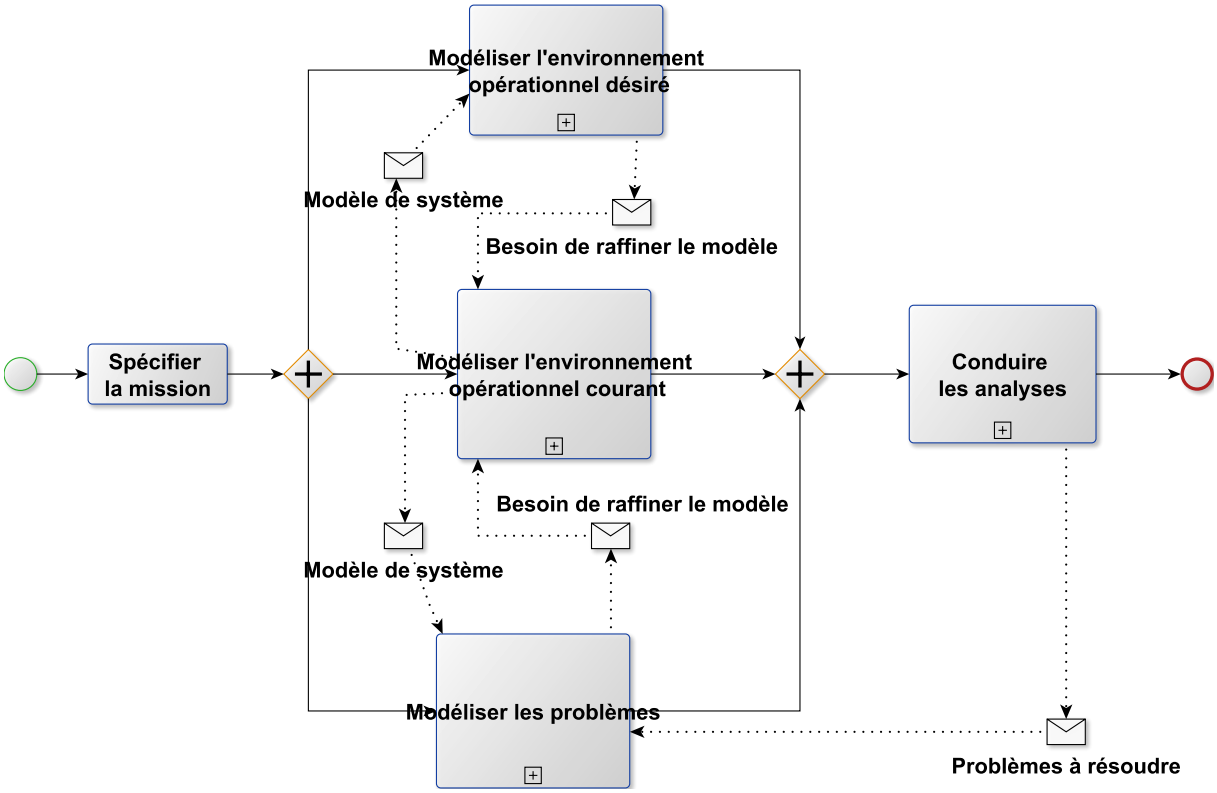


FIGURE 3.3 – Processus d'Operational Design de l'APT

Enfin on peut voir sur la Figure 3.3 un dernier processus d'analyse à l'issue duquel l'APT formule sa stratégie. L'objectif de l'analyse de nos modèles est de formuler une stratégie opérationnelle pour que l'APT parvienne à ses fins. Si l'analyse montre que la mission ne peut être accomplie dans les conditions proposées par les modèles, il est nécessaire de réitérer le processus de modélisation des obstacles. Ce faisant, les trois processus subissent une nouvelle itération. Nous détaillons dans la suite de cette partie les sous-processus internes à chacun de ces processus.

3.1.2.1 Spécification de la mission

La menace persistante avancée développe une stratégie ou approche opérationnelle afin de mener à bien sa mission. Dans le cadre de cette thèse, nous cherchons à comprendre ce processus en transposant la méthodologie d'Operational Design aux APT. Autrement dit, nous modélisons le processus de développement de stratégie par l'Operational Design utilisé par une APT pour accomplir une mission.

La mission se situe au niveau opérationnel militaire. Elle comprend une opération durant en moyenne un à plusieurs mois. La cible de la mission est une entité choisie par la menace persistante avancée. Cette cible est un système de complexité de l'ordre d'un site industriel ou d'un siège d'entreprise.

Les spécificités liées au domaine des APT font que le cadre de la mission met en jeu des aspects cyber contrairement aux missions de l'Operational Design classique. L'armement tel qu'il est envisagé par la méthodologie militaire n'est pas adapté à ces problématiques. Les aspects cyber impliquent l'usage de malware afin d'exploiter des vulnérabilités des systèmes pris pour cible ainsi que le recours à l'ingénierie sociale. Bien que les cibles d'APT soient variées et concernent de nombreux secteurs d'activités différents, les missions d'APT visent typiquement les biens vitaux ou critiques d'une organisation ou d'une entreprise.

FIGURE 3.4 – Structure de triple, élément de base d'une ontologie

Les objectifs des missions d'APT relèvent généralement de l'ex-filtration de données sensibles ou de modification, ou destruction, de données vitales au système. De plus, les APT se démarquent par leur mode opératoire furtif. Les missions d'APT sont des opérations qui usent de discrétion afin de parvenir aux objectifs.

Les besoins de la mission peuvent être exprimés de manière abstraite et partielles, tant que les informations permettent de déduire les détails nécessaires à la méthodologie de l'Operational Design. Cette étape du processus a lieu dans la première tâche présentée dans la Figure 3.3. Elle comprend une identification des différents besoins de la mission. Pour ce faire, le concepteur doit agréger un ensemble d'informations issues de sources variées, comme des documents de spécifications du système, ou de description sous différents points de vue, ou même des documents de type *publicitaires*. Il s'agit de traduire les intentions données à la mission et les connaissances sur le système cible en termes de concepts d'Operational Design. Comme présentés dans la Figure 3.2, il est crucial de pouvoir identifier clairement le système ciblé et les objectifs pour engager les trois processus suivants.

La spécification de la mission définit l'ensemble des concepts et des spécificités du système cible qui seront manipulés au cours de l'analyse. C'est pourquoi il est primordial de capturer précisément les différents éléments nécessaires au développement de la stratégie de la mission.

Ces éléments peuvent être classifiés selon cinq concepts inspirés de la méthodologie de l'Operational Design [12, 26, 41, 102, 105, 101] : l'environnement, les éléments constituants, les objectifs, les obstacles et les solutions.

Les données sur lesquelles se basent la méthodologie sont de plusieurs types différents et des relations complexes relient ces différentes données. Par exemple, un objectif peut faire référence à différents éléments du système cible. De même, un obstacle posé par un élément du système peut avoir plusieurs solutions techniques différentes.

Il est donc nécessaire de construire un modèle de données qui permet d'établir ces relations diverses. Notre méthodologie impose d'utiliser un modèle de données adapté au domaine et donc sémantiquement riche. De plus, il est également crucial d'optimiser les concepts manipulés afin de faciliter les processus de modélisation à venir dans la méthodologie.

Une ontologie [23, 28, 7] nous permet de représenter ces besoins. L'ontologie est un modèle de connaissances dédié à un domaine particulier. Elle permet donc de gérer des liens sémantiques riches entre les différents concepts du domaine. De plus, elle est conçue pour être extensible ce qui permet d'ajuster précisément le niveau d'abstraction des données modélisées à un ensemble nécessaire et suffisant.

Les relations entre les concepts sont stockées dans une ontologie sous la forme de triples sémantiques. Un triple est constitué de trois éléments : sujet, prédicat et objet. Il représente une assertion dans le modèle de données, de sorte que le sujet exerce le prédicat sur l'objet. Par exemple l'assertion "Alice connaît Bob" se décompose en triple avec comme sujet "Alice", comme prédicat "connaît" et comme objet "Bob". La Figure 3.4 représente la structure d'un triple dans une ontologie. Le sujet et l'objet sont des instances de concepts dans l'ontologie alors que le prédicat est une relation entre ces deux instances. L'ontologie est un ensemble de concepts, d'instances de ces concepts et de triples représentant leurs relations.

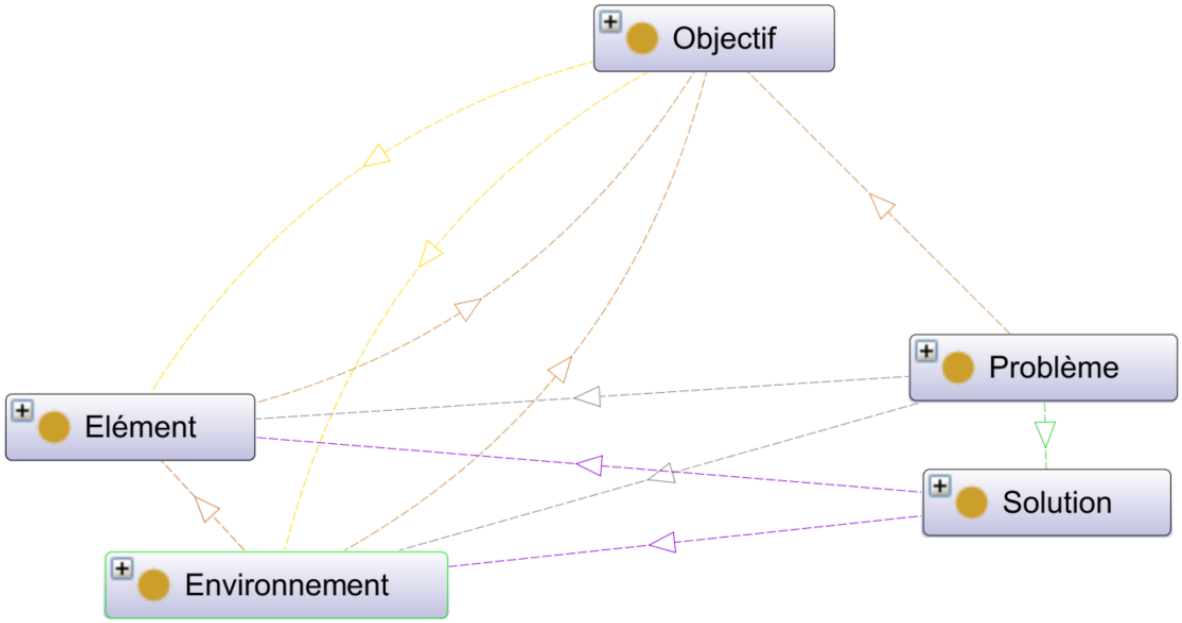


FIGURE 3.5 – Ontologie d'Operational Design

Contrairement aux bases de données relationnelles, les ontologies se distinguent par l'absence de tables de données. Ceci permet de gérer des données hétérogènes au même niveau au lieu de devoir créer des tables de relations supplémentaires.

La Figure 3.5 représente la structure de l'ontologie de mission. Elle a été réalisée à des fins d'illustration dans l'outil open-source Protégé [72] développé par le *National Institute of General Medical Sciences of the United States National Institutes of Health*.

On peut y voir les cinq concepts-clés du modèle ontologique : l'environnement (opérationnel), les éléments qui le constituent, les obstacles et leurs solutions, et enfin les objectifs. Les définitions suivantes, adaptées du Larousse [51], représentent les définitions des concepts de notre ontologie caractérisant notre contexte spécifique (Section 2.4).

Définition 3.1.1 (Environnement)**.** L'ensemble des éléments considérés dans la mission ou cadre de la mission.

Définition 3.1.2 (Élément)**.** L'entité ou l'objet ayant son unité et évoluant dans l'environnement.

Définition 3.1.3 (Objectif)**.** Le résultat vers lequel tend l'action de l'attaquant.

Définition 3.1.4 (Obstacle)**.** L'élément qui empêche l'attaquant de mener à bien sa mission.

Définition 3.1.5 (Solution)**.** L'occasion favorable qui permet de contourner un obstacle.

Ces concepts sont reliés par différentes relations présentées dans la Table 3.1. Dans la première colonne on peut voir le nom de chaque relation, c'est-à-dire son prédicat. Les colonnes "Sujet" et "Objet" indiquent les ensembles de définition respectifs du sujet et de l'objet de la relation. La Figure 3.5 représente l'ontologie métier pertinente à notre approche, et contenant les concepts relatifs à chaque processus.

Lors de la spécification de la mission, le concepteur capture les différents concepts qu'il juge pertinents dans le cadre de la mission. C'est à partir de ces concepts que les modèles d'environnement opérationnel courant, d'environnement opérationnel désiré et d'obstacles vont pouvoir être construits.

Relation	Sujet	Objet
A pour solution	Obstacle	Solution
A pour origine	Obstacle	Elément, Environnement
Agit sur	Solution	Elément, Environnement
Concerne	Objectif	Elément, Environnement
Contient	Environnement	Elément
Empêche	Obstacle, Élément, Environnement	Objectif

Table 3.1 – Relations du méta-modèle métier

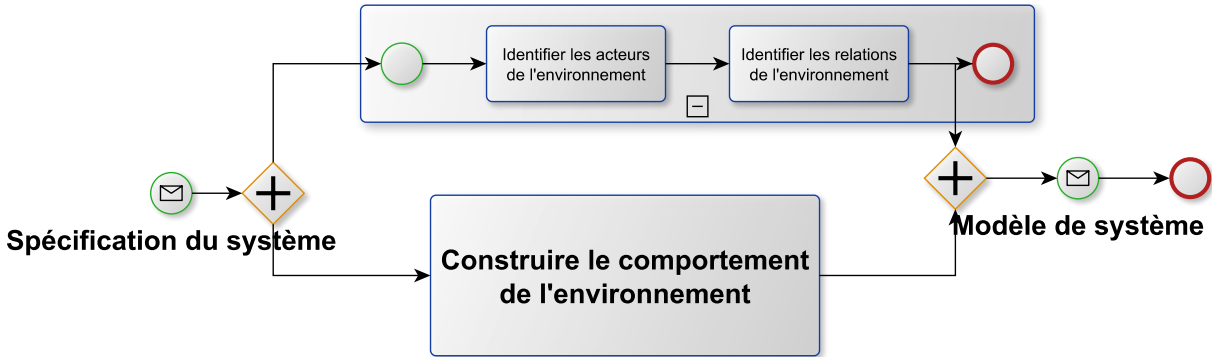


FIGURE 3.6 – Processus de modélisation de l'environnement opérationnel courant

3.1.2.2 Modélisation de l'environnement opérationnel courant

Identifier le système ciblé consiste à modéliser l'environnement opérationnel courant. Lors de ce processus, le concepteur doit développer un modèle représentant le théâtre des opérations dans lequel va se dérouler la mission. Ce processus a une place prépondérante pour conduire une méthodologie d'Operational Design de qualité, car ce sont les éléments identifiés par le concepteur lors de cette étape qui seront utilisés tout au long de la méthodologie. Toutes les analyses qui suivront reposent uniquement sur le modèle d'environnement opérationnel courant. Si un élément manque, alors il ne pourra pas être pris en compte par la suite, il faudra le rajouter dans le modèle de données.

Traditionnellement, dans un cadre militaire, le concepteur utilise des outils graphiques et textuels afin de représenter la situation de la manière qui lui semble la plus adaptée. Ce processus est donc relativement peu codifié et peut prendre des formes très différentes en fonction de la mission et du concepteur.

Dans le cadre des menaces persistantes avancées, l'environnement opérationnel courant comporte plusieurs caractéristiques récurrentes. En effet, les opérations des APT mettent en jeu un environnement au moins en partie informatique. Ceci implique un environnement constitué d'une ou plusieurs infrastructures réseaux ainsi que de divers éléments cyber-physiques Ces éléments s'ajoutent aux composants de systèmes usuellement considérés par la méthodologie militaire d'Operational Design, comme les structures physiques et les effectifs humains.

Bien que la modélisation de l'environnement opérationnel courant soit flexible, il est nécessaire de fournir un cadre de travail dans lequel le concepteur peut modéliser la situation dans sa globalité en prenant en compte les spécificités des éléments identifiés. C'est pourquoi il faut définir un méta-modèle adapté à la modélisation d'environnement opérationnel courant prenant en compte les spécificités des éléments à la fois cyber, cyber-physiques, purement physiques et humains.

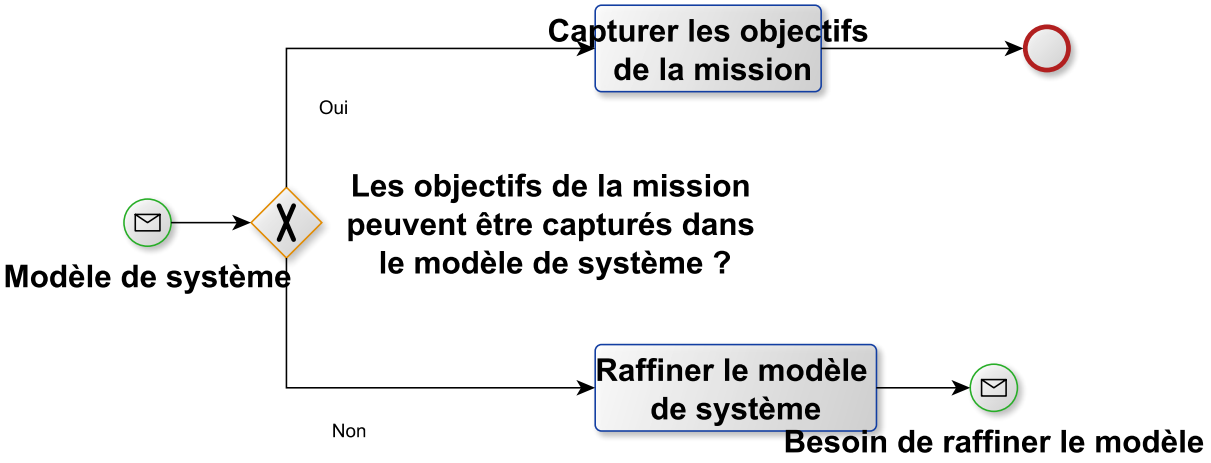


FIGURE 3.7 – Processus de modélisation de l'environnement opérationnel désiré

La Figure 3.6 présente le processus de modélisation de l'environnement opérationnel courant. À partir des spécifications du système cible et de diverses sources de renseignement obtenues au préalable, le concepteur doit construire la structure de l'environnement ainsi que son comportement. Dans le sous-processus du haut, on modélise la structure à travers les différents acteurs mis en jeu dans la situation puis à travers les relations qui les lient. En parallèle, le concepteur doit modéliser les différents comportements du système qui sont pertinents au regard de la mission à accomplir. À l'issue de ce processus, l'APT produit un modèle de système représentant l'ensemble du théâtre d'opérations considéré.

3.1.2.3 Modélisation de l'environnement opérationnel désiré

Identifier les objectifs de la mission consiste à modéliser l'environnement opérationnel désiré. Lors de ce processus, le concepteur s'appuie sur le modèle de système développé en tant qu'environnement opérationnel courant pour exprimer les objectifs de la mission en termes d'état du système désiré.

Les objectifs de la mission se traduisent par des propriétés désirées dans le système. Par exemple, l'objectif d'une menace persistante avancée peut être de mettre hors service un composant-clé du système, ou d'obtenir les accès nécessaires à l'ex-filtration de données du système.

Dans le cadre des menaces persistantes avancées, les propriétés désirées peuvent s'exprimer selon différents formalismes en fonction de la modélisation choisie pour l'environnement opérationnel courant. Toutefois, les objectifs présentent des caractéristiques propres au contexte des APT.

La Figure 3.7 présente le processus de modélisation de l'environnement opérationnel désiré. À partir du modèle de système créé auparavant, le concepteur tente de capturer les objectifs de la mission en termes de variable dans le modèle. Si cela est possible, les objectifs sont capturés pour l'étape d'analyse future. Dans le cas contraire, il est nécessaire de raffiner le modèle de système, c'est-à-dire qu'il faut reconduire un processus de modélisation de l'environnement opérationnel courant afin de prendre en compte les besoins des objectifs de la mission.

3.1.2.4 Modélisation des obstacles

Modéliser les obstacles posés par le système est un processus d'Operational Design nécessaire dans le cas non-trivial où les objectifs de la mission ne sont pas remplis avant

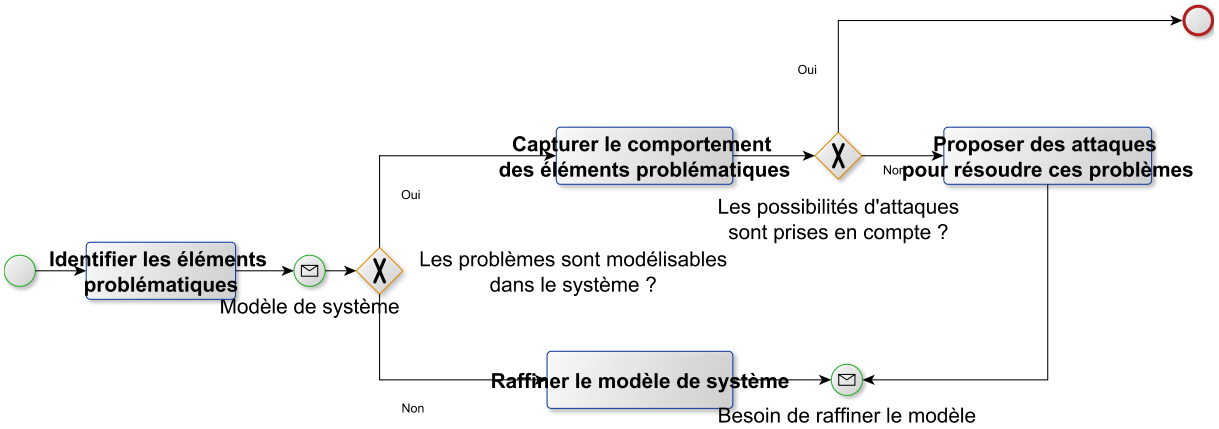


FIGURE 3.8 – Processus de modélisation des obstacles

l'intervention de la menace persistante avancée sur le système ciblé. Ce processus fait partie de la boucle de développement itératif de notre méthodologie. L'objectif de ce processus est de formuler différentes opportunités d'interactions de l'attaquant sur le système afin de suffisamment altérer le comportement du système et de parvenir à l'accomplissement de sa mission.

Lors de ce processus, le concepteur doit identifier les éléments qui l'empêchent d'orienter le système vers les objectifs. Ceux-ci sont identifiés à partir de la spécification de la mission et doivent se retrouver dans le modèle de système créé lors de l'étape de modélisation de l'environnement opérationnel courant. Cette étape est cruciale car elle identifie les éléments du système avec lesquels l'APT souhaite interagir pour parvenir à ses fins. Il faut donc identifier suffisamment d'éléments pour que les objectifs de l'attaque soient réalisables.

Une fois ces éléments identifiés, le concepteur doit imaginer des opportunités d'attaque contre ces éléments en s'appuyant sur la réalité. Par exemple, si un élément problématique est un employé dans l'entreprise du système ciblé, les vecteurs d'attaque de l'ingénierie sociale comme le *spear phishing* peuvent être envisagés. S'il s'agit d'un élément utilisant un système d'exploitation, les différentes failles de ce système peuvent être exploitées.

La Figure 3.8 présente le processus de modélisation des obstacles. À partir de la spécification de la mission et du modèle de système créé auparavant, le concepteur tente d'identifier les éléments problématiques. Si le concepteur envisage des obstacles qui ne sont pas modélisés dans le système, alors il est nécessaire de raffiner le modèle pour les prendre en compte. Si le modèle comporte ces éléments, alors il est possible d'envisager des opportunités d'attaques contre ces éléments. Ces possibilités amènent des modifications dans la structure ou dans le comportement du système qu'il faut donc modéliser en raffinant le modèle d'environnement opérationnel courant. Lorsque le concepteur est satisfait des différentes possibilités d'attaque qu'il a modélisées, il est possible de passer à l'étape d'analyse.

3.1.2.5 Analyse d'Operational Design

Notre approche propose une méthodologie d'analyse en trois étapes représentées dans la Figure 3.2 :

— satisfaction d'objectifs,

— élaboration d'influence,

— concrétisation du plan.

Ces étapes capturent la réflexion de l'APT sur le système cible. Elles utilisent les modélisations de l'environnement opérationnel courant, l'environnement opérationnel désiré et des obstacles afin de proposer une stratégie remplissant les objectifs de la mission.

La stratégie, ou *approche opérationnelle*, est une séquence ordonnée d'opérations dites "atomiques", c'est-à-dire que le concepteur estime que ces actions sont directement réalisables dans l'environnement sans requérir de réflexion supplémentaire.

Satisfaction d'objectifs

L'étape de test de satisfaction d'objectifs fait partie de la boucle de développement itératif de notre méthodologie. Lors de cette étape, le concepteur vérifie si l'environnement opérationnel souhaité est atteignable à partir de l'environnement opérationnel courant.

En d'autres termes, l'APT vérifie si elle peut orienter le comportement du système pour vérifier les propriétés désirées. Si c'est le cas, le concepteur peut passer à la concrétisation du plan et construire son approche opérationnelle finale. Dans le cas contraire, le concepteur doit modéliser les obstacles posés par le système.

Élaboration d'influence

Une fois les opportunités identifiées dans le processus de modélisation des obstacles, le concepteur doit les inclure dans l'environnement opérationnel courant. Autrement dit, le concepteur doit modifier la structure et/ou le comportement du système afin de prendre en compte ces opportunités. Il s'agit de la dernière étape de la boucle de développement itératif de notre méthodologie.

Une fois ces nouvelles opportunités ajoutées au modèle, le concepteur peut de nouveau tester si le modèle d'environnement opérationnel désiré est atteignable à partir du modèle d'environnement opérationnel courant et des opportunités de l'attaquant.

Concrétisation du plan

Si l'environnement opérationnel désiré est atteignable à partir de l'environnement opérationnel courant et des opportunités, le concepteur détermine le scénario le plus favorable pour parvenir aux objectifs de la mission. Ceci peut être fait selon différents critères comme par exemple le scénario qui fait intervenir le minimum d'opportunités de la menace persistante avancée ou le scénario qui minimise le risque en fonction d'une mesure de risque allouée à chaque opportunité.

Enfin, dans le cas où le scénario d'attaque fait intervenir une opportunité abstraite sur un composant dont la nature est incertaine, il est nécessaire de vérifier que l'opportunité est bien réalisable par de nouvelles opérations de reconnaissance préalables.

La Figure 3.9 présente le point de vue de processus-métier sur l'analyse. En exécutant le modèle, le concepteur détermine si la mission peut être accomplie. Si c'est le cas, le chemin d'attaque est identifié puis analysé afin de savoir si le chemin est concrètement réalisable. Dans la Figure 3.9 cette étape correspond au questionnement "Le chemin d'attaque est-il actionnable ?" Dans le cas contraire, les traces d'exécution permettent d'orienter le concepteur vers les problèmes à résoudre pour accomplir la mission. Ceci permet de ré-engager un processus de modélisation des obstacles.

3.1.2.6 Formulation de l'approche opérationnelle

Une fois le scénario d'attaque sélectionné et le plan concrétisé, le concepteur peut formuler la stratégie en des termes concrets et indépendants de la modélisation utilisée dans la méthodologie. Il formule son approche opérationnelle en un plan actionnable pour mener à bien sa mission.

La méthodologie que nous proposons a été formellement vérifiée au moyen de l'outil Pragmadev Process [11] utilisé pour modéliser les différents diagrammes BPMN. La véri-

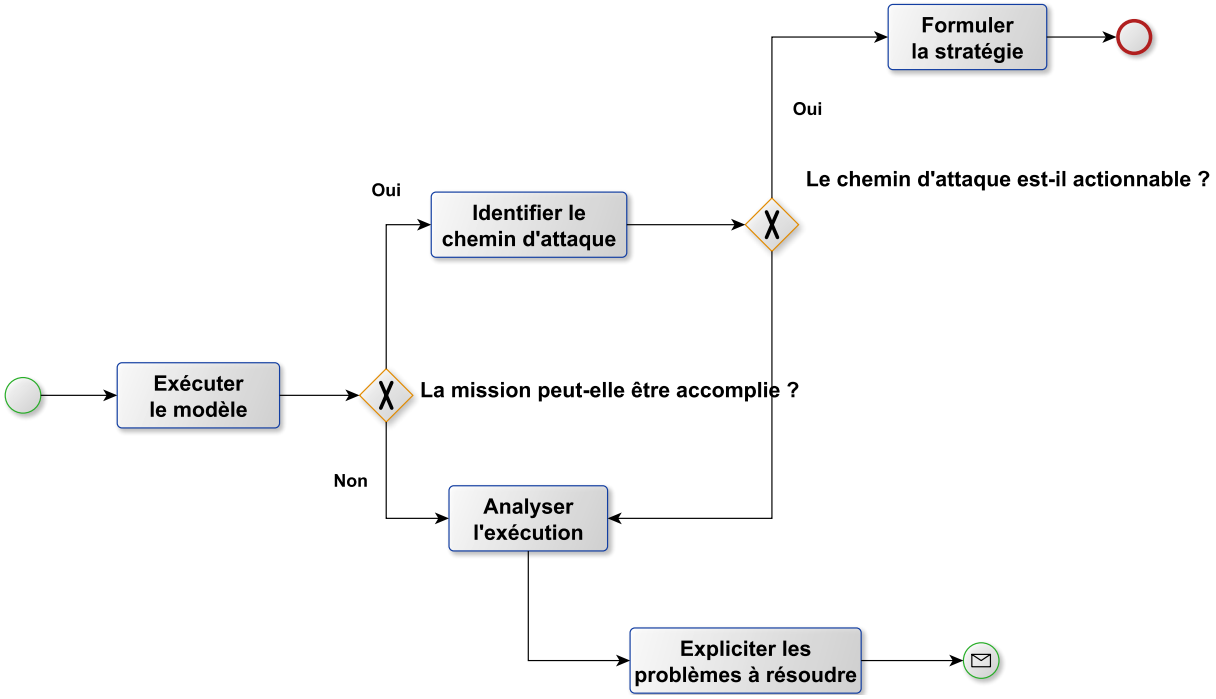


FIGURE 3.9 – Processus d'analyse

fication montre notamment l'enchaînement des différentes tâches qui mènent à la complétion du processus global. Le diagramme de séquences de l'Annexe B montre un exemple de l'utilisation de notre méthodologie dans le cas d'étude que nous proposons dans le Chapitre 4.

L'exploration des différentes séquences de tâches possibles met en évidence un aspect crucial à prendre en compte lors de l'application de notre méthodologie sur un cas concret : les trois processus que sont la modélisation de l'environnement opérationnel courant, la modélisation de l'environnement opérationnel désiré et la modélisation des obstacles reposent sur la manipulation et la modification d'un modèle de système qui évolue par itérations successives. Ceci introduit une problématique technique de synchronisation du modèle de système entre les trois processus avant de conduire les analyses. En effet, les objectifs exprimés dans l'environnement opérationnel désiré doivent être en adéquation avec le modèle de système courant proposé et les obstacles rencontrés.

En résumé, notre méthodologie globale permet à l'APT de concevoir une stratégie à partir d'une mission définie. Cette stratégie est formulée en termes de plan actionnable, ce qui permet à l'APT de mettre à exécution sa mission conformément à la Figure 3.1.

Pour ce faire, nous proposons une méthodologie itérative basée sur trois processus parallèles conformément à la méthodologie d'origine d'Operational Design. La méthodologie repose essentiellement sur une démarche itérative sur les différents processus qui interagissent par feedback mutuel.

Cette méthodologie générale peut être suivie de différentes manières en fonction de la mission à modéliser et des outils disponibles. Toutefois, nous pouvons affirmer que la méthodologie doit respecter les problématiques techniques suivantes :

— Spécifier la mission en termes d'environnement opérationnel, d'objectifs et d'obstacles rencontrés.

— Modéliser le système d'un point de vue structurel et comportemental.

— Modéliser les objectifs dans un formalisme compatible avec le modèle de système.

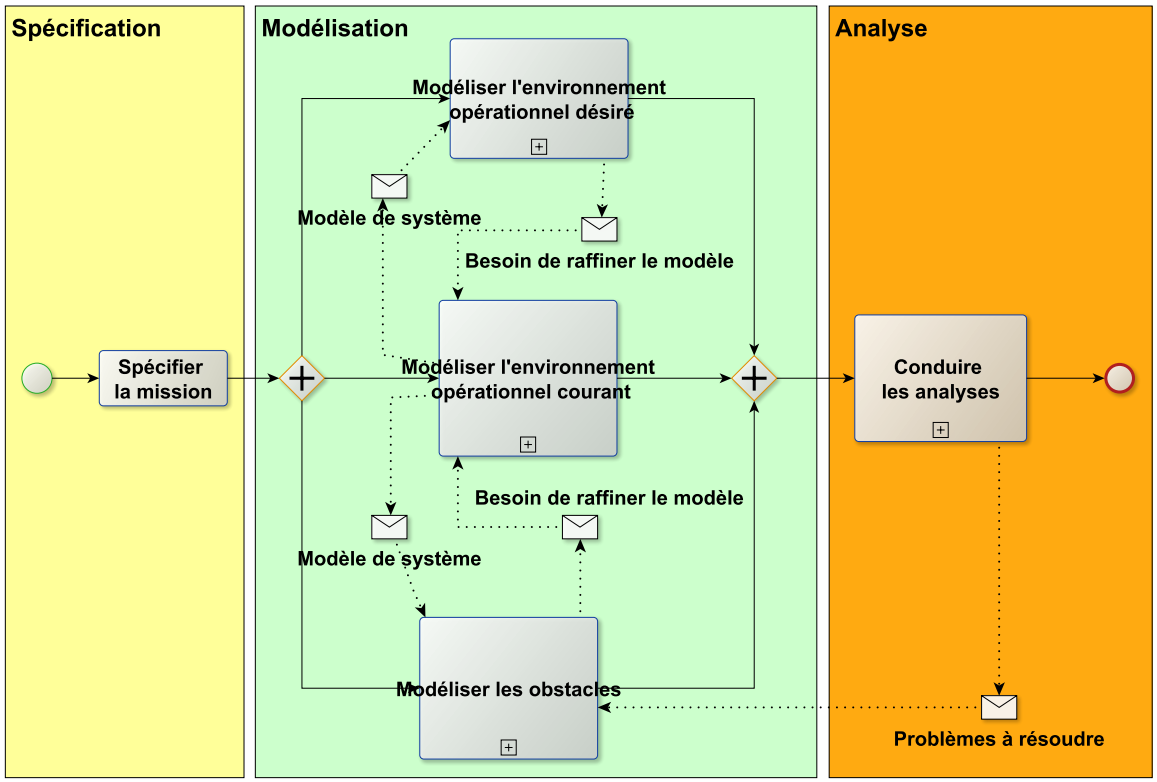


FIGURE 3.10 – Trois phases de la méthodologie

— Modéliser les obstacles dans un formalisme compatible avec le modèle de système.

— Gérer la synchronisation des différents processus et donc des différents modèles.

— Analyser les modèles au moyen d'outils formels.

3.1.3 Préoccupations techniques & Méthodologie détaillée

Dans le cadre de cette thèse, nous proposons de mettre en œuvre la méthodologie à travers un outillage impliquant différentes technologies. L'état de l'art du Chapitre 2 invoque plusieurs préoccupations techniques quant à la réalisation des processus abstraits que nous avons exposés. Dans cette partie, nous évoquerons les différentes solutions technologiques que nous avons apportées pour réaliser une méthodologie détaillée basée sur la méthodologie générale présentée précédemment.

3.1.3.1 Approche de modélisation

La méthodologie abstraite présentée en Figure 3.3 met en jeu différents processus de natures différentes. Tout d'abord, le rassemblement d'informations nécessaires à la méthodologie consiste à organiser des ressources documentaires de natures différentes afin de spécifier la mission.

Ensuite les processus de modélisation exigent d'employer conjointement une modélisation de systèmes, d'objectifs et de obstacles.

De plus, la méthodologie reposant sur le feedback de différents processus, il est attendu de notre approche qu'elle passe par de nombreuses itérations. Enfin, les analyses que nous outillons reposent sur une définition formelle des différents concepts de modélisation employés précédemment.

La Figure 3.10 présente les trois phases de la méthodologie : la spécification, la modélisation et l'analyse.

La synchronisation des modèles induit inévitablement des problématiques d'interopérabilité des modèles [109]. C'est pourquoi il est nécessaire d'adopter une approche modulaire dans la modélisation. En effet, les modèles manipulés sont amenés à évoluer régulièrement et leur modification entraîne d'autres modifications. Les approches d'intégration de modèles et d'unification de modèles sont donc contre-indiquées. Nous avons ainsi choisi d'adopter une approche de fédération de modèles pour pallier cette problématique d'interopérabilité conformément à l'état de l'art que nous avons établi dans le Chapitre 2.

Nous développons notre approche dans l'environnement de fédération OpenFlexo [35] afin d'outiller l'ensemble de notre approche de manière modulaire en utilisant la fédération de modèles.

Openflexo [1] [36, 37] est un environnement open-source de fédération de modèles. La fédération de modèles permet de lier les modèles de différents paradigmes à travers des relations sémantiques riches. Dans ce but Openflexo propose différents adaptateurs de technologies réutilisables dites "*commercial off-the-shelf*" (COTS) comme EMF, JDBC ou pdf.

Dans nos travaux, la fédération de modèles est notamment nécessaire pour gérer l'interopérabilité entre le domaine de la modélisation de système, le domaine de la modélisation d'objectifs et celui de modélisation des obstacles. En outre Openflexo nous permet de sérialiser nos modèles dans des paradigmes COTS afin de favoriser la réutilisabilité de nos travaux dans d'autres contextes de sécurité.

3.1.3.2 Spécification de la mission

Cette étape du processus consiste à agréger et à organiser l'ensemble des sources documentaires utilisées par le concepteur en plusieurs concepts pertinents pour l'approche. Les concepts que nous avons identifiés dans la méthodologie abstraite sont les suivants :

— L'environnement

— Les éléments constituants de l'environnement

— Les objectifs

— Les obstacles

— Les solutions

Les instances de concepts sont inférées depuis les ressources documentaires. On parlera donc dans la suite de ce manuscrit de **"concepts inférés"** (environnement, élément, objectif, obstacle et solution inférés).

Définition 3.1.6 (Concept inféré)**.** Un concept inféré est la représentation abstraite d'un objet ou d'une classe d'objets issue de l'agrégation et de l'organisation de ressources documentaires ou de modèles existants à des fins de modélisation.

L'outillage de cette étape pose des problèmes de gestion des connaissances :

— L'*élicitation* [18], c'est-à-dire la transformation des connaissances tacites du concepteur en connaissances explicites, claires et compréhensibles par l'ensemble des acteurs de la méthodologie.

— La *traçabilité* [8], c'est-à-dire la capacité à pouvoir relier chaque élément de connaissance à sa source et à son historique au cours de l'approche.

1. https://www.openflexo.org/

Pour répondre à ces préoccupations, l'environnement OpenFlexo dispose d'ores et déjà d'un outillage de fédération documentaire [38]. Nous développons plus en détail cet aspect d'implémentation dans la Section 3.2.

3.1.3.3 Modélisation de l'environnement opérationnel courant

Conformément à la méthodologie abstraite présentée en Figure 3.6, la modélisation de l'environnement opérationnel courant requiert de modéliser le système selon deux aspects : sa structure et son comportement.

Afin de conserver la fluidité de l'Operational Design dans la conception, nous avons décidé de modéliser la structure du système ciblé avec le langage Pimca. En effet, ce langage prend en compte des outils graphiques et des relations riches [92], ce qui est proche des méthodes utilisées dans la méthodologie d'Operational Design classique. Il permet également, grâce à son haut niveau d'abstraction, de modéliser des composants conceptuels. Ces composants peuvent représenter une incertitude sur le système de la part de la menace persistante avancée. Ils permettent également de représenter en un seul composant, un élément du système englobant une multitude de composants que le concepteur ne juge pas pertinent de modéliser en détail dans son analyse. De plus, Pimca se veut agnostique vis-à-vis des méthodes d'analyses de sécurité. Puisque la qualité de l'approche globale repose tout particulièrement sur la modélisation adéquate du système ciblé, le langage Pimca est donc un choix cohérent.

De manière similaire au cadre militaire, les systèmes ciblés par les APT ont une composante dynamique non négligeable, dans le sens où ce sont des systèmes qui évoluent dans le temps. Comme le montre le Chapitre 2, Pimca ne prend pas en compte ces problématiques. C'est pourquoi le langage Pimca n'est pas suffisant pour appréhender cette évolution.

Nous proposons donc de modéliser l'évolution du système par ce qu'on appelle un modèle de comportement. Le modèle de comportement est écrit dans un nouveau langage que nous avons développé comme extension dynamique de Pimca, DyPimca. Avec DyPimca, l'APT modélise le comportement des différents composants qui rentreront en jeu tout au long de l'approche. Le langage dynamique sera détaillé dans la partie 3.3.3. Comme pour la modélisation de la structure du système, il est crucial de modéliser tous les comportements des éléments du système que le concepteur souhaite faire intervenir par la suite.

Le modèle de système ciblé a donc deux composantes : une structure définie dans le langage Pimca et un comportement défini dans le langage DyPimca. Dans la Figure 3.6, le langage Pimca modélise les acteurs et les relations tandis que le comportement du système est modélisé en DyPimca. Avec nos outils, il est dès lors possible d'observer le comportement du système sans aucune interaction de la menace persistante avancée, c'est ce qu'on appelle le "comportement nominal" du système. Différents outils, comme la vérification de modèle, permettent de valider le comportement nominal du système comme un fonctionnement cohérent et attendu par le concepteur avant de conduire un autre processus.

On note toutefois que le langage Pimca n'est pas outillé, c'est-à-dire que sa syntaxe abstraite est définie par la DGA, mais que son implémentation concrète n'a pas été proposée dans la littérature. C'est pourquoi il est également nécessaire de proposer un outillage autour du langage Pimca pour mettre en place l'ensemble de la méthodologie. Nous développons ces aspects dans la Section 3.3.

À terme, le processus de modélisation de l'environnement opérationnel désiré produit un modèle de système DyPimca dans lequel l'APT va pouvoir organiser sa stratégie. Pour cela, l'APT doit capturer l'environnement opérationnel désiré et les obstacles. Par opposition avec les **"concepts inférés"**, les modèles de système produits lors de ce processus sont dits **"concepts opérationnels"**, dans le sens où ils sont directement utilisables dans les analyses.

Définition 3.1.7 (Concept opérationnel)**.** Un concept opérationnel est la représentation concrète et formelle d'un objet ou d'une classe d'objets issue de la modélisation à des fins d'analyses formelles.

3.1.3.4 Modélisation de l'environnement opérationnel désiré

Conformément à la méthodologie abstraite présentée en Figure 3.7, la modélisation de l'environnement opérationnel désiré requiert de capturer les objectifs de la mission dans le modèle de système cible. Le processus de réification des objectifs se base lui-même sur le processus de l'environnement opérationnel courant, c'est-à-dire sur la modélisation du système. De plus, notre méthodologie exige une réification de ces objectifs dans le but de conduire des analyses formelles.

Les préoccupations de la modélisation de l'environnement opérationnel désiré sont donc :

— Une compatibilité avec le modèle de système.

— Une compatibilité avec les analyses formelles.

Puisque ce processus est entièrement dépendant de la modélisation du système, nous devons proposer un modèle d'objectifs en adéquation avec notre choix du langage Pimca étendu d'une part et les outils d'analyse formelle que nous choisissons. Pour répondre à ces besoins, nous modélisons les objectifs opérationnels de la mission sous la forme de propriétés LTL compatibles avec le langage DyPimca que nous avons développé. Nous développons ces aspects dans la Section 3.4.

3.1.3.5 Modélisation des obstacles

Pour modéliser les obstacles présentés par le système ciblé, la méthodologie abstraite présentée en Figure 3.8 prescrit une approche en deux temps :

— Modéliser les éléments problématiques du système.

— Modéliser les attaques envisagées sur ces éléments problématiques.

De même que la modélisation de l'environnement opérationnel désiré, la modélisation des obstacles opérationnels repose sur le modèle de système. Il donc nécessaire de proposer un modèle de obstacles opérationnels adapté. Nous développons ces aspects dans la Section 3.5.

3.1.3.6 Analyse

Les analyses de la méthodologie abstraite présentées en Figure 3.9 se basent sur l'exécution conjointe des modèles de système, d'objectifs et de obstacles. Dans la pratique, en Operational Design, ce processus est conduit "à la main" par le concepteur. Il est donc sujet à une grande variabilité. Dans notre méthodologie, grâce aux modélisations du système, des objectifs et des obstacles, les différentes étapes de ce processus peuvent être partiellement automatisées pour aider le concepteur dans sa tâche et prodiguer un socle commun d'analyse.

Concrètement, dans le cas d'une modélisation de système avec DyPimca et de modélisation d'objectifs à travers la LTL, le concepteur peut utiliser des outils de vérification formelle comme OBP2 pour vérifier automatiquement si le système peut atteindre son état désiré.

Grâce à la vérification de modèles, le concepteur peut être guidé par des scénarios de fonctionnement qui servent de contre-exemple à l'accomplissement de la mission pour analyser les obstacles posés par le système.

Processus	Besoin	État de l'art	Technologie retenue
Approche globale	Interopérabilité	✓	Fédération de modèles
Spécification de la mission	Élicitation	✓	Fédération documentaire
	Traçabilité	✓	
Environnement opérationnel courant	Modèle structurel	∼	Langage Pimca étendu et outillé
	Modèle comportemental	X	
Environnement opérationnel désiré	Compatibilité système	X	Propriétés LTL
	Compatibilité analyse	X	
Obstacles	Identification	X	À la main
	Opportunité	X	
Analyse	Satisfaction d'objectifs	∼	Model-checking
	Concrétisation du plan	X	À la main

Table 3.2 – Exigences de la méthodologie

Pour ce faire, les scénarios contre-exemples produits par l'étape de satisfaction d'objectifs peuvent aider le concepteur à analyser le comportement du système et à identifier les éléments problématiques. Nous développons ces aspects dans la Section 3.6.

3.1.4 Bilan et positionnement de CyberOD

La méthodologie que nous proposons est formulée en termes abstraits. Nous proposons un outillage concret afin d'appliquer notre méthodologie sur un cas d'étude.

Le Tableau 3.2 identifie l'ensemble des exigences de l'implémentation de notre méthodologie ainsi que les manques que nous avons identifiés dans la littérature. L'outillage des cinq processus exigent une approche globale modulaire qui pose des problématiques d'interopérabilité. Pour ce faire, nous adoptons une approche de fédération à l'aide de l'environnement de fédération OpenFlexo [36, 37, 35, 38].

Afin d'appliquer cette méthodologie abstraite à travers un outillage concret, la Figure 3.3 présente les cinq processus qui devront être outillés. L'état de l'art montre qu'il existe un certain nombre de solutions pour mettre en place ces processus.

Pour le premier processus, spécifier la mission, la fédération documentaire est d'ores et déjà outillée dans OpenFlexo. Pour la modélisation de l'environnement opérationnel courant, le langage Pimca [92] proposé dans la littérature répond partiellement à nos besoins. Toutefois, il nécessite une extension pour prendre en compte les aspects dynamiques. Il nécessite également de développer un outillage concret. Pour la modélisation de l'environnement opérationnel désiré, nous capturons les objectifs de la mission sous la forme de propriétés LTL. Ce choix est motivé par la compatibilité des modèles DyPimca avec un formalisme LTL d'une part et par la compatibilité de notre processus d'analyse avec ce même formalisme. Nous développons enfin un modèle pour la modélisation des obstacles. Pour les analyses, nous choisissons de connecter l'ensemble des modèles outillés en OpenFlexo avec le model-checker OBP2. En effet, l'environnement de développement par fédération OpenFlexo permet le développement rapide de connexions avec d'autres outils. De plus, OBP2 est un model-checker proposant d'établir la compatibilité avec tout modèle à l'aide de la définition d'un plugin adapteur simple.

Compte tenu des différents besoins de l'ensemble des processus, la méthodologie d'Operational Design adaptée aux APT impose un environnement hétérogène de modélisation. Dans ce contexte, les approches d'intégration semblent donc contre-indiquées. De même, les approches d'unification fixeraient le langage-pivot entre les concepts communs des différents langages de modélisation. Pour capitaliser sur les modèles utilisés dans une ins-

tance de la méthodologie dans un nouveau contexte, l'unification semble contre-indiquée. C'est pourquoi nous proposons une approche par fédération de modèles pour répondre au besoin de capitalisation sur un ensemble de concepts-métiers, conformément à l'état de l'art établi en Section 2.4.2.

3.1.4.1 Exigences

La méthodologie requiert d'employer une approche de fédération pour plusieurs raisons.

L'ensemble de la méthodologie est établi dans un environnement hétérogène comprenant plusieurs phases de développement et impliquant des artefacts technologiques de différentes natures. Gérer l'interopérabilité de l'ensemble des solutions technologiques choisies pour chaque processus demande d'adopter une approche flexible. Il est donc inconcevable d'employer une approche rigide d'intégration, qui demanderait de définir un langage global rassemblant l'ensemble des **concepts** à la fois **inférés** et **opérationnels**. L'hétérogénéité de ces concepts empêche la définition d'un tel langage. Une approche d'unification pourrait résoudre ces problèmes à l'aide de la définition d'un langage-pivot rassemblant les concepts communs aux différents paradigmes de spécification, de modélisation et d'analyse.

Toutefois, la méthodologie que nous proposons peut s'outiller de manière flexible selon les besoins du concepteur et de la mission à accomplir. Ceci introduit un besoin de capitalisation sur les outils implémentés dans des instances antérieures de la méthodologie. En effet, l'APT peut avoir besoin d'utiliser une partie des solutions techniques mises en place dans le cadre d'une autre mission en la combinant avec des solutions technologiques nouvelles. De fait, l'approche d'unification n'est plus adaptée. La définition du langage-pivot fixe les concepts communs une fois pour toutes, ce qui empêche la réutilisabilité et la capitalisation des acquis de l'approche.

C'est pourquoi il est nécessaire d'adopter une approche de fédération pour gérer l'interopérabilité de la méthodologie.

3.1.4.2 Liens de fédération

L'approche de fédération consiste à définir des liens sémantiques dits "de fédération" entre les différents paradigmes qui interviennent au cours de la méthodologie. Contrairement à l'approche d'unification, qui définit un modèle-pivot entre les différents paradigmes, la fédération se concentre sur la définition de liens sémantiques entre les concepts de chaque paradigme. La fédération fait abstraction du modèle-pivot. Les liens sont définis ad hoc pour chaque paradigme utilisé. Pour cela, il est nécessaire de définir un ensemble de concepts-métiers auxquels relier les éléments des paradigmes correspondants. Afin de mettre en place une telle approche, il faut donc définir l'ensemble des considérations de cyber-sécurité qui serviront de liens de fédération.

En résumé, l'approche que nous proposons se base sur la méthodologie de l'Operational Design issue du domaine militaire. Nous adoptons cette méthodologie dans le contexte de la cyber-sécurité et plus particulièrement pour modéliser l'élaboration de stratégies par les menaces persistantes avancées.

Pour cela, nous proposons une méthodologie abstraite utilisant cinq processus décrits dans la Figure 3.3 :

— Spécification de la mission
— Modélisation de l'environnement opérationnel courant
— Modélisation de l'environnement opérationnel désiré
— Modélisation des obstacles
— Analyse

Ensuite, nous implémentons la méthodologie abstraite de manière concrète en utilisant un ensemble de technologies. Le développement est conduit dans l'environnement de fédération de modèles OpenFlexo. Le modèle de système est construit dans le langage DyPimca que nous développons en enrichissant et en outillant le langage Pimca de la littérature. Les modèles d'objectifs et d'obstacles sont développés de manière à faciliter la compatibilité avec l'outil d'analyse que nous avons choisi : le model-checker OBP2. Pour cela, nous capturons les objectifs sous la forme de propriétés LTL.

La méthodologie concrète est adaptée à la validation du Chapitre 4. La méthodologie abstraite et l'outillage proposés restent également réutilisables dans d'autres contextes. La séparation des aspects abstrait et concret permet néanmoins une flexibilité dans l'application de notre méthodologie.

Dans la suite de ce chapitre, nous présentons plus en détail les différents outils que nous avons développés afin de combler les manques de la littérature pour conduire notre méthodologie jusqu'à une implémentation concrète.

3.2 Spécification de la mission

La phase de spécification est la première phase de la méthodologie. Celle-ci ne comporte qu'un seul processus qui consiste à spécifier la mission. Lors de ce processus, le concepteur rassemble les sources d'informations nécessaires à l'exécution de l'ensemble de la méthodologie. Il effectue également une identification des **concepts inférés** liés au métier de la cyber-sécurité qui serviront de données d'entrée pour la phase de modélisation. Ces concepts inférés sont organisés de manière cohérente afin de faciliter la conduite des phases de modélisation et d'analyse.

3.2.1 Besoins

La spécification de la mission est basée sur un ordre de mission explicité par la hiérarchie. À partir de cet ordre de mission, le concepteur identifie puis exprime les aspects métiers de la mission au moyen de ressources documentaires. Ces ressources sont diverses et hétérogènes. Par exemple, le concepteur peut étudier des documents de spécification du système pour spécifier la configuration de l'environnement de la mission ou des bases de vulnérabilités pour spécifier des opportunités d'attaques au cours de la mission.

Ceci introduit un besoin qu'on appelle, en ingénierie des connaissances, l'**élicitation**.

Définition 3.2.1 (Élicitation)**.** L'action d'aider un expert à formaliser ses connaissances pour permettre de les sauvegarder ou de les partager [1].

Dans notre contexte, l'élicitation décrit, d'une part, le besoin du concepteur d'expliciter l'ensemble des éléments nécessaires à la conduite de l'approche à partir de l'ordre de mission donné. Ces éléments englobent les différents aspects métiers qui seront utilisés dans les étapes de modélisation et d'analyses. D'autre part, l'élicitation décrit le besoin du concepteur d'organiser ces différents aspects métiers de manière formelle, afin de permettre de sauvegarder et de partager un ensemble cohérent de connaissances.

L'ensemble des aspects métiers nécessaires au bon déroulement de l'approche peut varier en fonction de la mission en question. Toutefois, nous avons explicité dans la Section 3.1.3 les concepts minimums indépendamment de la mission, à savoir : l'environnement, les éléments constituants, les objectifs, les obstacles et les solutions. À l'issue de la phase de spécification, le concepteur doit au minimum établir ces concepts inférés avant de pouvoir poursuivre.

Les choix de conception de l'approche globale sont directement influencés par les ressources documentaires consultées et les concepts inférés par le concepteur. C'est pourquoi il est nécessaire de conserver une trace de ces choix justifiés par des références aux ressources. La spécification de la mission introduit donc la problématique technique de la **traçabilité**.

Définition 3.2.2 (Traçabilité)**.** La capacité à suivre un produit tout au long de la chaîne, de l'approvisionnement en matière premières à la mise au rebut [49, 55].

Dans le contexte de notre approche, la traçabilité désigne plus spécifiquement la capacité à suivre l'évolution des concepts au cours des différentes phases de l'approche. En particulier, il est nécessaire de pouvoir retracer l'origine d'un concept opérationnel par rapport au concept métier correspondant. De même, il est crucial de pouvoir retrouver la source documentaire justifiant les choix de conceptions tout au long de l'approche.

La spécification de la mission est la première étape de notre méthodologie. À ce titre, elle prédétermine l'ensemble des résultats de l'approche. Il est donc nécessaire de faire preuve d'une rigueur particulière lors de cette phase. Cette rigueur se traduit en deux besoins que nous venons d'expliciter : l'**élicitation** et la **traçabilité**.

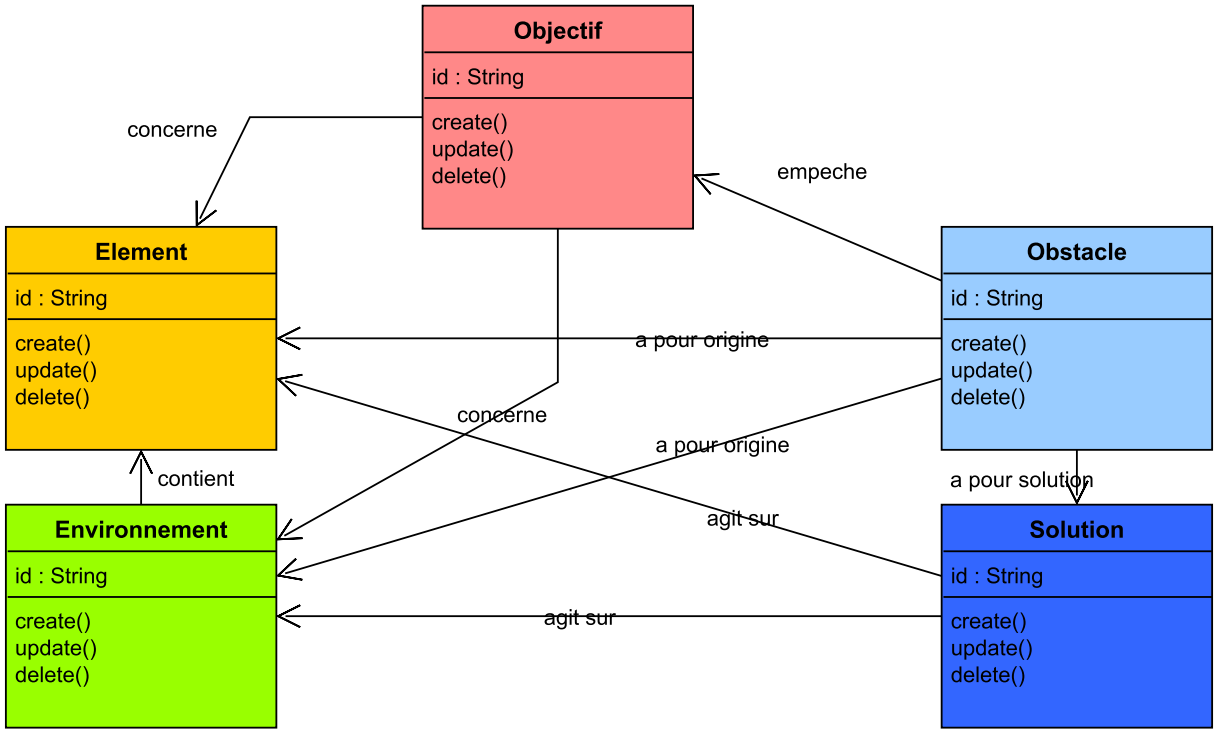


FIGURE 3.11 – Spécification formelle du modèle de connaissances

3.2.2 Fédération documentaire

Pour répondre aux besoins d'élicitation et de traçabilité, nous choisissons de mettre en œuvre la spécification de la mission au moyen de la fédération documentaire. La fédération documentaire [38] est une approche qui consiste à relier des besoins informels à une spécification formelle à travers une traçabilité précise. Les besoins informels peuvent s'exprimer selon des formats très divers comme des documents textuels ou des bases de données. La spécification formelle, quant à elle, suit un format bien défini que nous explicitons à l'aide des concepts métiers identifiés au préalable. Cette spécification prend la forme du modèle de données organisées des concepts inférés. Dans cette section, nous expliciterons la spécification formelle produite au terme de la phase de spécification et les mécanismes abstraits de la fédération documentaire.

3.2.2.1 Spécification formelle des concepts inférés

La spécification formelle des concepts inférés est le modèle de connaissances auquel se réfèreront tous les processus de la phase de modélisation. Le modèle de connaissances est formulé par le concepteur pour prendre en compte les préoccupations spécifiques liées à la mission. Néanmoins, les cinq concepts (environnement, élément constituant, objectif, obstacle et solution) identifiés dans la Section 3.1.3 sont des éléments que nous jugeons nécessaires d'inclure dans le modèle de connaissances pour chaque mission. Dans le cadre de ce manuscrit, nous présentons un modèle de connaissances minimum utilisant les cinq concepts comme démonstration de faisabilité. Cette spécification formelle sera utilisée concrètement dans le cas d'étude du Chapitre 4.

Le modèle de connaissances suit un formalisme plus restreint qu'une ontologie pour des besoins d'implémentation dans le cadre de la validation. La Figure 3.11 présente le méta-modèle auquel se conforme le modèle de connaissances. Les concepts sont reliés par des relations sémantiques qui sont utilisées dans la phase de modélisation.

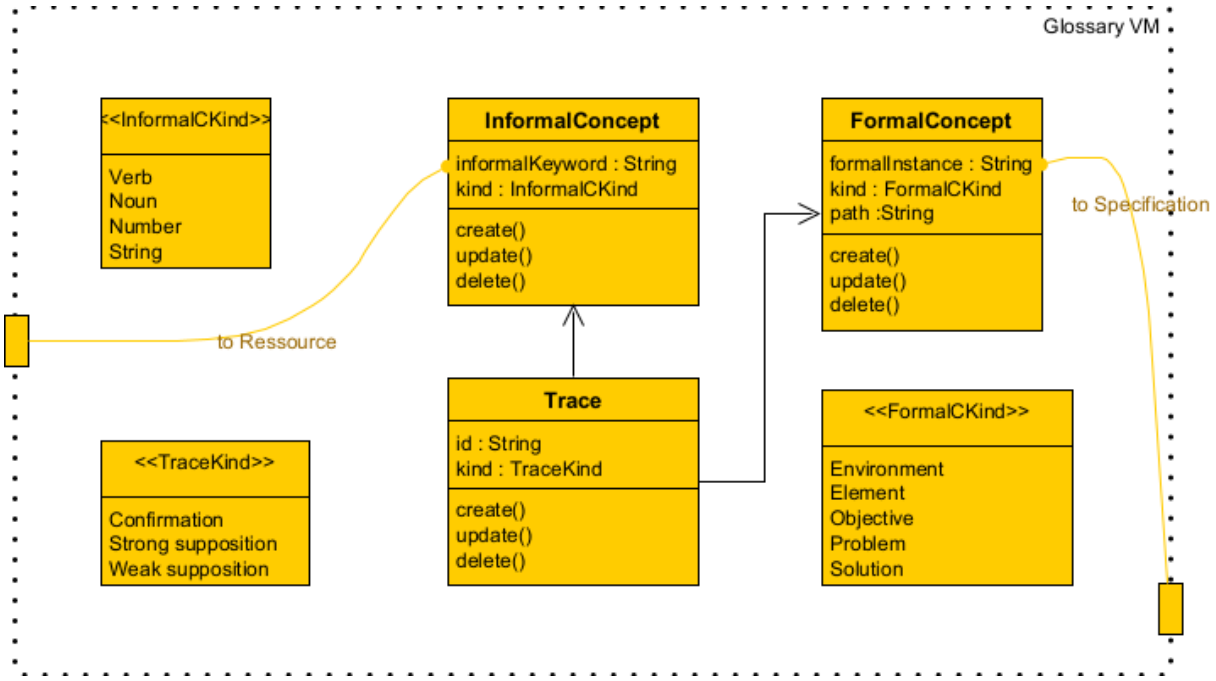


FIGURE 3.12 – Trace, lien de fédération documentaire

— L'**environnement** décrit le cadre des opérations qui seront conduites dans le cadre de la mission. Il peut faire référence à des instances d'éléments contenus à travers la relation "contient".

— Les **éléments** constituants de l'environnement sont des composants que le concepteur juge importants de spécifier.

— Les **objectifs** décrivent le but de la mission, ils font référence à l'environnement dans son ensemble et/ou à des éléments spécifiques de l'environnement. Ils peuvent faire référence à des instances d'environnement ou d'élément à travers la relation "concerne".

— Les **obstacles** décrivent les éléments pouvant nuire à la réalisation d'un ou plusieurs objectifs. Ils font référence aux objectifs qu'ils concernent à travers la relation "empêche" et aux éléments ou environnement où ils trouvent leur origine à travers la relation "a pour origine". Enfin ils peuvent faire référence à des instances de solutions dédiées à résoudre le problème à travers la relation "a pour solution".

— Les **solutions** sont des actions envisagées par le concepteur pour résoudre les problèmes identifiés. Elles agissent sur des éléments ou l'environnement à travers la relation "agit sur".

3.2.2.2 Liens de fédération documentaire

Les liens entre les différentes ressources documentaires et la spécification formelle de la mission se font au moyen d'un modèle de glossaire précédemment défini par Golra *et al.*[38] présenté en Figure 3.12. Le concept de *Trace* relie un concept informel issu de la documentation (*InformalConcept*) à un concept formel du modèle formel de connaissances (*FormalConcept*).

L'*InformalConcept* est relié à une ressource documentaire à travers un mot ou une phrase appelée *informalKeyword*. Le mot-clé est d'un type grammatical spécifié par l'attribut *kind* parmi les types disponibles dans l'énumération *InformalCKind*.

Le *FormalConcept* est relié à un concept du modèle formel de connaissances à travers une instance de connaissance appelée *formalInstance*. Le type de cette instance est précisé dans l'attribut *kind* parmi les cinq concepts utilisés par le modèle de connaissances.

La *Trace* relie les concepts informel et formel. Elle possède un identifiant unique ainsi qu'un type défini dans l'attribut *kind* parmi l'énumération *TraceKind*. Le type de *Trace* dépend du degré d'incertitude que le concepteur émet sur les concepts qu'il infère.

— La **confirmation** connecte un élément issu de la documentation à un élément de concept formel inféré. Le concepteur a la certitude que les éléments issus de la documentation sont fiables et correspondent à la réalité dans le cadre de l'environnement opérationnel.

— La **supposition forte** connecte un élément issu de la documentation à un élément de concept formel inféré. Le concepteur émet une hypothèse fondée sur des ressources documentaires afin de poursuivre l'approche d'Operational Design. Par exemple, une solution technique peut être envisagée par le concepteur alors que son développement n'est pas encore terminé.

— La **supposition faible** connecte un élément issu de la documentation à un élément de concept formel inféré. Le concepteur émet une hypothèse fondée sur peu d'éléments de ressources documentaires afin de poursuivre l'approche d'Operational Design. Ce type de *Trace* est à proscrire au terme de plusieurs itérations d'Operational Design mais peut être nécessaire pour mener une première approche.

La fédération documentaire est une solution qui répond aux besoins d'**élicitation** et de **traçabilité** en mettant en place un modèle de fédération dit "glossaire". Ce modèle virtuel permet de lier les éléments issus de ressources documentaires à des éléments du modèle de connaissances inférées à travers le concept de ***Trace***.

3.2.3 Implémentation

Concrètement nous avons implémenté la fédération documentaire dans le cadre de l'environnement de fédération OpenFlexo [35]. En effet, celui-ci propose un outillage permettant la fédération documentaire [38].

Cette approche permet la spécification de la mission, c'est-à-dire la description rigoureuse et formelle de la mission à partir d'artefacts divers et informels issus de la documentation et d'opérations de reconnaissance préalables. Ce faisant, l'outil OpenFlexo permet de maintenir les liens entre les documents d'origine et la spécification dans le temps et à travers différentes itérations des documents d'origine et de la spécification.

Après la phase de spécification de la mission, il est désormais possible de commencer la phase de modélisation. Cette phase se décompose en trois processus (modélisation du système, des objectifs et des obstacles) que nous aborderons dans cet ordre dans la suite de ce chapitre.

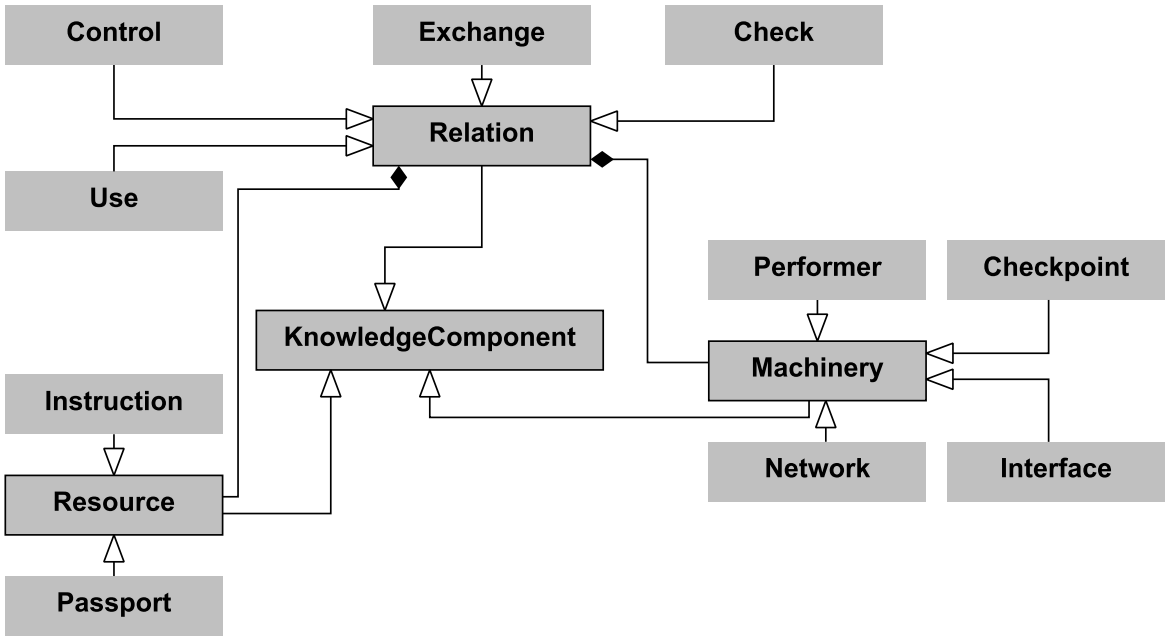


FIGURE 3.13 – Méta-modèle Pimca

3.3 Modélisation de systèmes

La phase de modélisation de notre approche comporte trois processus, dont la modélisation de systèmes. Ce processus consiste en la modélisation de la structure et du comportement de l'environnement opérationnel courant, c'est-à-dire du système cible. Il s'appuie sur la spécification de la mission qui a permis d'identifier les éléments pertinents à modéliser au cours de ce processus. Le modèle produit servira ensuite comme support de réflexion et d'analyse pour la suite de l'approche.

Le framework que nous proposons permet d'effectuer l'ensemble des tâches de la méthodologie concrète avec aisance. Nos outils permettent à l'utilisateur de gérer l'interopérabilité de différents modèles, d'identifier les potentielles incohérences entre les modèles et de mener à bien des analyses de sécurité pour formuler une approche opérationnelle cohérente et réalisable.

En particulier pour la modélisation de système, nous choisissons d'outiller le langage Pimca présenté dans le Chapitre 2.4.

3.3.1 Besoins

Le langage Pimca modélise les systèmes pour permettre différentes analyses de sécurité comme l'analyse de surface d'attaque ou le développement de modèles d'attaquant. Ceci implique de faire intervenir d'autres types de modèles dans le cadre de ces analyses selon le principe de l'ingénierie dirigée par les modèles. C'est pourquoi il est nécessaire de proposer un framework adapté à l'interopérabilité de modèles Pimca avec des modèles d'analyses de sécurité. Le framework que nous avons développé autour du langage Pimca gère ce problème d'interopérabilité à travers une technique de fédération de modèles offerte par l'environnement Openflexo.

Le langage Pimca possède d'ores et déjà une syntaxe abstraite définie par la DGA [92]. La Figure 3.13 rappelle sous la forme d'un diagramme de classe le méta-modèle du langage Pimca, c'est-à-dire le modèle qui décrit les règles de construction du langage tel qu'il est

manipulé par la machine. La sémantique du langage est également déjà définie par la DGA au plus proche du domaine métier de la modélisation de systèmes pour les analyses de sécurité.

Afin de concrétiser notre approche, il est néanmoins nécessaire d'ajouter une **syntaxe concrète** au langage Pimca. Cette syntaxe doit également être compatible avec l'ensemble de notre outillage.

De plus, comme l'a souligné le Chapitre 2.4, l'approche d'Operational Design nécessite de modéliser l'évolution du système au cours du temps et des manœuvres de l'attaquant. Cet aspect n'est pas pris en compte dans le langage Pimca à l'heure actuelle, il est donc nécessaire de l'enrichir avec un **aspect dynamique**.

Pour les besoins de notre approche, nous employons le langage Pimca pour la modélisation de systèmes. Ce langage est d'ores et déjà adapté à plusieurs exigences de notre approche comme le montre l'état de l'art. Toutefois, il reste à combler deux besoins afin d'utiliser le langage Pimca : formuler une **syntaxe concrète** du langage adaptée à notre outillage et étendre le langage Pimca par un **aspect dynamique**.

3.3.2 Syntaxe concrète du langage Pimca

Il reste donc à définir une syntaxe concrète du langage Pimca pour pouvoir l'inclure dans le framework que nous développons dans cette thèse. À l'aide de l'environnement de développement Openflexo, nous proposons deux syntaxes concrètes différentes pour s'adapter aux besoins d'un maximum d'utilisateurs : une syntaxe graphique et un éditeur de modèle en arborescence.

3.3.2.1 Syntaxe concrète graphique

Conformément à la philosophie de l'IDM, nous définissons des symboles graphiques pour représenter les machineries et ressources définies par la syntaxe abstraite de Pimca. La Figure 3.14 montre les différentes représentations des machineries et des ressources dans le langage graphique.

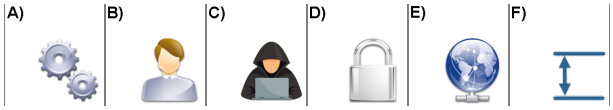
Le concept de *knowledgeComponent* étant abstrait, c'est-à-dire non instanciable, il n'a pas besoin d'avoir une représentation graphique. Enfin, les différentes relations sont représentées par des flèches de différentes couleurs pour chaque type. En particulier la relation d'échange est la seule relation bidirectionnelle, elle est représentée en bleu sans légende. Les autres relations sont unidirectionnelles, elles sont donc représentées par les couleurs rouge pour le contrôle, vert pour l'utilisation, noir pour la production et violet pour la vérification. De plus, les flèches sont annotées pour indiquer le type de relation qu'elles représentent.

Toutes ces représentations sont manipulables par l'utilisateur au moyen de l'interface graphique de notre framework utilisant le framework de fédération. Pour cela, Openflexo met en place un adaptateur de diagrammes et le langage au cœur d'OpenFlexo FML.

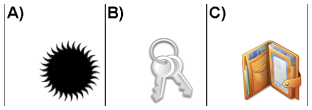
3.3.2.2 Syntaxe concrète en arborescence

De même que pour la syntaxe concrète graphique, nous définissons une syntaxe concrète en arborescence en utilisant le format de fichier xml. Ce format est également supporté par OpenFlexo à travers un adaptateur textuel.

La Figure 3.15 montre l'interface graphique du framework Pimca que nous avons mis en place avec la technologie OpenFlexo :



(a) A) Machinery, B) Performer, C) Attacker, D) Checkpoint, E) Network, F) Interface



(b) A) Resource, B) Passport, C) Instruction

FIGURE 3.14 – Représentation graphique des instances de Machinery et Resource

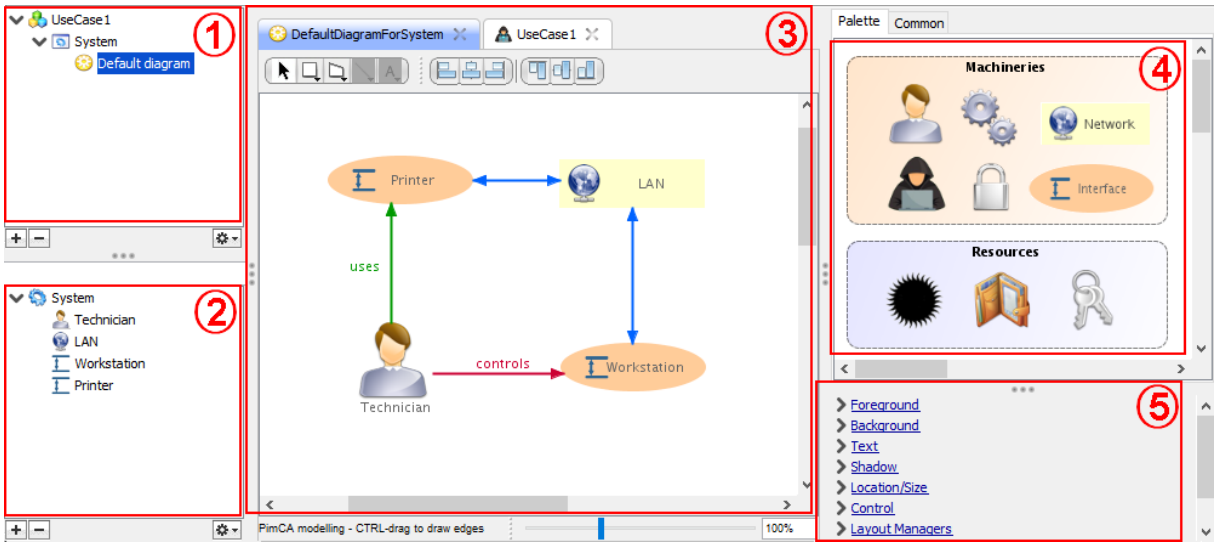


FIGURE 3.15 – Implémentation du langage Pimca avec la technologie OpenFlexo

1. L'explorateur de structure du projet contient le modèle XML Pimca et les diagrammes exportables dans divers formats graphiques (png, jpg, etc.) Chaque modèle peut être affiché dans l'éditeur de modèle en (3).
2. L'explorateur du modèle Pimca liste toutes les instances de *machinery*, *resource* et *relation* d'un modèle donné. L'utilisateur peut éditer les propriétés des instances directement à travers ce panel. Le panel peut également être utilisé pour instancier, supprimer, montrer ou cacher les concepts du langage.
3. L'éditeur de modèle montre la représentation graphique associée à chaque modèle Pimca. L'utilisateur peut explorer différents niveaux de granularité du modèle. Dans le cas d'un élément composite par exemple, l'utilisateur peut zoomer sur le nœud pour afficher les éléments qui le composent ainsi que les relations qui les lient. L'éditeur sert à créer et à éditer directement les modèles Pimca de manière graphique. Les instances du modèles sont interactives et permettent d'instancier les relations entre les éléments.
4. La palette de concepts présente les différents concepts du langage Pimca disponibles, ce qui permet leur instanciation rapide par un simple glisser-déposer.
5. L'inspecteur de modèle permet de paramétrer et d'éditer les propriétés graphiques des éléments individuels et du modèle global.

À partir de notre framework Pimca, il est désormais possible de créer et de maintenir des modèles Pimca. Le modèle Pimca évolue de manière itérative avec la compréhen-

sion que l'utilisateur a du système. Grâce à la technique de fédération de l'environnement OpenFlexo, l'utilisateur peut fédérer au modèle Pimca des documents pertinents à la compréhension du système.

Par exemple, si l'utilisateur a accès à un document de conception du système comme une spécification en SysML, il peut l'ajouter au projet et instancier des concepts Pimca directement reliés au document dans un nouveau modèle de système. Dans le cas où le système est partiellement connu, un raffinement itératif du modèle Pimca peut faire intervenir des éléments dont la nature est incertaine. La fédération de modèles permet de relier ces éléments à des sources d'informations que l'utilisateur peut inclure dans le projet. Autrement dit, elle assure la traçabilité des informations.

Le modèle architectural du projet, le modèle Pimca et le diagramme sont tous fédérés dans le framework Pimca. La technologie Openflexo permet de s'assurer de la synchronisation de ces différents modèles entre eux et conjointement avec des sources d'informations autres. L'approche de fédération permet de gérer l'interopérabilité avec des sources d'informations en entrée et des modèles d'analyse en sortie. La mise au point du framework Pimca fait l'objet d'une publication à la 6th International Conference on Information Systems Security and Privacy (ICISSP 2020) [92].

3.3.3 Extension Dynamique de Pimca

Grâce à l'outillage que nous avons mis en place, il est désormais possible de développer une approche opérationnelle d'APT s'appuyant sur le langage Pimca entièrement à partir de notre framework. Toutefois, l'état de l'art montre que si le langage Pimca constitue une base solide pour une telle approche, il ne couvre pas l'entièreté des besoins de l'Operational Design. En effet, le langage ne permet pas de modéliser l'évolution du système, ce qui pourrait permettre de simuler des scénarios d'attaque notamment. C'est pourquoi nous proposons d'étendre Pimca afin de supporter des aspects comportementaux pour les différents composants du système afin d'explorer divers scénarios d'évolution du système sous influence de l'APT.

De plus, notre approche a pour but de diagnostiquer un système sous attaque en produisant une stratégie d'APT. Il est donc nécessaire de concevoir un langage dynamique compatible avec cette problématique. La littérature montre que le domaine du Cyber Operational Design s'appuie en grande partie sur la qualité de l'analyse produite par le commandant en chef. Afin de pallier ce biais, nous choisissons de soutenir le développeur dans son analyse à l'aide d'outils d'analyse formelle. Ceci requiert un langage exécutable pour les analyses formelles.

Il existe de nombreuses méthodes pour modéliser le comportement dynamique d'un système pour des analyses formelles.

En particulier, les langages de commandes gardées basés sur les travaux de Dijkstra [22] permettent de résoudre cette problématique de manière compacte et adaptée au model-checking. Ils sont notamment utilisés dans des contextes de spécification et de validation de systèmes. La commande gardée (*guarded-command* ou GC) est la brique élémentaire d'un tel langage. Comme son nom l'indique, elle est classiquement composée de deux éléments :

— *Commande* : la commande est une déclaration qui prend effet lorsqu'elle est exécutée. Elle est aussi appelée action.

— *Garde* : la garde est une proposition booléenne qui conditionne l'exécution de la commande. La garde doit être vraie pour que la commande puisse s'exécuter. Dans le cas contraire, la commande gardée ne sera pas exécutée.

Ce formalisme a plusieurs avantages. En plus d'être compact, il est directement exécutable en spécifiant sa sémantique d'exécution.

3.3.3.1 Syntaxe & Sémantique d'Exécution

Cette section introduit la formalisation de DyPimca, présenté anciennement comme *Target System Modeling* (TSM) dans une de nos publications [93], l'extension dynamique du langage Pimca. DyPimca ajoute un langage de commandes gardées (GC) au méta-modèle Pimca afin de modéliser les comportements des différentes machineries du système. L'extension prend la forme d'un modèle comportemental de commandes gardées adjoint au modèle structurel Pimca. Le lien entre ces deux modèles se fait au niveau des machineries du modèle structurel par fédération de modèles. Ce lien est modélisé par une relation entre la méta-classe machinerie et la méta-classe unité d'exécution décrite dans la section suivante. En effet, les machineries sont des éléments Pimca pourvus de comportement, nous choisissons donc de spécifier ce comportement dans le modèle de GC pour chacun de ses éléments structurels. Ceci permet de proposer une modélisation modulaire, c'est-à-dire qu'il faut spécifier séparément le comportement de chaque machinerie. Cela demande de modéliser les interactions entre les différents composants à travers des canaux de synchronisation qui seront détaillés par la suite.

Conformément à la philosophie de l'ingénierie dirigée par les modèles, nous développons l'extension dynamique en commençant par sa **syntaxe abstraite**. Celle-ci est décrite dans un modèle formel.

Formellement, un modèle $\mathcal{M}$ de commandes gardées est un tuple $(\mathbb{V}, \mathcal{V}_\prime, \mathbb{A}, \mathbb{S})$ où $\mathbb{V}$ est une collection de variables, $\mathcal{V}_\prime$ est la valuation initiale des variables de $\mathbb{V}$. $\mathbb{A}$ est une collection de commandes gardées et $\mathbb{S}$ est une collection de canaux de synchronisation.

Par la suite, on notera $val_\mathbb{V}$ l'ensemble de définition de $\mathbb{V}$, c'est-à-dire l'ensemble des valeurs possibles pour la collection de variables $\mathbb{V}$. Pour les besoins du model-checking, les variables peuvent prendre un nombre restreint et fini de valeurs, c'est-à-dire que $val_\mathbb{V}$ est un ensemble fini. Il sera généralement défini par un ensemble d'énumérations pour chaque valeur de variables possible.

Une commande gardée est un tuple $(u : \mathbb{B}, s : \mathbb{S} \cup \{\varnothing\}, g : val_\mathbb{V} \to \mathbb{B}, c : val_\mathbb{V} \to val_\mathbb{V})$, où u est un booléen représentant l'urgence de la commande gardée, une mécanique de priorité que nous détaillerons par la suite, s est un canal de synchronisation (ou une absence de canal), g est une garde, c'est donc une expression booléenne qui s'exprime en fonction des variables de $\mathbb{V}$, c est une commande, elle représente l'action.

Un canal de synchronisation est lui-même un tuple $(d : \{\mathbf{in}, \mathbf{out}\}, id)$ où d désigne la direction du canal (entrant ou sortant), id est un identifiant unique.

La **syntaxe concrète** de la commande gardée dans la suite du manuscrit sera :

```
GC_name:
    urgent ?
    (channel ( ? | ! )) ?
    [guard] ? /
    (command ;) *
```

La notion d'urgence est ajoutée à certaines commandes gardées pour imposer un ordre d'exécution lorsque différentes commandes peuvent être exécutées. Intuitivement, si une commande gardée urgente dans $\mathbb{A}$ peut être exécutée, alors aucune commande gardée non-urgente ne peut être exécutée lors de ce pas. Si plusieurs commandes gardées urgentes peuvent être exécutées, n'importe laquelle d'entre elles peut être exécutée. On introduit donc le terme $hasUrgent_\mathbb{A} : val_\mathbb{V} \to \mathbb{B}$ pour dénoter une valuation ρ de l'ensemble des variables $val_\mathbb{V}$ pour laquelle il existe au moins une commande gardée urgente déclenchable, c'est-à-dire que sa garde est vraie pour la valuation ρ. Formellement :

$$\forall \rho \in val_{\mathbb{V}}, hasUrgent_{\mathbb{A}}(\rho) \quad \Leftrightarrow \quad \exists (u, \textbf{none}, g, \_) \in \mathbb{A}, u \wedge g(\rho)$$
$$\vee \quad \exists (u_1, (\textbf{out}, id), g_1, \_), (u_2, (\textbf{in}, id), g_2, \_) \in \mathbb{A}, (u_1 \vee u_2) \wedge g_1(\rho) \wedge g_2(\rho)$$

3.3.3.2 Unité d'Exécution

L'unité d'exécution est la partie du modèle de commandes gardées qui capture le comportement d'un élément du modèle structurel Pimca. Autrement dit, l'unité d'exécution représente le comportement d'une et une seule machinerie Pimca. En termes d'interopérabilité de modèles, il y a un lien direct entre le concept Pimca de machinerie et le concept DyPimca d'unité d'exécution.

Contrairement à un modèle de commandes gardées classique, nous proposons une compartimentation en différentes unités d'exécution dans un seul modèle. Ceci permet notamment une modélisation modulaire, c'est-à-dire élément par élément, ce qui donne une certaine flexibilité dans le modèle.

Cependant, cette modélisation impose une contrainte puisqu'il nécessaire de modéliser les interactions entre différentes unités d'exécution. Pour ce faire, nous introduisons la notion de canal de synchronisation. Une unité d'exécution utilise un canal de synchronisation pour interagir avec une autre unité d'exécution. Un canal de synchronisation s est défini dans $\mathbb{S}$ et peut intervenir dans la définition d'une commande gardée sous deux formes : en émission (dénotée par un **!**) ou en réception (dénotée par un **?**)

La sémantique du canal de synchronisation nécessite qu'une commande gardée qui utilise un canal de synchronisation en émission ne peut s'exécuter que dans le cas où sa garde est vérifiée et où il existe au moins une commande gardée en réception sur ce même canal dont la garde est également vérifiée. De même, une commande gardée qui utilise un canal de synchronisation en réception ne peut s'exécuter que dans le cas où sa garde est vérifiée et où il existe au moins une commande gardée en émission sur ce même canal dont la garde est également vérifiée. Lors de l'exécution, les commandes gardées d'émission et de réception sont exécutées en séquence émission-réception dans un même pas d'exécution.

Une unité d'exécution est un sous-modèle de commandes gardées $\mathcal{M}'$ tel que $\mathcal{M}' \cup \mathcal{M}$ inclus dans le modèle global. C'est donc un tuple $(\mathbb{V}', \mathbb{A}', \mathbb{S}')$ où $\mathbb{V}'$ est une sous-collection de variables de la collection $\mathbb{V}$ du modèle comportemental $\mathcal{M}$, $\mathbb{A}'$ est une sous-collection d'éléments de la collection de commandes gardées $\mathbb{A}$ et $\mathbb{S}'$ est une sous-collection d'éléments de la collection de canaux de synchronisation $\mathbb{S}$.

On note $\mathcal{P}(\mathbb{V})$ la partition de $\mathbb{V}$ qui contient chaque unité d'exécution du système. La partition des éléments de l'ensemble $\mathbb{V}$ est telle qu'aucune variable n'appartient simultanément à plusieurs unités d'exécution du système. Formellement on écrit : $\forall v \in \mathbb{V}, v \in \mathcal{M}' \Rightarrow \nexists \mathcal{M}'', \mathcal{M}' \neq \mathcal{M}'' \wedge v \in \mathcal{M}''$.

On note $\mathcal{P}(\mathbb{A})$ la partition de $\mathbb{A}$ qui contient chaque unité d'exécution du système. La partition des éléments de l'ensemble $\mathbb{A}$ est telle que chaque commande gardée se trouve dans une et une seule unité d'exécution, c'est-à-dire que chaque commande caractérise une action d'une machinerie en particulier. Formellement on écrit : $\forall a \in \mathbb{A}, \exists! \mathbb{A}' \in \mathcal{P}(\mathbb{A}), a \in \mathbb{A}'$.

Afin d'ajouter la notion de synchronisation au modèle comportemental, nous définissons une sémantique de fonctionnement qui prend en compte la notion d'urgence dans la Figure 3.16. La Figure 3.16 représente les règles d'inférence qui déterminent les commandes gardées du modèle qui peuvent être exécutées pour une valuation de variables ρ_1, c'est-à-dire une configuration donnée. Les règles d'inférence sont présentées sous la forme de prémisses et de résultats séparés par une barre horizontale. Si toutes les prémisses sont vérifiées, la règle d'inférence arrive à son résultat.

Si une commande gardée urgente qui n'utilise pas de canal de synchronisation a sa garde vérifiée, alors elle peut s'exécuter (règle $single_u$).

$$single_u : \frac{\forall (u, \mathbf{none}, g, c) \in \mathbb{A}, \forall\, \rho_1, \rho_2 \in val_{\mathbb{V}} \quad u \wedge g(\rho_1) \wedge c(\rho_1) = \rho_2}{\langle \|_{\mathbb{A}}, \rho_1 \rangle \rightarrow \rho_2}$$

$$single : \frac{\forall (u, \mathbf{none}, g, c) \in \mathbb{A}, \forall\, \rho_1, \rho_2 \in val_{\mathbb{V}} \quad \neg hasUrgent_{\mathbb{A}}(\rho_1) \wedge \neg u \wedge g(\rho_1) \wedge c(\rho_1) = \rho_2}{\langle \|_{\mathbb{A}}, \rho_1 \rangle \rightarrow \rho_2}$$

$$sync_u : \frac{\forall (u_1, (\mathbf{out}, id), g_1, c_1), (u_2, (\mathbf{in}, id), g_2, c2) \in \mathbb{A}, \forall\, \rho_1, \rho_2 \in val_{\mathbb{V}} \quad (u_1 \vee u_2) \wedge g_1(\rho_1) \wedge g_2(\rho_1) \wedge c_2(c_1(\rho_1)) = \rho_2}{\langle \|_{\mathbb{A}}, \rho_1 \rangle \rightarrow \rho_2}$$

$$sync : \frac{\forall (u_1, (\mathbf{out}, id), g_1, c_1), (u_2, (\mathbf{in}, id), g_2, c2) \in \mathbb{A}, \forall\, \rho_1, \rho_2 \in val_{\mathbb{V}} \quad \neg hasUrgent_{\mathbb{A}}(\rho_1) \wedge \neg(u_1 \vee u_2) \wedge g_1(\rho_1) \wedge g_2(\rho_1) \wedge c_2(c_1(\rho_1)) = \rho_2}{\langle \|_{\mathbb{A}}, \rho_1 \rangle \rightarrow \rho_2}$$

FIGURE 3.16 – Sémantique des commandes gardées synchronisées et urgentes

Si une commande gardée non-urgente qui n'utilise pas de canal de synchronisation a sa garde vérifiée, alors elle peut s'exécuter si aucune commande gardée urgente ne peut s'exécuter (règle $single$).

Si un couple de commandes gardées en émission et en réception possède au moins une commande urgente, alors le couple de commandes gardées est considéré comme urgent. Le couple peut s'exécuter si les deux gardes sont vérifiées (règle $sync_u$).

Si un couple de commandes gardées en émission et en réception ne possède pas de commande urgente, alors le couple de commandes gardées n'est pas considéré comme urgent. Il ne peut s'exécuter que s'il n'existe aucune commande urgente exécutable (règle $sync$).

Le méta-modèle de commandes gardées a pour but de capturer le comportement des différentes machineries du langage Pimca. De fait, le lien de fédération est établi entre les unités d'exécutions DyPimca et les machineries Pimca. Toutes les variables et commandes gardées des unités d'exécutions sont ainsi liées à des instances de machineries dans le modèle structurel.

Les variables référencées dans les gardes ou les commandes n'ont pas forcément à être liées au même concept Pimca. Cependant, si une variable d'un concept Pimca est référencée par une garde ou une commande d'un autre concept Pimca, ces deux concepts doivent être liés par une relation dans le modèle Pimca. De même un couple de GC synchronisé doit appartenir à deux concepts Pimca différents mais liés par une relation. Le framework Pimca permet de vérifier constamment la cohérence des modèles Pimca et DyPimca par fédération.

3.3.3.3 Implémentation

Pour ce qui est de l'implémentation du langage DyPimca, nous avons expérimenté avec plusieurs syntaxes concrètes à travers notre framework :

— Une syntaxe textuelle est utilisée pour illustrer les concepts dans ce manuscrit. Elle n'est pas directement exécutable, ce qui ne remplit pas les exigences de nos travaux.

— Une syntaxe en automate UPPAAL a été utilisée à des fins d'expériences préliminaires pour démontrer la faisabilité de notre approche (le cas d'étude traité dans le Chapitre 4 en utilisant cette syntaxe est disponible en Annexe A). Elle est directement exécutable dans UPPAAL.

— Une syntaxe concrète en Java exécutable est utilisée dans le cadre de la validation de notre approche.Cette syntaxe est celle que nous choisissons pour remplir l'exigence de l'exécutabilité du modèle. Une fois branchée avec le model-checker OBP2, elle permet de diagnostiquer un système et à terme de produire une approche opérationnelle d'APT sur le système.

Par la suite, nous utilisons la syntaxe concrète en Java dans nos analyses formelles. Le langage DyPimca nous permet de capturer le comportement du système. Conjointement avec Pimca, ces deux langages nous permettent de simuler des scénarios d'exécution nominale du système.

Le processus de modélisation de systèmes produit un modèle d'environnement opérationnel courant sur lequel peut ensuite s'appuyer le développement des modèles d'objectifs et d'obstacles. Comme souligné dans la Section 3.1, la phase de modélisation est un ensemble de processus qui s'alimentent mutuellement. Aussi il n'est pas à exclure que le concepteur doive faire évoluer le modèle de systèmes avec des besoins identifiés dans les autres processus de modélisation. L'approche par fédération permet d'assurer la synchronisation du modèle de système avec les autres modèles utilisés.

3.4 Modélisation d'objectifs

Une fois le système modélisé dans son aspect structurel et dans son aspect comportemental nominal, l'APT a modélisé le système tel qu'elle l'imagine dans le but de l'attaquer. Elle a défini l'ensemble des éléments du système ainsi que leur comportement pris en compte dans les différents scénarios d'attaque qu'elle développera. Pour développer les attaques contre le système, conformément à la méthodologie d'Operational Design, l'APT explicite son environnement opérationnel souhaité. L'environnement opérationnel souhaité est un état du système dans lequel les objectifs de l'APT sont remplis. Par exemple, l'APT cherche à violer la propriété d'intégrité d'une variable dans le système. Il peut donc s'agir d'un état dans lequel un élément du système pouvant éditer cette variable est directement sous le contrôle de l'APT.

3.4.1 Besoins

L'environnement opérationnel souhaité peut se modéliser en termes d'objectifs à atteindre pour remplir la mission. Les objectifs inférés lors de la phase de spécification pointent vers des éléments inférés ou l'environnement inféré tout entier. L'APT doit modéliser les objectifs opérationnels en lien direct avec ces concepts inférés. Ceci requiert de **maintenir les liens** avec le modèle de données tout au long de la démarche.

Ce faisant, il est nécessaire de formuler un modèle d'objectifs **compatible** avec le modèle de système. En effet, le modèle de système est le point d'appui pour l'ensemble de la réflexion du concepteur. Il faut donc que le modèle capture les objectifs de la mission en faisant référence aux éléments modélisés dans le système.

Enfin, puisque le concepteur cherche à obtenir une stratégie opérationnelle concrète à l'issue de la phase d'analyse de l'approche, il est nécessaire d'employer un formalisme d'objectifs adéquat. Pour parvenir à produire automatiquement une stratégie opérationnelle, nous avons choisi les outils d'analyse formelle dans la mesure où nous sommes capables de gérer l'explosion combinatoire, l'obstacle principal à l'application des méthodes formelles à l'industrie. Nous détaillons cet aspect de la contribution dans la Section 3.6. C'est pourquoi le modèle d'objectifs doit suivre un formalisme qui se prête aux analyses formelles, c'est-à-dire une **modélisation formelle**.

Dans l'ensemble, le modèle d'objectifs doit répondre à trois besoins : la **compatibilité avec le modèle de système**, la **synchronisation avec le modèle de données** et le besoin de **modélisation formelle** pour les besoins de la phase d'analyse.

3.4.2 Propriétés de sécurité

En premier lieu, il faut vérifier que les objectifs de la mission peuvent être capturés dans le modèle de système. Cette étape de validation informelle valide ou non le fait que le modèle de système est adéquat pour la mission. Le modèle de données de la phase de spécification permet cette vérification conjointement avec le modèle de système. Si les éléments inférés concernés par les objectifs inférés sont tous présents dans le modèle de système, alors le modèle de système permet de concevoir le modèle d'objectifs. Dans le cas où le modèle de système ne correspond pas, il est nécessaire de raffiner le modèle de système en modélisant l'environnement opérationnel courant de nouveau avec de nouvelles contraintes exprimées lors de ce processus. Lors de cette validation informelle, le concepteur peut d'ores et déjà **se synchroniser avec le modèle de données** en utilisant les liens de fédérations décrits dans la Section 3.2.1. Ce faisant, il crée les liens vers les objectifs opérationnels qui seront produits dans cette étape.

	Point de vue de l'APT	Point de vue du système cible
Propriété vérifiée	Objectif non-atteignable	Système protégé contre l'attaque
Propriété non-vérifiée	Objectif atteignable	Système vulnérable à l'attaque

Table 3.3 – Propriétés de sécurité : deux points de vue (inspiré de [45])

Du point de vue du système cible, les objectifs de l'APT représentent des événements redoutés pour le système définis dans la méthodologie EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité) [29]. Aussi, il est possible de traduire ces objectifs en propriétés de sécurité [45]. Chaque propriété représente un objectif pour l'attaquant et un événement redouté pour le système cible. Le Tableau 3.3 présente les deux points de vue opposés des propriétés de sécurité. Ainsi, dans la phase d'analyse, déterminer si le système peut vérifier ces propriétés avec les actions de l'attaquant permet de diagnostiquer si l'attaque permet d'atteindre les objectifs de la mission ou non.

Pour écrire les propriétés de sécurité correspondant aux objectifs de la mission, le concepteur utilise les objectifs inférés au préalable dans la phase de spécification. Les éléments inférés du système concerné par les objectifs inférés sont reliés par fédération aux éléments opérationnels du modèle de système. Ceux-ci capturent des états ou des comportements désirés par l'attaquant sous la forme de variables. Les propriétés de sécurité peuvent donc s'écrire comme des assertions logiques sur ces variables. La vérification d'une propriété de sécurité assure que le système ne permet pas à l'APT d'atteindre son objectif dans l'état actuel. Cela signifie que le système est résistant à l'attaque de l'APT. L'échec de la vérification d'une propriété de sécurité montre qu'il existe un chemin d'attaque permettant à l'APT d'atteindre son objectif. Cela montre que le système peut être attaqué avec les possibilités que l'attaquant se donne.

Ainsi, la modélisation d'objectifs consiste à formuler les objectifs opérationnels de la mission en fonction des objectifs inférés au préalable et de l'environnement opérationnel. Ce faisant, la fédération de modèles permet de garder un **lien ténu** entre les **objectifs inférés et opérationnels** d'une part et entre **l'environnement et les objectifs opérationnels** d'autre part.

3.4.3 Logique temporelle linéaire

Les objectifs de la mission doivent être traduits par des propriétés de sécurité correspondant aux objectifs de la mission. Le concepteur peut ensuite vérifier automatiquement la satisfaction des objectifs de la mission lors de la phase d'analyse.

Dans le cas d'un système ciblé modélisé avec le langage DyPimca, les objectifs de la mission peuvent être capturés par des propriétés sur les variables des éléments opérationnels du système. Ceci assure la **compatibilité** entre le modèle de système et d'objectifs. Ces propriétés s'expriment naturellement à l'aide de la logique temporelle linéaire ou LTL [27]. La logique temporelle linéaire est un **formalisme qui permet notamment les analyses formelles**.

Prédicats sur le système : À l'aide de sa compréhension du fonctionnement du système, le concepteur stipule des propriétés LTL utilisant les variables d'état du système modélisé en langage DyPimca. Par exemple, si l'objectif est de stopper le fonctionnement d'un élément actif dans le système, le concepteur utilise une variable d'état qui capture l'état de fonctionnement de la machinerie correspondante, ce qui peut s'écrire en prédicat LTL de la manière suivante :

```
machineState = down
```

Si l'objectif est de parvenir à atteindre une ressource, la propriété correspondante est la variable d'état du système qui correspond à la ressource en question, ce qui peut s'écrire en prédicat LTL de la manière suivante :

```
reachedResource
```

Ces variables sont soumises à des prédicats pour capturer les propriétés voulues sur le système. Par exemple si l'objectif est de faire déborder un réservoir la variable qui capture le niveau de ce réservoir doit dépasser le niveau accepté par le réservoir, ce qui peut s'écrire en prédicat LTL de la manière suivante :

```
waterLevel > waterLimit
```

À l'aide de ces prédicats, le concepteur peut écrire les propriétés de sécurité correspondant aux objectifs de la mission en utilisant la logique temporelle linéaire. En LTL, l'opérateur temporel "□" (parfois noté "G" pour globalement) signifie "toujours". On peut donc écrire "□!(*prédicat*)" pour la propriété de sécurité signifiant "Jamais (prédicat)". En effet, comme expliqué dans le Tableau 3.3, les propriétés de sécurité sont des assertions qui montrent que le système est robuste vis-à-vis d'un état à éviter pour le système. Du point de vue de l'attaquant, c'est précisément cet état que la mission cherche à atteindre. Aussi l'objectif de la mission consiste à violer la propriété de sécurité, c'est-à-dire à trouver un chemin d'attaque qui permet de mettre le système dans son état redouté. Un exemple de chemin d'attaque est détaillé dans le Chapitre 4.1.4.

Si le modèle de système ne capture pas de variables correspondant aux objectifs, il faut raffiner le système pour inclure ces variables. Par exemple, dans le cas du réservoir, il faut implémenter un comportement interne au réservoir qui fait varier le niveau d'eau. Ceci met en évidence une insuffisance dans la compréhension du système ciblé compte tenu des objectifs de l'environnement opérationnel souhaité. Ce problème demande donc de raffiner la compréhension de l'environnement opérationnel courant par l'APT.

La Figure 3.17 montre comment la fédération de modèles permet de relier les objectifs inférés et opérationnels. L'objectif inféré issu du modèle de données (en jaune) est l'instance de concept informel pointé par l'artefact de fédération "Objectif fédéré" (en bleu). L'artefact de fédération pointe deux instances d'éléments de modèles opérationnels, l'objectif opérationnel et le système opérationnel (en vert). En effet, l'objectif inféré indique quels éléments du système cible sont concernés par l'objectif en question. Ces éléments du système sont modélisés dans le système opérationnel comme des unités d'exécution. C'est à partir de variables issues de ces unités d'exécution que le concepteur va pouvoir formuler l'objectif sous forme opérationnelle, c'est-à-dire en termes de propriété de logique temporelle linéaire. Sur le schéma, cette propriété est représentée sous la forme d'un arbre syntaxique comprenant des opérateurs logiques temporels et des références vers les variables issues du système opérationnel.

La modélisation des objectifs a pour but de capturer les objectifs opérationnels de la mission à partir des objectifs inférés lors de la phase de spécification. Pour cela, nous employons le formalisme LTL pour formuler les objectifs en termes de propriétés sur les variables des éléments du modèle de système. Ce formalisme permet d'**assurer l'automatisation de la phase d'analyse**. L'ensemble des objectifs opérationnels sont **liés aux autres modèles à travers notre environnement de fédération** explicité dans la Section 3.2. Ainsi l'approche peut subir plusieurs itérations successives jusqu'à obtenir des résultats satisfaisants tout en maintenant les liens entre les différents concepts inférés et opérationnels.

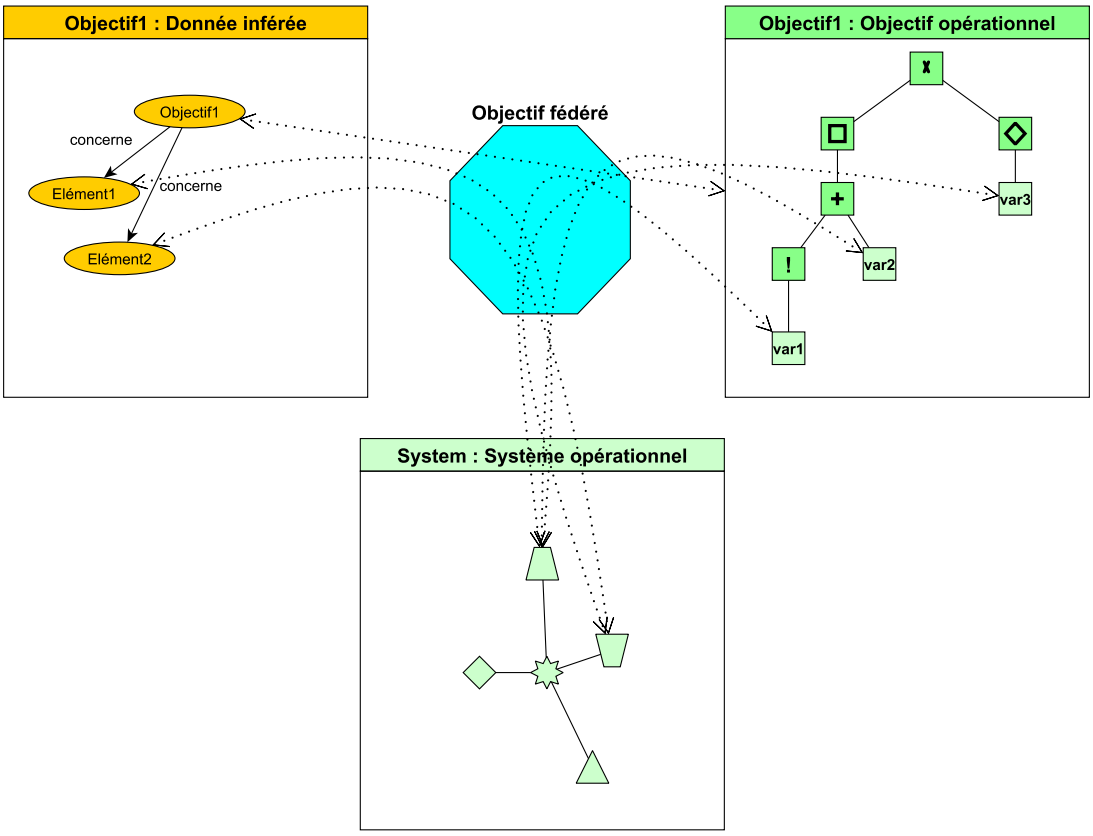


FIGURE 3.17 – Objectif fédéré

On note que ces propriétés n'ont pas nécessairement à être vérifiées dans l'environnement opérationnel courant. En effet, elles représentent les objectifs de la menace persistante avancée sur le système. Il est donc attendu que ces objectifs entrent en conflit avec le comportement nominal du système. L'attaquant va devoir développer les moyens de parvenir à ses objectifs en identifiant les obstacles posés par le système au préalable dans le processus suivant, la modélisation des obstacles.

3.5 Modélisation des obstacles

Une fois le système cible et les objectifs de la mission modélisés d'un point de vue opérationnel, l'APT modélise les obstacles qui l'empêcheront de réaliser sa mission. En effet, la stratégie d'attaque de l'APT consiste à passer outre ces différents obstacles au moyen d'opportunités que le concepteur envisage. Lors de cette étape, l'APT identifie les obstacles au sein du système et formule un certain nombre d'opportunités qui lui permettraient de parvenir à ses objectifs. Ainsi lors de la phase d'analyse, l'APT pourra formuler une approche opérationnelle pour remplir la mission sous la forme d'un ensemble d'opportunités à exploiter.

3.5.1 Besoins

Modéliser les obstacles du système cible et formuler des opportunités pour contourner ces obstacles est un des processus de la démarche d'Operational Design. En effet, ce sont ces opportunités qui constituent les briques élémentaires de la stratégie d'attaque de l'APT.

Toutefois, il est important de noter le fossé qui existe entre la stratégie initiale et la réalité du terrain. Les opportunités envisagées par le concepteur sont certes appuyées par la phase de spécification mais elles ne reflètent que la compréhension que le concepteur a du système cible. Aussi, il est probable que certaines opportunités ne soient pas réalisables sur le terrain. C'est pourquoi il est nécessaire de pouvoir raffiner la modélisation d'obstacles en continu pour pouvoir prendre en compte la réalité du terrain. Ceci met en lumière un besoin d'**exprimer les obstacles et les opportunités en continu**.

De plus, pour pouvoir formuler une stratégie efficace, le concepteur doit identifier un maximum d'obstacles à la réalisation de la mission. Ces obstacles peuvent être de natures très diverses comme par exemple des éléments humains ou des systèmes d'exploitation. C'est pourquoi il est nécessaire de pouvoir **modéliser un maximum d'obstacles de différentes natures et des opportunités variées**.

Dans l'ensemble, le modèle d'obstacles doit répondre à deux besoins : la **possibilité de modéliser les obstacles et les opportunités en continu** et la **prise en compte d'obstacles et d'opportunités de différentes natures**.

3.5.2 Concepts

Modéliser les éléments problématiques du système en Pimca revient à valider le fait que les préoccupations de la mission correspondent au modèle de système proposé. À l'issue de cette étape, le concepteur détermine si le modèle de système capture exactement les structures et les comportements pertinents pour la mission. Dans le cas contraire, il est nécessaire de raffiner le modèle d'environnement opérationnel courant. Cette étape de validation ne requiert pas d'outillage particulier au delà de la modélisation Pimca.

Ensuite, pour modéliser les attaques envisagées sur le système, en particulier sur ces éléments problématiques, le concepteur stipule les opportunités qu'il souhaite faire intervenir, conformément à la phase de spécification et au modèle de données. Un modèle d'obstacles capture ces éléments en termes de correspondance entre des éléments problématiques et des attaques qui leur sont associées. Une attaque possède des pré-conditions concernant le système, qui doivent être remplies pour permettre son exécution. Elle possède également des post-conditions qui prennent la forme de comportements induits chez le système.

3.5.3 Implémentation & Discussion

Dans le cadre de notre framework Pimca, nous avons choisi de nous restreindre aux opportunités de l'attaquant qui entraînent une modification de comportement du système sans en impacter la structure. Les comportements des éléments présentant des opportunités sont modifiés de sorte que leur comportement nominal soit conservé jusqu'à ce que l'attaquant exploite les opportunités correspondantes.

Les opportunités de l'attaquant sont capturées par un composant abstrait externe au système appelé "l'attaquant". Celui-ci est exprimé dans le formalisme de DyPimca et permet de modéliser la menace persistante avancée qui exploite ses différentes opportunités au niveau du système pour accomplir sa mission.

Avec le langage Pimca et son haut niveau d'abstraction, le concepteur a la possibilité de formuler des opportunités abstraites sur les composants dont la nature est incertaine. Par exemple, si le système d'exploitation d'un composant est inconnu, le concepteur peut tout de même formuler une opportunité consistant en l'exploitation d'une faille supposée sur le système d'exploitation inconnu, en l'attente de reconnaissance ultérieure.

Possibilités d'interaction : L'APT capture ses possibilités d'interactions dans le système à l'aide d'ajout, de modification de commandes gardées dans la modélisation comportementale du système.

Ce faisant, il est offert la possibilité de capturer le fait d'utiliser une capacité d'interaction avec des éléments du système via la matérialisation dans le modèle de variables spécifiques à la capacité d'attaquer ces éléments. Par exemple, si l'attaquant possède les capacités d'interagir avec trois éléments différents du système, le concepteur ajoute dans son modèle trois variables booléennes qui modélisent chaque capacité d'interaction en plus des modifications et ajouts induits par l'ajout d'un nouveau comportement d'attaque de la manière suivante :

$$\mathtt{attack}_1,\ \mathtt{attack}_2,\ \mathtt{attack}_3$$

Chacune des variables booléennes est initialisée à *faux*. Elles changent d'état si et seulement si l'attaquant utilise la capacité d'interaction en question. Le comportement d'attaque est contrôlé par une unité d'exécution spécifique contenant ces variables d'états et des commandes gardées déclenchant l'attaque. Ces commandes gardées déclenchent le changement d'état des variables et le comportement du système tel qu'il est induit dans l'attaque. Un exemple concret d'implémentation d'attaque est présenté dans la Section 4.1.3.

Par la suite l'attaquant pourra faire jouer les opportunités qu'il s'est créées pour tenter d'atteindre ses objectifs. Dans le cadre de nos travaux, cette phase de modélisation permet d'injecter des opportunités dans le modèle de manière textuelle, toutefois l'implémentation reste basique.

Difficultés d'implémentation : La modélisation d'obstacles et d'opportunités repose sur une expertise métier poussée dans chaque domaine envisageable pour conduire une attaque. Ces domaines sont variés et incluent l'ingénierie sociale, les communications réseaux et l'architecture physique du système. Ces différents domaines nécessitent de faire appel à un expert métier dédié, ce qui rend la modélisation d'obstacles difficile à généraliser. En effet, pour parvenir à accomplir ses objectifs, l'APT cherche à induire différents comportements dans le système. Ces comportements sont particulièrement spécifiques de l'objectif et du système en question. C'est pourquoi il est difficile de proposer un modèle générique d'opportunités. Les travaux de Drouot [24] mettent en évidence cette difficulté et proposent un début de résolution en définissant le concept de vulnérabilité. La vulnérabilité est une faiblesse qui appartient au système et requiert un certain niveau de connaissance ou de capacité pour être exploitée. L'opportunité devient la combinaison d'une vulnérabilité et d'un attaquant ayant les moyens et l'intention de l'exploiter Cette approche

montre que la modélisation d'obstacles et d'opportunités est possible mais qu'elle nécessite de fournir un effort de modélisation supplémentaire.

La modélisation des obstacles a pour but de capturer les obstacles opérationnels à la réalisation de la mission et les opportunités envisagées par le concepteur pour résoudre ces problèmes. Ce processus s'appuie sur la phase de spécification, en particulier sur les obstacles et les opportunités inférés. Pour cela, nous nous appuyons sur la fédération de modèles pour répondre aux besoins d'**expression des obstacles et des opportunités en continu** et de **diversité des obstacles**.

Toutefois, il est encore nécessaire de fournir un effort de formalisation pour parvenir à un modèle d'obstacles et d'opportunités complet. Dans la suite de ce manuscrit et notamment sur le cas d'étude, nous utiliserons un modèle d'obstacles textuel sommaire dans un but d'illustration.

Une fois l'ensemble des obstacles et des opportunités modélisé, le concepteur peut passer à la phase d'analyse et formuler une approche opérationnelle pour remplir sa mission.

3.6 Analyse formelle de la stratégie

Une fois la phase de modélisation terminée, l'APT peut passer à la phase d'analyse. Lors de cette phase, l'APT utilise les différents modèles de systèmes, d'objectifs et d'obstacles pour tenter de produire une stratégie opérationnelle, c'est-à-dire un chemin d'attaque qui lui permettra d'accomplir ses objectifs. Tout système est vulnérable, mais le concepteur détermine si le système est attaquable avec les opportunités qu'il se donne.

3.6.1 Besoins

Il est important de rappeler que l'approche que nous proposons est une approche à haut niveau d'abstraction qui cherche à produire une stratégie prévisionnelle d'attaque globale pour l'APT. En effet, les différents modèles produits lors de cette approche se basent sur la compréhension que l'APT a de la réalité du terrain. Aussi, il est attendu que la stratégie produite par cette approche soit amenée à évoluer à mesure que l'APT progresse sur le système réel. Comme la stratégie est amenée à évoluer rapidement et se base sur des hypothèses du concepteur, il est inutile de chercher à modéliser le plus précisément possible les détails de la stratégie. Les priorités de notre approche sont **la rapidité et la fiabilité de diagnostic**. C'est pourquoi il est nécessaire de proposer une technique de diagnostic qui requiert le minimum de décision de la part du concepteur et qui reste déterministe, c'est-à-dire une technique qui produit toujours le même résultat pour un même jeu de données d'entrée. Nous choisissons donc d'adopter une **technique d'analyse automatique formelle** présentée dans la Section 2.4.3.

Afin de **capitaliser sur nos travaux de modélisation**, il semble judicieux de choisir une technique d'analyse s'appuyant sur nos modèles de systèmes, d'objectifs et d'obstacles. Parmi celles qui ont été présentées dans la Section 2.4.3, notre choix s'oriente vers le *model checking*. De plus, on constate que l'échelle des applications industrielles du *model checking* est plus proche de l'échelle des préoccupations de l'APT de notre approche puisque le concepteur cherche à perturber le comportement du système pour parvenir à ses objectifs. Il cherche donc à mettre en *erreur* le système, c'est à dire atteindre un état redouté du point de vue du système mais où l'objectif de l'attaquant est atteint. C'est pourquoi la phase d'analyse utilise le *model checking* pour déterminer **automatiquement** une stratégie d'attaque en **capitalisant sur les travaux de modélisation**.

3.6.2 Model Checking

Le *model checking* est une technique de vérification automatique et exhaustive utilisant un modèle d'automate fini pour vérifier des propriétés, ce qui s'adapte parfaitement à notre modèle de système en commandes gardées et à notre modèle d'objectifs en propriétés LTL. Il correspond donc à nos besoins de la phase d'analyse. Cependant, le *model checking* comporte quelques inconvénients. Tout d'abord, il n'est **applicable que sur un système dit "borné"**, c'est-à-dire que le comportement du système doit comprendre un nombre fini d'états différents afin de mettre en application la technique de vérification. Dans la réalité, les systèmes complexes qui sont les cibles des APT ne conviennent pas forcément tant leur espaces d'états sont vastes et parfois difficiles à caractériser. De plus, le problème principal auquel se heurte le *model checking* est son passage à l'échelle sur de vastes espaces d'états pouvant mener à une explosion combinatoire. En effet, cette technique repose sur l'exploration exhaustive de l'espace d'états du système. Cependant, ces problématiques n'affectent pas notre approche car le système que le concepteur aborde, est fini par construction du modèle. En effet, l'APT modélise le système cible en limitant le nombre d'éléments nécessaire à sa compréhension et à sa connaissance du système. Les objectifs de l'APT reste de

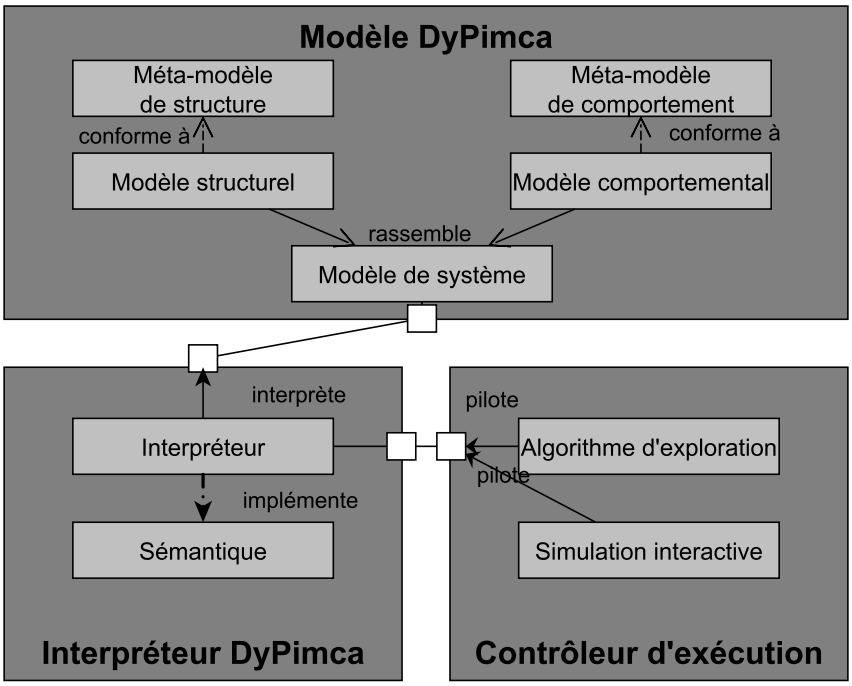


FIGURE 3.18 – Schéma conceptuel de l'architecture de *model-checking* de notre approche

conduire une analyse au niveau système sans prendre en compte des détails de spécialité métier comme le réseau, les systèmes d"exploitation, etc. Les APT cherchent des scénarios d'attaque sur tout le périmètre système sans prendre en compte des obstacles techniques-scientifiques spécifiques, au moins dans un premier temps.

Le risque d'explosion combinatoire reste contenu par la taille des modèles système et les analyses qui y sont effectuées.

L'architecture de *model-checking* que nous utilisons est présentée de manière conceptuelle dans la Figure 3.18. Le modèle que nous vérifions est le modèle de système DyPimca conçu dans la phase de modélisation. Le modèle de système DyPimca est conforme à la structure du système décrite dans le modèle Pimca qui est lui-même conforme à la syntaxe abstraite définie dans le méta-modèle Pimca. Le modèle de système contient le comportement décrit dans le modèle comportemental qui est lui-même conforme à la syntaxe abstraite définie dans le méta-modèle de commandes gardées.

Contrairement à une approche de *model-checking* traditionnelle dans laquelle le model-checker est conçu pour un langage particulier, le model-checker OBP2 [1] est une solution logicielle modulaire agnostique au langage vérifié. OBP2 fédère une sémantique d'exécution dédiée au langage et un ensemble d'artefacts de *model-checking* agnostiques et réutilisables. La sémantique d'exécution prend la forme d'un plugin à développer pour le langage à vérifier. OBP2 utilise une architecture s'articulant autour d'un interpréteur de modèles et d'un contrôleur d'exécution.

L'interpréteur de modèles encode une implémentation de la sémantique du langage DyPimca. Cet interpréteur manipule une instance du modèle de système. L'état courant du système est capturé dans l'ensemble des variables d'état de l'instance du modèle manipulée. Á partir de l'état courant, l'interpréteur calcule les différentes transitions déclenchables en évaluant les gardes de l'ensemble des commandes gardées. Ensuite il revient au contrôleur d'exécution de déterminer quelle transition déclencher.

1. http://www.obpcdl.org/

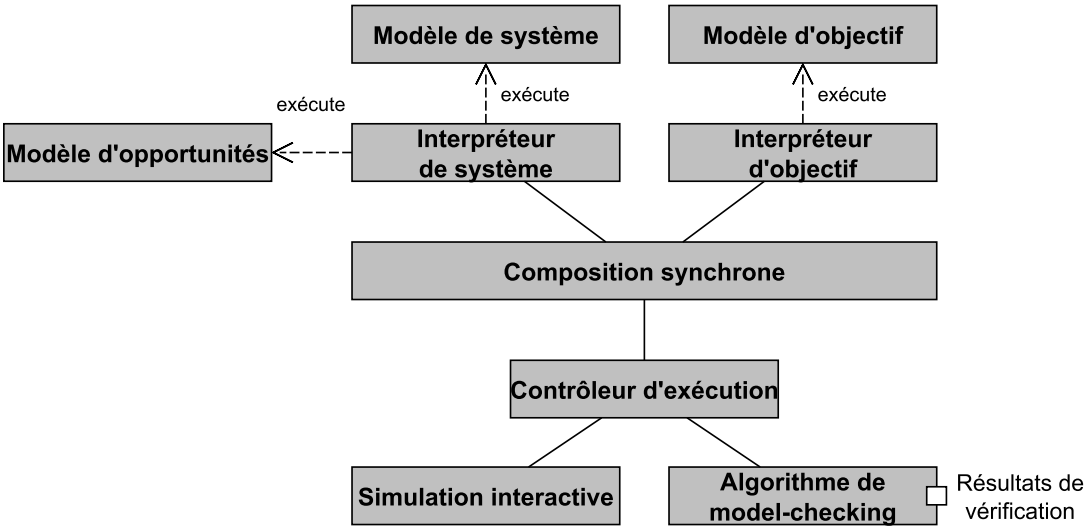


FIGURE 3.19 – Architecture de *model-checking* utilisée

Le contrôleur d'exécution est une solution originale d'OBP2 pour pouvoir fédérer différentes activités utilisant le même interpréteur et la même sémantique d'exécution [34]. Dans notre cas, les deux activités que nous proposons sont :

— la simulation interactive pas à pas où l'utilisateur choisit les transitions à déclencher,

— l'algorithme d'exploration exhaustive de l'espace d'états pour les besoins d'automatisation du *model-checking*.

L'interface d'OBP2 permet de simuler pas à pas le comportement du système. OBP2 garde en mémoire la configuration courante ainsi que la trace d'exécution, ce qui permet de recharger une configuration antérieure. L'interface permet également de visualiser et de déclencher les transitions tirables depuis la configuration courante pour une exploration manuelle. Par ailleurs, OBP2 permet de vérifier des propriétés LTL avec GPSL [6]. Lors de l'exploration automatique impliquant la composition d'automates de Büchi, le model-checker explore l'espace d'états du système et indique si une propriété est valide. Dans le cas contraire, OBP2 retourne un contre-exemple qui viole la propriété, sous forme de trace d'exécution.

Le model-checker OBP2 requiert les fonctionnalités suivantes au plugin du langage, DyPimca dans notre cas :

— Charger le modèle.

— Charger l'ensemble des états initiaux.

— Définir la fonction de transition qui détermine les états du système directement atteignable à partir d'un état source.

— Évaluer les propositions atomiques pour générer un nouvel état du système.

Dans notre cas, nous utilisons un modèle de commandes gardées dont la sémantique d'exécution a été définie au préalable dans la Section 3.3.3. Nous avons donc fourni toutes les fonctionnalités requises pour le plugin OBP2.

3.6.3 Scénario d'Attaque

La Figure 3.19 présente l'architecture concrète de model-checking dans notre approche. L'interpréteur de modèle système exécute la version du système enrichie des opportunités de l'APT. Conjointement, OBP2 utilise un moteur d'exécution de la propriété à vérifier sur

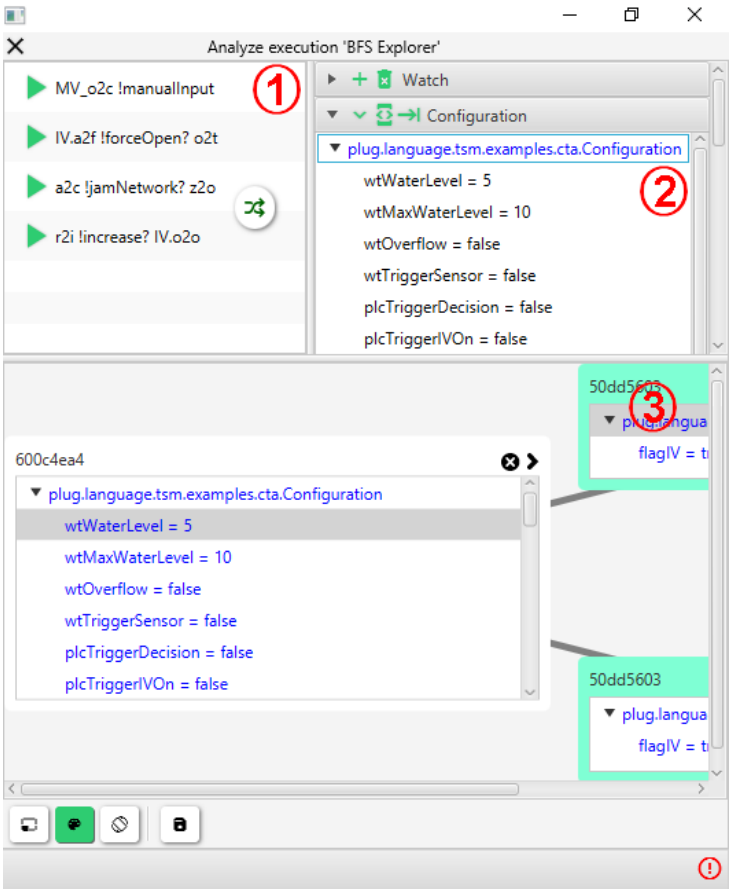


FIGURE 3.20 – Interface graphique d'OBP2

le modèle. L'exécution globale est une composition synchrone d'une instance de modèle de système et d'une propriété sous la forme d'un automate de Büchi. Par la suite, le contrôleur d'exécution permet à l'utilisateur de dérouler des scénarios d'attaque à la main ou à un algorithme d'effectuer une analyse exhaustive.

Objectifs globaux : La vérification des objectifs de la mission requiert de combiner les opportunités de l'APT avec le modèle d'objectifs. Les prédicats capturant les objectifs de l'APT vus en Section 3.4.3 et les variables booléennes de l'attaquant vues en Section 3.5.3 permettent d'écrire des formules booléennes capturant les propriétés à vérifier par le *model-checking*. Le model-checker vérifie si chaque combinaison de capacités d'interactions résulte en l'accomplissement d'un des objectifs souhaités.

Par exemple pour la propriété qui vérifie si l'APT peut atteindre l'objectif 1 capturé par le prédicat *predicate1* en faisant usage des opportunités capturées par les booléens $attack_1$, ... , $attack_n$ on écrit l'assertion suivante :

$$\mathrm{attack}_1 \wedge \ldots \wedge \mathrm{attack}_n \Rightarrow \Diamond \mathrm{predicate1}$$

Informellement on peut lire cette assertion comme : *"Si l'APT utilise les opportunités $attack_1$, ..., $attack_n$, éventuellement l'objectif est atteint"*. Toutefois, comme définie en Section 3.4.2, la propriété de sécurité que nous voulons vérifier du point de vue de l'APT est la négation de cette assertion. En effet, la propriété de sécurité vérifiée implique que l'objectif de l'APT n'est pas atteignable alors que la propriété violée indique qu'il existe un chemin d'attaque capturé par un scénario contre-exemple.

On écrit donc la propriété capturant l'objectif 1 de la manière suivante :

Objectif 1
$\square$ ((!attack$_{n+1}$ $\wedge$ ...) $\Rightarrow$!predicate1)

Cette propriété signifie en langage courant : *"Tant que l'APT n'emploie pas les opportunités attack$_1$, ..., attack$_n$, l'objectif 1 n'est pas atteint"*. Ainsi, si le model-checker trouve un scénario d'exécution dans lequel cette propriété est violée, alors il s'agit d'un scénario dans lequel les opportunités attack$_1$, ..., attack$_n$ permettent d'accomplir l'objectif 1. En revanche, si cette propriété est vérifiée alors cela signifie qu'il n'existe pas de chemin d'attaque permettant d'atteindre l'objectif 1 avec ces opportunités. Un exemple concret sur notre cas d'étude de l'écriture de ces propriétés est disponible en Section 4.1.5.

OBP2 affiche les résultats du *model-checking* sous la forme de trace. La trace permet de consulter les différents états parcourus par l'exécution. La Figure 3.20 présente l'interface graphique d'OBP2 sur un sous-ensemble de la trace.

1. Les transitions tirables sont affichées dans ce panel. Le label des transitions est précisé dans le langage DyPimca.
2. La configuration courante est représentée sous la forme des valeurs des différentes variables d'état. Ces variables sont définies dans le langage DyPimca.
3. L'arborescence de ce panel représente l'ensemble des états parcourus jusqu'alors. Il est possible de charger n'importe quel état déjà atteint comme configuration courante et de poursuivre la simulation à partir de cet état.

Le *model checking* détermine également le cas où les opportunités conçues par l'APT ne suffisent pas. Le concepteur doit alors ajouter des opportunités en repassant en phase de modélisation.

3.7 Conclusion

Les menaces persistantes avancées possèdent un mode opératoire atypique. Elles sont connues pour leurs campagnes de longue durée employant des techniques sophistiquées, des ressources nombreuses et des tactiques de dissimulation. Pour mieux comprendre leur mode opératoire, nous nous intéressons précisément aux mécanismes de conception d'une attaque des APT. Pour cela, nous faisons le lien avec les opérations militaires plus classiques, car les forces armées possèdent un mode opératoire très similaire.

Dans ce chapitre, nous avons détaillé les contributions théoriques permettant de modéliser le processus d'élaboration de la stratégie des APT. Dans un premier temps, nous avons expliqué la méthodologie de notre approche dans la Section 3.1. L'approche applique la méthodologie militaire de l'**Operational Design** dans le cadre des APT. Elle se décompose en trois phases : spécification, modélisation et analyse. Dans la première, la **phase de spécification**, détaillée dans la Section 3.2, le concepteur agrège et organise les données pertinentes pour la mission à accomplir. Ce faisant, il construit un modèle de données inférées. Dans la deuxième, la **phase de modélisation**, le concepteur modélise le système cible (détaillé dans la Section 3.3), les objectifs de la mission (détaillés dans la Section 3.4), les obstacles et opportunités (détaillés dans la Section 3.5). Cette phase met en jeu différents modèles dont on gère l'interopérabilité grâce à la fédération de modèles. De cette façon, chaque modèle répond précisément aux besoins de son processus, ce qui permet de proposer une implémentation concrète de l'approche. Dans la troisième phase, la **phase d'analyse**, détaillée dans la Section 3.6, le concepteur analyse ses modèles afin de déterminer une stratégie d'attaque, ou approche opérationnelle, pour remplir la mission. Pour cela, nous proposons d'utiliser le *Model Checking*, qui permet de capitaliser sur les efforts de modélisation de la phase précédente tout en garantissant un résultat fiable et automatique.

L'ensemble de ces contributions permettent de répondre à notre problématique de **modélisation de la phase de reconnaissance et d'armement des APT en appliquant la méthodologie de l'Operational Design** d'un point de vue théorique. Dans le Chapitre 4, nous verrons une application concrète de notre méthodologie afin de valider notre méthodologie sur un cas d'étude.

Chapitre 4

Validation de la méthodologie

Sommaire

Le Chapitre 3 présente la méthodologie proposée pour comprendre le développement de stratégie des menaces persistantes avancées.

Dans ce chapitre, nous montrons comment appliquer à un cas d'étude concret la méthodologie adaptée de l'*Operational Design* que nous avons développée. Étant donnée la sensibilité des cas d'étude réels utilisés par les forces armées, nous ne pouvons présenter ces travaux. C'est pourquoi l'étude que nous proposons concerne un système de station de pompage d'eau utilisé dans la littérature [86].

Dans un premier temps, la Section 4.1 introduit et traite le cas d'étude de la station de pompage. Nous suivons les différents processus de la méthodologie afin de remplir une mission comprenant deux objectifs. Après une phase de spécification, de modélisation puis d'analyse, notre approche produit la stratégie opérationnelle souhaitée. Dans un second temps, la Section 4.2 discute des résultats tirés du cas d'étude et permet de faire le bilan de notre approche vis-à-vis des différentes exigences explicitées dans le Chapitre 2.

4.1 Cas d'étude : Attaque d'une station de pompage d'eau

Cette section illustre la méthodologie de l'Operational Design appliquée au cas d'étude d'une station de pompage cible d'une APT. L'attaquant a plusieurs objectifs : mettre hors-service la station et ne pas se faire repérer lors de ses opérations. Nous montrons comment un concepteur d'attaque utilise notre méthodologie en suivant les phases de spécification, de modélisation et d'analyse.

Dans ce cas d'étude, nous traiterons successivement les objectifs de la mission. Cette manière de procéder est optionnelle mais elle permet de montrer comment développer plusieurs itérations de la méthodologie. De plus, elle montre comment traiter simplement une mission comportant plusieurs objectifs.

Conformément à la méthodologie que nous proposons, cette section commence par la spécification de la mission. Ensuite, une première phase de modélisation concerne le premier objectif de la mission. S'en suit une phase d'analyse traitant le premier objectif. Enfin, une seconde phase de modélisation traite le second objectif de la mission. Nous concluons cette section avec la seconde phase d'analyse permettant de produire la stratégie de l'APT.

4.1.1 Spécification de la mission

Tout d'abord, le concepteur construit le modèle de données de la mission dans la phase de spécification. Des recherches préalables permettent de dégager différents éléments composant le système cible ainsi que les potentiels problèmes qu'ils peuvent poser quant à l'accomplissement des objectifs. La Figure 4.1 montre le modèle de données avec ses cinq types de données : environnement, élément, objectif, obstacle et solution. On remarque notamment que la mission se décompose en deux objectifs de nature très différente : (i) "Faire déborder le réservoir" et (ii) "Ne pas se faire repérer". Le premier est un objectif propre à la mission et relativement distant de contraintes cyber classiques ce qui montre que notre approche peut modéliser des problématiques très spécifiques à un domaine. Il concerne l'élément "Réservoir" du système et possède deux obstacles : la régulation de l'"Arrivée d'eau" du système et celle de l'"Écoulement d'eau" du système. Le second objectif est un objectif récurrent dans le domaine de la cyber-sécurité ou dans le domaine militaire. Ceci montre que notre approche est également adaptée pour modéliser des problématiques plus abstraites et classiques de la cyber-sécurité.

Par des opérations de reconnaissance antérieures, l'APT possède d'ores et déjà une certaine connaissance du système. La station de pompage, présentée en Figure 4.2, est un système cyber-physique contrôlant automatiquement le niveau d'un réservoir d'eau grâce

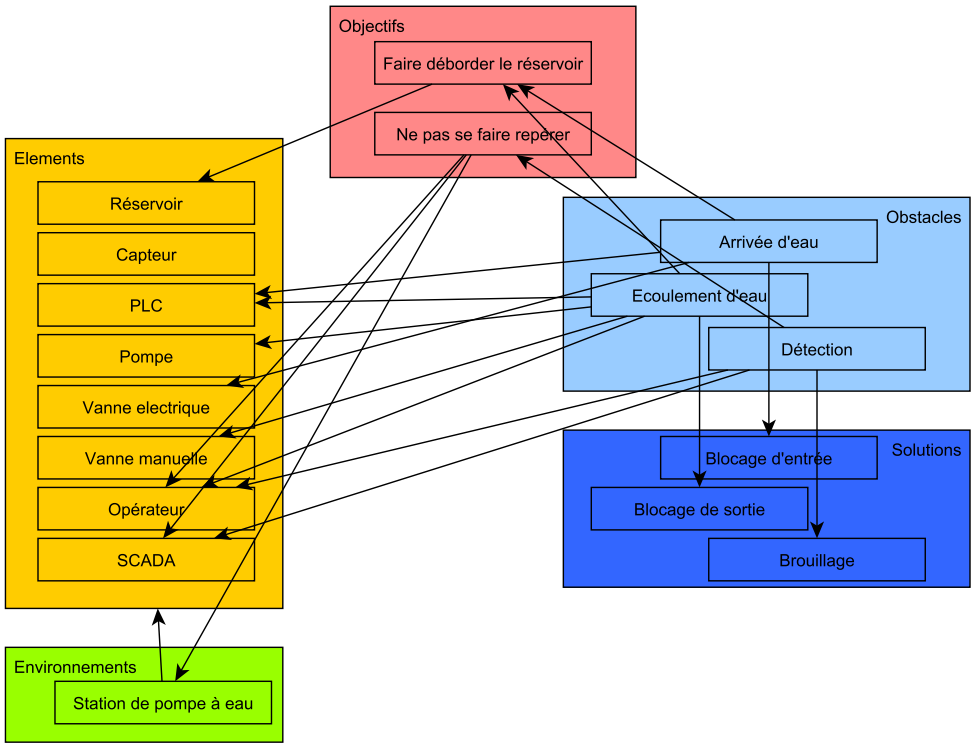


FIGURE 4.1 – Modèle de données de la station de pompage

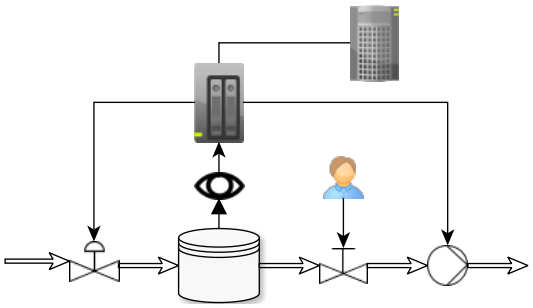


FIGURE 4.2 – Structure de la station de pompage

à un PLC (Programmable Logic Controller). Une vanne électrique d'entrée alimente le réservoir en eau, tandis qu'une vanne manuelle de sortie, connectée à une pompe électrique, vide le réservoir. La vanne manuelle est actionnable par un technicien. La vanne électrique et la pompe sont directement contrôlées par le PLC qui relève le niveau d'eau du réservoir au moyen d'un capteur. De plus, le PLC relaie régulièrement les mesures du niveau d'eau à une centrale SCADA (Supervisory Control And Data Acquisition) sur le réseau du site industriel. Le comportement du PLC suit les commandes suivantes : (i) ouvrir la vanne d'entrée et éteindre la pompe si le niveau d'eau atteint un seuil bas, (ii) fermer la vanne d'entrée et allumer la pompe si le niveau d'eau atteint un seuil haut, (iii) envoyer chaque mesure de niveau au SCADA.

Ces informations sont reliées au modèle de données par le mécanisme de fédération documentaire décrit en Section 3.2.2. Ainsi, le concepteur spécifie son modèle de données au préalable, qu'il enrichit ensutie de référence documentaire comme le schéma de la Figure 4.2 et les ressources textuelles.

4.1.2 Modélisation de l'attaque du réservoir

Dans un premier temps, nous traiterons de l'objectif (i) "Faire déborder le réservoir". Cet objectif est propre au système cible puisqu'il est lié au contexte de la station de pompage et n'a donc de sens que dans ce contexte. À travers cet objectif, nous montrons comment notre méthodologie permet de satisfaire les besoins de modélisation d'objectifs relatifs au domaine métier spécifique du système cible.

4.1.2.1 Modélisation du système

À partir de la description du système, nous identifions les différents composants qui se traduisent en éléments de modèle Pimca, Figure 4.3. Le réservoir, les vannes, la pompe, le capteur, le technicien, le PLC et le SCADA présents dans le schéma, Figure 4.3, sont capturés par des machineries car ils possèdent un comportement. De plus, nous ajoutons au modèle le réseau du site industriel à travers lequel communiquent le SCADA et le PLC, car c'est un point d'échange actif du système, donc une machinerie. Enfin, nous modélisons deux cibles potentielles pour un attaquant, à savoir les instructions suivies par le PLC et l'eau qui circule dans le système.

Ensuite, nous identifions les différentes relations du modèle Pimca. Les connexions entre les composants représentés dans le schéma initial sont modélisées par la relation bidirectionnelle *échange*. Pour les relations de plus haut-niveau, nous pouvons distinguer différentes classes : (i) le *contrôle* que le PLC exerce sur la vanne d'entrée, la pompe et le capteur, (ii) le *contrôle* exercé par le technicien sur la vanne manuelle, (iii) la *production* d'eau au regard du système de la vanne d'entrée pour le réservoir, (iv) l'*utilisation* de l'eau du réservoir par la pompe, (v) la *production* de commandes du SCADA au PLC, (vi) l'*utilisation* de l'eau par le réservoir et (vii) la *vérification* du niveau du réservoir par le PLC.

À partir de ce modèle de système initial, l'APT formule sa réflexion pour accomplir ses deux objectifs : (i) faire déborder le réservoir d'eau ; (ii) éviter d'être détectée. L'APT développe sa stratégie à l'aide de la méthodologie d'Operational Design en traitant chaque objectif séparément. On notera notamment que les éléments liés uniquement au second objectif ne sont pas encore modéliser dans la mesure où ils ne sont pas pertinents pour le premier objectif. Cette manière de procéder permet de maintenir un modèle cohérent et fonctionnel tout au long de l'approche, à la manière d'une "méthode agile".

La méthodologie que nous proposons adopte une approche à plusieurs points de vue sur la même cible (système, objectifs et obstacles). C'est pourquoi il est nécessaire de modéliser les trois préoccupations conjointement en faisant évoluer chaque modèle en même temps. Ce procédé est possible grâce à la fédération de modèles. Par la suite, nous identifions les contraintes dans le modèle de système qui sont nécessaires pour développer le modèle d'objectifs et le modèle d'obstacles.

4.1.2.2 Contraintes des objectifs

Pour faire déborder le réservoir d'eau, l'APT décide de se concentrer spécifiquement sur les composants physiques du système, c'est-à-dire le réservoir, la vanne d'entrée, la vanne de sortie et la pompe. Dans le modèle de système initial, seule la vanne d'entrée électrique semble capable d'augmenter le niveau d'eau du réservoir. Il est donc nécessaire qu'elle soit en marche dans l'environnement opérationnel désiré. À l'inverse, la pompe et la vanne manuelle sont directement responsables de la baisse de niveau du réservoir d'eau, il faut donc fermer au moins l'un des deux dans l'environnement opérationnel désiré.

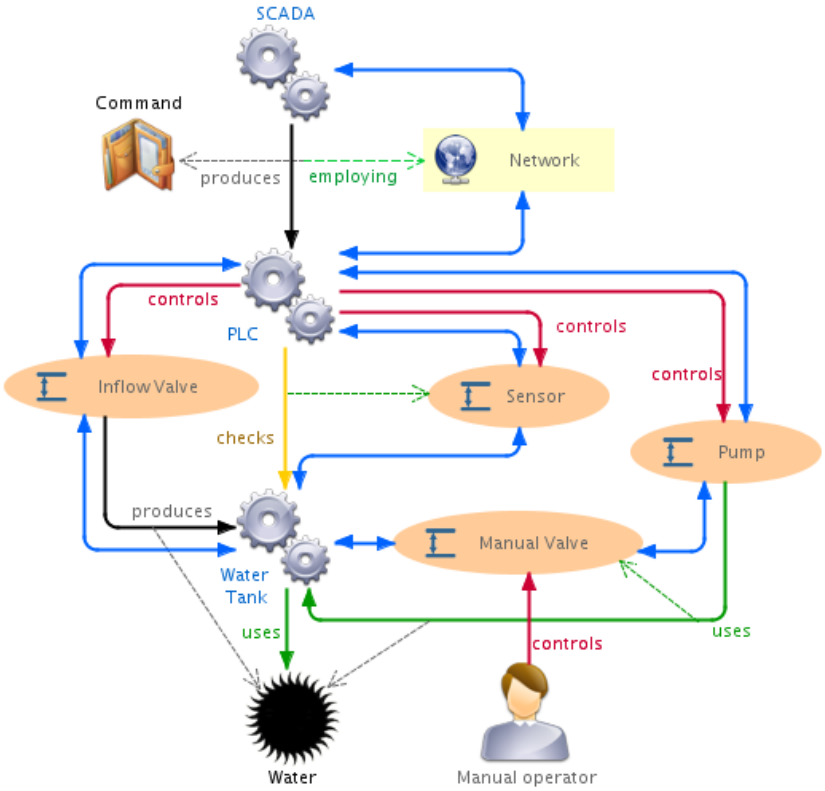


FIGURE 4.3 – Modèle structurel initial de la station de pompage

4.1.2.3 Contraintes des obstacles

Les problèmes posés par le système pour accomplir l'objectif (i) sont le mécanisme de régulation d'arrivée d'eau et le mécanisme d'écoulement d'eau du réservoir. Plusieurs éléments peuvent empêcher le débordement du réservoir d'eau : le PLC qui contrôle la vanne d'entrée et la pompe, le capteur qui relaie le niveau d'eau au PLC et l'opérateur qui peut ouvrir ou fermer la vanne de sortie manuelle.

4.1.2.4 Raffinement comportemental de l'environnement opérationnel courant

Cette identification des contraintes des objectifs et des obstacles permet à l'APT d'identifier les manques dans son modèle de système et d'enrichir son modèle Pimca d'environnement opérationnel courant avec un aspect comportemental DyPimca. L'APT modélise le comportement des éléments actifs, ou *machineries* Pimca, identifiés dans les problèmes de l'environnement. Chacun d'entre eux est associé à une unité d'exécution stipulant son comportement avec ses propres variables et ses propres commandes gardées. Le tableau 4.1 montre les différentes unités d'exécution mises en place et leurs commandes gardées respectives pour constituer le modèle de système comportemental pertinent pour le premier objectif. Ces commandes gardées sont les actions que les unités d'exécution peuvent entreprendre pour modifier l'état du système. Dans les tables de ce chapitre, les commandes seront représentées par leur nom et leur code d'exécution sera explicité séparément.
PLC : Le rôle du PLC est de maintenir le niveau d'eau du réservoir entre deux bornes prédéfinies, un seuil bas et un seuil haut. Pour cela, le PLC contrôle la vanne d'entrée électrique qui permet de faire entrer l'eau dans le réservoir et la pompe qui permet de faire sortir l'eau du réservoir. Concrètement, le PLC exerce son contrôle en envoyant deux types

WaterTank	PLC	InflowValve	ManualValve	Pump	Sensor	Operator
flowIn	update	flowOut	flowIn	flowIn	update	input
flowOut	regular	open	flowOut	open	refreshPLC	
refreshSens	highThres	close	open	close		
overflow	lowThres		close			
underflow	valveOn					
	valveOff					
	pumpOn					
	pumpOff					

Table 4.1 – Unités d'exécution et leurs commandes gardées dans le modèle d'environnement opérationnel courant pour l'objectif 1

de signaux à la vanne électrique et à la pompe : "ouvrir" ou "fermer". Enfin, le PLC envoie des ordres à ces deux éléments en fonction des décisions qu'il prend à partir d'informations relayées par le capteur. Le capteur transmet le niveau d'eau du réservoir au PLC, ce qui déclenche la commande *update* chez le PLC. Cette commande déclenche le mécanisme de décision du PLC qui peut correspondre à trois cas de figure :

— Si le niveau d'eau est au dessus du seuil haut acceptable, le PLC ordonne à la vanne électrique de se fermer avec le signal "fermer". Il ordonne également à la pompe de se mettre en marche avec le signal "ouvrir", ce qui permet au niveau d'eau du réservoir de baisser.

— Si le niveau d'eau est en dessous du seuil bas acceptable, le PLC ordonne à la vanne électrique de s'ouvrir avec le signal "ouvrir". Il ordonne également à la pompe de s'éteindre avec le signal "fermer", ce qui permet au niveau d'eau du réservoir de monter.

— Si le niveau d'eau se trouve entre les deux bornes acceptables, le PLC n'envoie pas de commandes pour changer le comportement de ses deux actionneurs.

Formellement, l'unité d'exécution du PLC est définie dans le langage de commandes gardées de la manière suivante :

```
update :                                              -Nom de la commande
    updateLevel?/                          -Canal de synchronisation (ici en réception)
    plcWaterLevel := sensorWaterLevel;          -Instruction : modification
    TriggerDecision := true;                          de l'etat du systeme

regular :                                             -Nom de la commande
    urgent                                                      -Urgence
    [TriggerDecision ∧                                    -Garde entre []
    plcWaterLevel < plcUpperThreshold ∧
    plcWaterLevel > plcLowerThreshold]/
    TriggerDecision := false;                                -Instruction

highThres :
    urgent
    [TriggerDecision ∧
    plcWaterLevel ≥ plcUpperThreshold]/
    TriggerDecision := false;
    TriggerCloseValve := true;
    TriggerOpenPump := true;
```

```
lowThres:
    urgent
    [TriggerDecision ∧
    plcWaterLevel ≤ plcLowerThreshold]/
    TriggerDecision := false;
    TriggerOpenValve := true;
    TriggerClosePump := true;

valveOff:
    urgent
    commandValveOff!                    -Canal de synchronisation (ici en émission)
    [TriggerCloseValve]/
    TriggerCloseValve := false;

valveOn:
    urgent
    commandValveOn!
    [TriggerOpenValve]/
    TriggerOpenValve := false;

pumpOff:
    urgent
    commandPumpOff!
    [TriggerClosePump]/
    TriggerClosePump := false;

pumpOn:
    urgent
    commandPumpOn!
    [TriggerOpenPump]/
    TriggerOpenPump := false;
```

Capteur : Le capteur suit un comportement plus simple. Dès que le niveau du réservoir change, le capteur en est averti à travers le canal *measure*. Ceci déclenche une mise à jour du niveau relevé par le capteur qui est ensuite transmis au PLC.

Formellement, l'unité d'exécution du capteur est définie dans le langage de commandes gardées de la manière suivante :

```
update:
    measure?/
    sensorWaterLevel := value;
    TriggerSensor := true;

refreshPLC:
    urgent
    updateLevel!
    [TriggerSensor]/
    TriggerSensor := false;
```

Vanne manuelle : La vanne manuelle suit un comportement similaire à celui du capteur. Dès que la pompe se met en marche et tente d'aspirer le contenu du réservoir avec sa commande gardée *flowIn*, celle-ci notifie la vanne manuelle qui se synchronise avec la pompe à travers sa commande gardée *flowOut*. La vanne manuelle transmet alors de manière

urgente au réservoir d'eau l'aspiration, à travers sa propre commande gardée *flowIn*. Toutefois, ce comportement n'est possible que si la vanne manuelle est ouverte, dans le cas contraire, la pompe ne peut pas réduire le niveau d'eau du réservoir. La vanne manuelle possède donc une variable booléenne interne *isOpen* qui tient compte de son état de fonctionnement. Le technicien est capable de changer cet état de fonctionnement à travers deux commandes gardées : *open* et *close*.

Formellement, l'unité d'exécution de la vanne manuelle est définie dans le langage de commandes gardées de la manière suivante :

```
flowIn:
    urgent
    decrease!
    [TriggerDecrease ∧
    isOpen]/
    TriggerDecrease := false;

flowOut:
    flow?
    [isOpen]/
    TriggerDecrease := true;

open:
    manualInput?
    [!isOpen]/
    isOpen := true;

close:
    manualInput?
    [isOpen]/
    isOpen := false;
```

Vanne électrique : La vanne électrique peut être soit ouverte, soit fermée. Ceci est capturé par une variable booléenne interne *isOpen*. Lorsqu'elle est ouverte, la commande *flowOut* peut être déclenchée, elle injecte l'eau dans le réservoir ce qui fait monter son niveau. L'ouverture et la fermeture de la vanne sont contrôlées par le PLC à travers des commandes gardées synchronisées *open* et *close*.

Formellement, l'unité d'exécution de la vanne électrique est définie dans le langage de commandes gardées de la manière suivante :

```
flowOut:
    increase!
    [isOpen]/;

open:
    commandIVOn?
    []/
    isOpen := true;

close:
    commandIVOff?
    []/
    isOpen := false;
```

Pompe : De même la pompe peut être soit ouverte, soit fermée. Elle possède également une variable booléenne interne *isOpen*. Lorsqu'elle est ouverte, la commande *flowIn* peut être déclenchée, elle se synchronise avec la commande *flowOut* de la vanne manuelle pour pomper l'eau du réservoir. L'ouverture et la fermeture de la pompe sont contrôlées par le PLC à travers des commandes gardées synchronisées *open* et *close*.

Formellement, l'unité d'exécution de la pompe est définie dans le langage de commandes gardées de la manière suivante :

```
flowIn:
    flow!
    [isOpen]/;

open:
    commandPumpOn?
    []/
    isOpen := true;

close:
    commandPumpOff?
    []/
    isOpen := false;
```

Réservoir d'eau : Le réservoir gère l'évolution de son niveau d'eau avec la variable entière *waterLevel*. Il se synchronise en réception de la vanne électrique à travers la commande gardée *flowIn* pour faire augmenter le niveau d'eau *waterLevel* d'une part. Il se synchronise en réception de la vanne manuelle à travers la commande gardée *flowOut* pour faire diminuer le niveau d'eau *waterLevel* d'autre part. Ces deux commandes changent le niveau d'eau, ce qui déclenche la commande gardée urgente *refreshSens* qui notifie le capteur du nouveau niveau d'eau.

Formellement, l'unité d'exécution du réservoir d'eau est définie dans le langage de commandes gardées de la manière suivante :

```
flowIn:
    urgent
    increase?
    []/
    TriggerRefresh := true;
    waterLevel := waterLevel+1;

flowOut:
    urgent
    decrease?
    []/
    TriggerRefresh := true;
    waterLevel := waterLevel-1;

refreshSens:
    urgent
    measure!
    [TriggerRefresh]/
    TriggerRefresh := false;
```

Technicien : Le technicien ou opérateur manuel est capable d'actionner la vanne manuelle à travers sa seule commande gardée *input*. Celle-ci change l'état d'ouverture ou de fermeture de la vanne.

Formellement, l'unité d'exécution du technicien est définie dans le langage de commandes gardées de la manière suivante :

```
input:
    manualInput!
    []/;
```

Le réseau et la centrale SCADA sont modélisés par des machineries dans le modèle Pimca du système de la centrale de pompage. Toutefois, l'APT décide de ne pas raffiner leurs unités d'exécution avec des commandes gardées à ce stade car ces éléments n'interviennent pas dans le cadre de l'objectif 1 qui consiste à faire déborder le réservoir.

4.1.2.5 Modélisation des objectifs

L'objectif de l'APT est de faire déborder le réservoir d'eau. Il est donc nécessaire de raffiner le modèle courant et d'inclure une commande gardée spécifique qui mesure si le réservoir déborde, c'est-à-dire si le niveau d'eau du réservoir dépasse la capacité limite du réservoir. Cette commande est la seule à manipuler une variable booléenne *waterOverflow* initialisée à *false* tant que le débordement n'a pas eu lieu et créée spécifiquement pour modéliser l'objectif de l'APT. La commande *overflow* est incluse dans le réservoir à cet effet et est formellement définie de la manière suivante :

```
overflow:
    urgent
    [waterLevel>=waterLevelMax]/
    waterOverflow := true;
```

Accomplir l'objectif 1 (Attaque du réservoir) de l'APT revient à trouver un scénario d'évolution du système dans lequel la variable *waterOverflow* devient vraie à un moment donné. L'objectif 1 n'est pas accompli si le réservoir ne déborde jamais, c'est-à-dire si la variable *waterOverflow* reste fausse indéfiniment. Concrètement, ceci est vérifiable par model-checking en cherchant un contre-exemple à la propriété suivante :

```
Objectif 1
    □ !waterOverflow
```

Le model-checker OBP2 montre que le système vérifie cette propriété dans tous les scénarios d'exécution actuels. Cela signifie que le réservoir d'eau ne déborde pas au cours du fonctionnement nominal du système modélisé par l'APT. Il faut donc que l'APT emploie diverses opportunités d'interaction avec le système pour accomplir son objectif.

4.1.2.6 Modélisation des obstacles

Le concepteur détermine les opportunités envisageables à partir du modèle de système courant. L'APT estime que le PLC est une machinerie trop complexe pour être la cible d'une attaque. En revanche, corrompre le technicien est une opération envisageable. C'est autour de cette opportunité que l'APT oriente sa réflexion. En corrompant le technicien, il lui semble possible de bloquer la vanne électrique dans un état constamment ouvert, de manipuler la vanne manuelle à sa guise et de bloquer la pompe indéfiniment.

Ces opportunités sont capturées à travers des commandes gardées présentées dans la Table 4.2.

InflowValve	Pump
forceOpen	block
close*	open*

Table 4.2 – Opportunités pour l'objectif 1 (* représente une altération)

Au niveau de la vanne électrique d'entrée d'eau, l'APT inclut une commande gardée *forceOpen* synchronisée avec l'unité d'exécution de l'APT. Lorsque *forceOpen* est déclenchée, la vanne reste indéfiniment ouverte, ce qui est capturé par une nouvelle variable booléenne interne *isForced*. Il est donc nécessaire de modifier la garde de la commande *close* afin d'empêcher son déclenchement. Puisque la commande *close* est une réception de message, il est crucial que la réception soit possible même si la vanne se retrouve bloquée afin que la simulation du modèle ne rencontre pas de deadlock. Il faut donc créer deux commandes *close* séparées pour capturer le comportement nominal lorsque l'attaque n'a pas eu lieu et le comportement bloqué lorsque l'attaque a eu lieu, ce qui s'écrit de manière formelle comme suit :

```
forceOpen:
    attackIV?
    []/
    isOpen := true;
    isForced := true;

close1:
    commandIVOff?
    [!isForced]/
    isOpen := false;

close2:
    commandIVOff?
    [isForced]/;
```

Au niveau de la vanne manuelle, l'ouverture et la fermeture sont contrôlées par le technicien. Si l'APT parvient à corrompre l'opérateur, alors elle obtient un contrôle sur la vanne manuelle, ce qui ne demande aucune modification du modèle courant.

Au niveau de la pompe, de manière similaire à la vanne d'entrée, l'APT injecte une commande *block* synchronisée avec une action d'attaque. Lorsqu'elle est déclenchée, la pompe ne peut plus se mettre en marche. De même que précédemment, cet état est capturé par une variable booléenne interne *isBlocked* à la pompe. Celle-ci permet de séparer l'ancienne commande *open* en deux commandes *open1* et *open2* capturant le comportement nominal et le comportement bloqué respectivement. Ceci est nécessaire pour assurer que le modèle de système ne rencontre pas de deadlock à la simulation car la commande *open* est une réception synchronisée avec le PLC. Formellement, les nouvelles commandes gardées de la pompe s'écrivent de la manière suivante :

```
block:
    attackPump?
    []/
    isOpen := false;
    isBlocked := true;

open1:
    commandPumpOn?
```

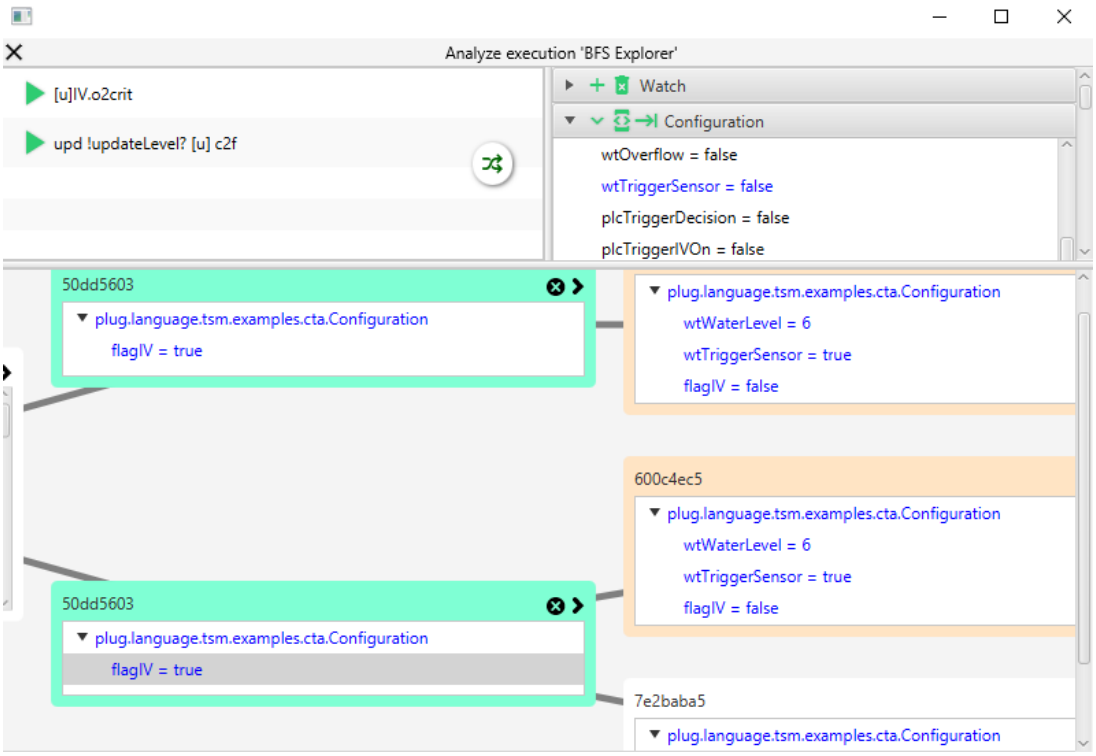


FIGURE 4.4 – Exploration de scénarios à travers OBP2

```
    [!isBlocked]/
    isOpen := false;

open2:
    commandPumpOn?
    [isBlocked]/;
```

Enfin, l'unité d'exécution du technicien est désactivée puisque l'APT en prend le contrôle pour agir directement sur le système. Le modèle d'obstacles permet à l'attaquant de définir des opportunités qu'il pourra exploiter au cours de l'attaque pour parvenir à faire déborder le réservoir : forcer la vanne électrique, bloquer la pompe et fermer la vanne manuelle.

4.1.3 Analyse de l'attaque du réservoir

À partir du modèle de système, d'objectifs et d'obstacles, l'APT peut simuler le comportement nominal du système de la station de pompage. Le model-checker OBP2.0 permet notamment d'explorer des scénarios d'évolution du système comme le montre la Figure 4.4. On peut voir le model-checker avec différents panels : (i) en haut à gauche, les transitions tirables sont affichées et peuvent être tirées ; (ii) en haut à droite, les variables d'états du système dans leur valeur courante sont affichées ; (iii) en bas, l'ensemble d'espace d'états explorés sont affichés et colorés lorsqu'ils représentent le même état du système. Sur ce panel on peut sélectionner l'état à partir duquel on souhaite continuer l'exploration. On notera par ailleurs que le modèle de système actuel fonctionne sans deadlock.

L'approche opérationnelle qui permet d'accomplir l'objectif 1 est développée à l'aide du model-checking en modifiant le modèle d'origine pour prendre en compte les opportunités identifiées. L'exploitation de ces opportunités est contrôlée par une unité d'exécution spécifique d'attaquant qui interagit avec le système. Les opportunités sont liées à des canaux

de synchronisation que l'unité d'exécution de l'attaquant utilise, à savoir : *attackIV*, *attackPump* et *manualInput*. Chaque opportunité est également liée à une variable booléenne interne à l'attaquant, qui mesure si l'opportunité a déjà été exploitée. Concrètement, l'unité d'exécution de l'attaquant possède les trois commandes gardées suivantes :

```
forceIV:
    attackIV!
    [!hasAttackedIV]/
    hasAttackedIV := true;

blockPump:
    attackPump!
    [!hasAttackedPump]/
    hasAttackedPump := true;

closeMV:
    manualInput!
    [!hasAttackedMV]/
    hasAttackedMV := true;
```

Grâce à cette unité d'exécution, le model-checking permet de vérifier si l'objectif de faire déborder le réservoir est réalisable compte tenu de ces trois opportunités. Pour cela, nous construisons des propriétés intermédiaires qui représentent les différentes combinaisons d'opportunités possibles utilisées par l'attaquant. Chaque combinaison est représentée par un vecteur booléen de taille 3 indiquant pour chaque opportunité si elle est exploitée ou non au cours de l'exploration de scénarios d'attaque. Par exemple, le vecteur *(true, true, false)* autorise l'attaquant à forcer la vanne électrique et à bloquer la pompe mais pas à fermer la vanne manuelle. La propriété intermédiaire correspondante impose cette valeur booléenne lors de l'exécution afin de vérifier si l'objectif est atteignable dans ce cas de figure. On écrit cette propriété de la manière suivante :

```
Objectif vanne_elec pompe
    □ ((!hasAttackedMV) ⇒ !waterOverflow)
```

Cette propriété signifie en langage courant : *"Tant que l'attaquant ne ferme pas la vanne manuelle, l'eau ne déborde pas"*. Elle impose à l'attaquant de ne pas faire usage de l'opportunité liée à la vanne manuelle mais autorise celui-ci a utiliser les deux autres opportunités.

Ajustements : L'ajout des différentes opportunités envisagées par le concepteur change le comportement du modèle. Le model-checker OBP2 montre que plusieurs scénarios d'exécution permettent de violer la propriété de l'objectif 1 en exploitant différentes opportunités. Toutefois, le système adopte un comportement inattendu et incohérent avec la réalité dans certains cas. Par exemple, lorsque la vanne électrique d'entrée est ouverte alors que la pompe et la vanne manuelle le sont également, le model-checker détecte un scénario d'exécution qui fait déborder le réservoir. Dans celui-ci, le réservoir reçoit une entrée d'eau depuis la vanne électrique à plusieurs reprises sans que la pompe ait l'occasion de faire sortir l'eau du réservoir alors qu'elle est en marche. Ceci s'explique par le fait que le fonctionnement nominal du système ne prévoit pas que ces deux éléments soient en fonctionnement simultanément. Ce scénario n'est pas crédible en réalité, car l'entrée et la sortie d'eau du réservoir sont modélisées comme ayant des débits relativement similaires d'après les informations collectées par l'APT. De fait l'APT ne peut compter sur une différence notoire de débit pour accomplir sa mission.

Pour prendre cela en compte, il faut implémenter ces règles de fonctionnement dans le modèle pour obtenir un comportement cohérent avec la réalité. Ainsi, le model-checker OBP2 pourra considérer des scénarios d'attaque réalistes et donc concrétisables pour l'APT. On reconnaît les problématiques d'exclusion mutuelle lors de partage d'une ressource. C'est pourquoi, pour ce faire [pour faire quoi ? pour reconnaître les problématiques ?], nous mettons en place l'algorithme de Peterson [81] entre la vanne électrique et la pompe. Cet algorithme permet de gérer le partage d'une ressource (ici le réservoir d'eau) entre plusieurs processus (ici la montée et la descente du niveau d'eau). Il permet notamment de gérer l'alternance de l'allocution de la ressource entre deux processus qui doivent l'utiliser tour à tour.

Data : *bool* flag[2], *int* tour
flag = {false, false};
tour = 0;
procedure P0
flag[0] = true;
turn = 1;
while *flag[0]* $\wedge$ *tour* == *1* **do**
| ... ▷ Attente
end
... ▷ Section critique P0
flag[0] = false;
end procedure
procedure P1
flag[1] = true;
turn = 0;
while *flag[1]* $\wedge$ *tour* == *0* **do**
| ... ▷ Attente
end
... ▷ Section critique P1
flag[1] = false;
end procedure

Algorithme 1 : Algorithme de Peterson entre deux processus P0 et P1

Appliqué au modèle, cet algorithme permet à la pompe et à la vanne électrique d'avoir un débit équivalent lorsque les deux fonctionnent en même temps. Concrètement, cet algorithme modifie les unités d'exécutions de la vanne électrique et de la pompe en ajoutant une variable partagée *turn* qui indique quelle unité a la priorité dans le cas où les deux pourraient interagir avec le réservoir. Cette priorité passe à l'unité d'exécution qui n'a pas interagi en dernier. Les deux unités d'exécution signalent leur possibilité d'interagir avec le réservoir à travers deux variables booléennes partagées *flagIV* et *flagPump*.

Formellement la nouvelle unité d'exécution de la vanne électrique est la suivante :

```
canFlowOut:
    urgent
    [isOpen ∧ !flagIV]/
    flagIV := true;

flowOut:
    increase!
    [isOpen ∧ (!flagPump \vee turn ==TURN_IV)]/
    turn := TURN_PUMP;;
```

```
open:
    commandIVOn?
    []/
    isOpen := true;

forceOpen:
    attackIV?
    []/
    isOpen := true;
    isForced := true;

close1:
    commandIVOff?
    [!isForced]/
    isOpen := false;
    flagIV := false;

close2:
    commandIVOff?
    [isForced]/;
```

Formellement la nouvelle unité d'exécution de la pompe est la suivante :

```
canflowIn:
    urgent
    [isOpen ∧ !flagPump]/
    flagPump := true;

flowIn:
    flow!
    [isOpen ∧ (!flagIV ∨ turn ==TURN_PUMP)]/
    turn := TURN_IV;

close:
    commandPumpOff?
    []/
    isOpen := false;

block:
    attackPump?
    []/
    isOpen := false;
    isBlocked := true;

open1:
    commandPumpOn?
    [!isBlocked]/
    isOpen := false;

open2:
    commandPumpOn?
    [isBlocked]/;
```

Forcer la vanne électrique		•			•	•
Bloquer la pompe			•			•
Fermer la vanne manuelle				•	•	
Objectif 1	X	X	X	X	O	O

Table 4.3 – Model-checking de l'objectif 1 (O : succès, X : échec)

PLC	Network	SCADA
signal	receive	regularLvl
	send	dangerLvl
		alert

Table 4.4 – Ajout de commandes gardées au modèle d'environnement opérationnel courant pour l'objectif 2

Grâce à cet algorithme, le modèle est fidèle avec la réalité, ce qui permet à l'APT de déterminer des scénarios réalisables dans lesquels son objectif 1 s'accomplit. L'approche opérationnelle est vérifiée formellement avec OBP2. La Table 4.3 présente nos résultats. Chaque colonne représente une combinaison d'opportunités utilisée par l'APT et le résultat de l'attaque menée vis-à-vis des objectifs. On peut notamment voir dans la dernière colonne le scénario dans lequel l'attaquant a accès a deux opportunités : forcer la vanne électrique et bloquer la pompe. Dans ce cas de figure, l'attaquant arrive à faire déborder le réservoir. Au cours de cette analyse, 3256 états et 3255 transitions uniques ont été explorés pour vérifier 8 propriétés représentant différentes combinaisons d'opportunités considérées par l'APT pour accomplir l'objectif 1. Le temps de calcul de cette analyse est de 6238 millisecondes.

En ce qui concerne l'objectif 1, le model-checker OBP2 montre que l'APT doit exploiter au moins deux des opportunités que le concepteur a identifiées. Il est nécessaire de forcer l'ouverture de la vanne d'entrée et de fermer la vanne manuelle ou la pompe. À partir de ces déductions, l'APT peut concrétiser un scénario en une stratégie opérationnelle concrète sur le terrain. Toutefois la stratégie actuelle ne prend en compte qu'un seul objectif, l'APT a un second objectif à accomplir.

4.1.4 Modélisation pour assurer la discrétion de l'attaque

L'APT raffine ensuite son modèle pour prendre en compte un second objectif : ne pas se faire repérer. Cet objectif est plus générique que celui de faire déborder le réservoir. Il concerne la problématique de la discrétion omniprésente lors d'une attaque cyber. Nous montrons comment notre méthodologie peut modéliser cet aspect dans le cadre du système de station de pompage mais d'un point de vue suffisament générique pour être réutilisable dans d'autres contextes.

4.1.4.1 Raffinement comportemental du modèle de système

Afin de prendre en compte le nouvel objectif de discrétion, l'APT doit modéliser la centrale SCADA qui est responsable de sonner l'alerte en cas de débordement. La centrale SCADA relève les informations de niveau d'eau depuis le réseau interne du système qui la relie au PLC. Il est donc nécessaire de modéliser la chaîne de communication entre le PLC, le réseau et le SCADA dans l'environnement opérationnel courant.

Le tableau 4.4 détaille les différentes commandes gardées ajoutées au modèle pour capturer le comportement de la chaîne de communication.

PLC : Le PLC notifie la centrale SCADA à travers l'infrastructure réseau du système lorsqu'une nouvelle mesure est prise par le capteur. Pour ce faire, la prise de décision du PLC entre les commandes gardées *regular*, *highThres* et *lowThres* déclenche une nouvelle commande gardée *signal*. Le signal transmis peut prendre deux valeurs : *REGULAR* si le niveau d'eau du réservoir se situe entre les bornes acceptables, et *DANGEROUS* sinon. La valeur du signal à envoyer est stockée dans une variable *message* interne au PLC. La nouvelle valeur est assignée lors de la prise de décision. L'APT ajoute donc une commande gardée *signal* au PLC et modifie trois commandes gardée déjà présentes (*regular*, *highThres* et *lowThres*). Formellement les nouvelles commandes de l'unité d'exécution du PLC sont définies de la manière suivante :

```
regular:
    urgent
    [TriggerDecision ∧
    plcWaterLevel < plcUpperThreshold ∧
    plcWaterLevel > plcLowerThreshold]/
    TriggerDecision := false;
    TriggerSignal := true;
    message := REGULAR;

highThres:
    urgent
    [TriggerDecision ∧
    plcWaterLevel ≥ plcUpperThreshold]/
    TriggerDecision := false;
    TriggerCloseValve := true;
    TriggerOpenPump := true;
    TriggerSignal := true;
    message := DANGEROUS;

lowThres:
    urgent
    [TriggerDecision ∧
    plcWaterLevel ≤ plcLowerThreshold]/
    TriggerDecision := false;
    TriggerOpenValve := true;
    TriggerClosePump := true;
    TriggerSignal := true;
    message := DANGEROUS;

signal:
    urgent
    PLC2Network!
    [TriggerSignal]/
    TriggerSignal := false;
```

Réseau : Le réseau joue le rôle d'intermédiaire entre le PLC et la centrale SCADA. Lorsque le PLC envoie un signal avec sa commande gardée *signal*, le réseau la reçoit avec sa commande gardée *receive*. Le réseau copie le message binaire du PLC à transmettre à la centrale SCADA. Ensuite, le réseau déclenche de manière urgente la commande gardée *send* pour transmettre le signal au SCADA. Formellement, l'unité d'exécution du réseau est définie de la manière suivante :

```
receive:
    PLC2Network?/
    message := PLC.message;
    TriggerNetwork := true;

send:
    urgent
    Network2SCADA!
    [TriggerNetwork]/
    TriggerNetwork := false;
```

Centrale SCADA : La centrale SCADA reçoit et traite les informations envoyées par le PLC. Si le SCADA mesure deux signaux *DANGEROUS* à la suite, alors il lance l'alerte. Sinon le SCADA retourne dans son état normal lorsqu'il reçoit un signal *REGULAR*. L'alerte est mesurée par la variable booléenne interne *alert* initialisée à *faux*. Lorsque la commande *send* du réseau transmet un signal, le SCADA réceptionne avec une commande différente en fonction du signal transmis précédemment. Le signal précédent est donc gardé en mémoire par le SCADA dans une variable interne *message*. Formellement, le comportement de la centrale SCADA est modélisé par l'unité d'exécution suivante :

```
receiveRegular:
    Network2SCADA?
    [message == REGULAR]/
    message := Network.message;

receiveDangerous:
    Network2SCADA?
    [message == DANGEROUS]/
    if {Network.message == DANGEROUS} {
        alert := true;
        }
    else {
        message := REGULAR;
        };
```

4.1.4.2 Modélisation des objectifs

Le second objectif de l'APT est de ne pas se faire repérer pendant ses opérations. L'alerte est modélisée par la variable de la centrale SCADA *alert* qui devient vraie lorsque le niveau du réservoir d'eau est anormal. Accomplir l'objectif 2 de l'APT revient donc à trouver un scénario d'évolution du système dans lequel la variable *alert* reste indéfiniment fausse. Concrètement, ceci est vérifiable par model-checking en cherchant un contre-exemple à la propriété suivante :

```
Objectif 2
    ◊ alert
```

La mission est contrainte à la fois par l'objectif 1 et par l'objectif 2. La propriété globale dont le model-checker OBP2 doit chercher le contre-exemple est la suivante :

```
Mission
    (□!waterOverflow) ∨ (◊alert)
```

Sensor	Network
disable	jam
refreshPLC*	send*

Table 4.5 – Opportunités pour l'objectif 2 (* représente une altération)

Cette propriété se traduit en langage courant par *"Jamais l'eau ne déborde du réservoir à moins qu'une alerte finisse par se déclencher"*. Elle traduit le comportement attendu du système par son propriétaire : le niveau d'eau doit être régulé dans le réservoir ou l'alerte doit être sonnée. En tant qu'attaquant, l'APT cherche à compromettre ce comportement attendu. À travers notre méthode d'analyse utilisant le model-checking, nous cherchons donc un contre-exemple à cette propriété, c'est-à-dire un scénario dans lequel cette propriété est violée. Dans ce scénario, l'attaquant sait que le réservoir déborde sans que l'alerte ne soit sonnée. C'est donc un scénario d'attaque réussie. De manière générale, la propriété globale de la mission se construit comme une propriété dont le contre-exemple montre que tous les objectifs de la mission sont remplis. De ce fait, si on écrit au préalable des propriétés représentant l'accomplissement de chaque objectif particulier, comme c'est le cas ici, la propriété globale de la mission P se construit simplement grâce à l'opérateur logique transitif OU entre toutes les propriétés p_n de la manière suivante : $P = p_1 \vee ... \vee p_n$.

Ainsi, si l'attaquant parvient à trouver un contre-exemple dans lequel P est violée, on a alors simplement : $!P = !p_1 \wedge ... \wedge !p_n$. Ce qui signifie que les propriétés de chaque objectif sont violées simultanément. Or les propriétés sont construites de telle manière que le fait qu'une propriété soit violée équivaut à un objectif accompli. Il y a donc équivalence entre la violation de la propriété globale P et l'accomplissement simultané de tous les objectifs.

4.1.4.3 Modélisation des obstacles

Plusieurs éléments interviennent dans la chaîne de communication qui peut lancer l'alerte, à savoir le capteur, le PLC, le réseau et la centrale SCADA.

Opportunités identifiées : De nouveau, le concepteur détermine les opportunités envisageables pour atteindre la chaîne de communication du modèle de système courant. L'APT estime que la centrale SCADA est une machinerie trop complexe pour être la cible d'une attaque au même titre que le PLC. En revanche, brouiller le réseau et désactiver le capteur sont deux opportunités envisagées. Les modifications faites sur le modèle sont présentées dans le tableau 4.5.

Au niveau du réseau, l'APT ajoute une commande gardée *jam*, synchronisée avec l'unité d'exécution de l'APT. Lorsque le réseau est brouillé par cette commande, le réseau ne transmet plus les signaux du PLC au SCADA. Ceci est représenté par l'ajout d'une condition dans la garde de la commande *send*, qui ne lui permet de s'exécuter que si le réseau n'est pas brouillé. Ce qui s'écrit de manière formelle comme suit :

```
jam:
    jamNetwork?/
    isJammed := true;

send:
    urgent
    Network2SCADA!
    [TriggerNetwork ∧ !isJammed] /
    TriggerNetwork := false;
```

Au niveau du capteur, l'APT ajoute la commande gardée *disable* pour mettre le capteur hors service. Lorsque le capteur est désactivé, celui-ci ne transmet plus le niveau d'eau au PLC via la commande *refreshPLC*. Comme pour le réseau, cela se traduit par l'ajout d'une condition dans la garde de la commande *refreshPLC*, qui ne lui permet de s'exécuter que si le capteur est actif. Ce qui s'écrit de manière formelle comme suit :

```
refreshPLC:
    urgent
    updateLevel!
    [triggerSensor ∧ !isDisabled]/
    triggerSensor := false;

disable:
    corruptSensor?/
    isDisabled := true;
```

Le modèle d'obstacles permet de définir des opportunités supplémentaires pour l'attaquant. Pour remplir les deux objectifs de la mission, l'attaquant peut maintenant : brouiller le réseau et désactiver le capteur.

4.1.5 Analyse pour assurer la discrétion de l'attaque

L'approche opérationnelle pour atteindre les deux objectifs est développée à l'aide du model-checking. Au cours de cette analyse, 9365 états et 9364 transitions uniques ont été explorés pour vérifier 44 propriétés représentant différentes combinaisons d'opportunités envisagées par l'APT pour accomplir ses deux objectifs. Lors de cette instance précise du cas d'étude, le calcul a pris 68 596 millisecondes. Le temps de calcul moyen de l'analyse est d'environ 1 minute.

De même que lors de l'analyse précédente, nous construisons des propriétés intermédiaires pour représenter différentes combinaisons d'opportunités que l'attaquant peut exploiter pour accomplir sa mission. Ces propriétés se construisent à partir d'un vecteur booléen représentant la combinaison d'opportunités autorisées avec la propriété globale de la mission de la manière suivante :

```
Mission combinaison_n
    □ ((!opportunites_interdites) ⇒ !Mission)
```

La Figure 4.5 montre l'interface de OBP2 On y voit deux panels : (i) À gauche, les différentes propriétés peuvent être vérifiées soit automatiquement (flèche de gauche) ou pas à pas (icône humaine). Le nombre d'états et de transitions parcourus ainsi que le temps de calcul s'affichent également sous la propriété une fois l'exécution terminée. Le résultat de la vérification est symbolisé par la couleur du symbole, vert signifie que la propriété est vérifiée alors que rouge signifie que la propriété n'est pas vérifiée. L'exécution pas à pas permet d'explorer la trace d'exécution comme dans la Figure 4.4. Ce faisant [en explorant la trace?], on peut analyser le contre-exemple et déterminer la stratégie d'attaque. (ii) À droite, l'éditeur de propriétés permet d'écrire les propriétés à faire vérifier par le model-checker, cet éditeur permet d'écrire les propriétés à la main ou de les importer directement depuis un fichier comme c'est le cas ici. Par exemple, la propriété "overflow_capability1" vérifie si le réservoir peut déborder ("wtOverflow") si l'APT force la vanne électrique d'entrée ("aHasForced").

Le tableau 4.6 détaille les résultats d'exécution d'OBP2 lors de la simulation et de la vérification du modèle. Ce tableau a été construit après avoir vérifié différentes propriétés intermédiaires. En effet,le model-checking permet de formellement vérifier une propriété

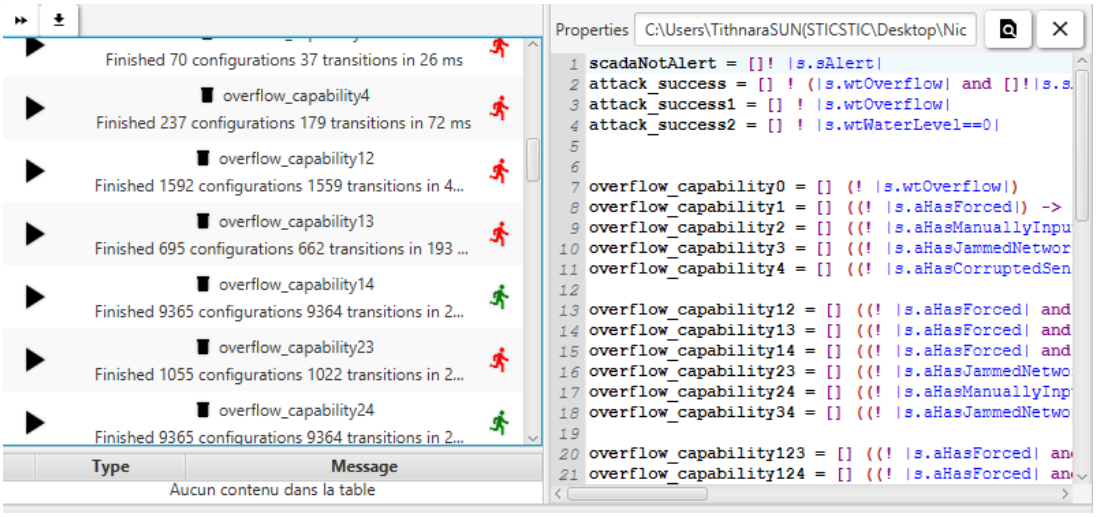


FIGURE 4.5 – Model checking des attaques sur OBP2

s'il n'existe aucun contre-exemple. Dans le cas où la propriété est violée, cela signifie qu'il existe un contre-exemple, c'est-à-dire un scénario dans lequel la propriété est violée. Ce scénario représente un chemin d'attaque pour l'attaquant puisque son objectif est de trouver une stratégie mettant en défaut le système. Chaque combinaison d'opportunités est représentée dans la partie supérieure, tandis que la partie inférieure montre si les objectifs sont accomplis pour cette combinaison en particulier. À des fins de concision, les 44 propriétés intermédiaires ne sont pas toutes représentées : seules les plus pertinentes sont représentées, à savoir les combinaisons minimales d'opportunités exploitées permettant d'atteindre les objectifs. La partie de gauche du tableau reprend les résultats du Tableau 4.3 alors que la partie de droite met en jeu les deux nouvelles opportunités. Il est inutile de représenter les combinaisons mettant en jeu plus d'opportunités car, s'il existe un chemin d'attaque mettant en jeu moins d'opportunités, il est évident que plus d'opportunités permettront d'arriver au même résultat. On voit notamment que plusieurs combinaisons d'opportunités exploitées permettent d'atteindre les deux objectifs en même temps. Par exemple, forcer la vanne électrique, fermer la vanne manuelle et brouiller le réseau (avant dernière colonne) sont trois opportunités qui, ensemble, permettent d'accomplir la mission. Comme on peut s'y attendre, brouiller le réseau ou désactiver le capteur permettent d'accomplir l'objectif 2 : ne pas se faire repérer au cours de l'attaque. Cet objectif de discrétion est un objectif générique des attaquants de la cyber-sécurité. Par contre, brouiller le réseau n'influe pas sur l'objectif 1. Il est donc impératif d'utiliser les opportunités déterminées précédemment pour accomplir la mission dans son ensemble. En revanche [OKX cza s'oppose à quoi ?], il est intéressant de noter que la désactivation du capteur à un moment précis permet d'atteindre tous les objectifs en exploitant cette seule opportunité. À l'aide de cette méthodologie, l'APT identifie l'opportunité unique minimum pour parvenir à ses fins. Elle peut ensuite mettre son plan à exécution en le concrétisant. La stratégie opérationnelle à adopter consiste à désactiver le capteur mesurant le niveau d'eau du réservoir au moment opportun, c'est-à-dire quand la vanne d'entrée est ouverte et la vanne de sortie ou la pompe est fermée.

Ce cas d'étude démontre comment notre méthodologie s'applique de manière concrète. Ce faisant, elle s'appuie sur des ressources documentaires variées (textes, schémas, spécification de comportements) pour la phase de spécification de la mission. À l'issue de cette première phase, l'APT a construit le modèle de données qu'elle utilisera tout au long de l'approche. Ensuite, le concepteur procède en deux temps pour traiter les deux objectifs

Forcer la vanne électrique		•			•	•			•		•	•
Fermer la vanne manuelle			•			•		•				•
Bloquer la pompe				•	•			•			•	
Brouiller le réseau							•	•	•		•	•
Désactiver le capteur										•		
Faire déborder le réservoir	X	X	X	X	O	O	X	X	X	O	O	O
Ne pas se faire repérer	X	X	X	X	X	X	O	O	O	O	O	O

Table 4.6 – Model-checking des deux objectifs (O : succès, X : échec)

de la mission. Il procède tout d'abord à une première phase de modélisation au cours de laquelle il obtient un modèle fonctionnel pour la phase d'analyse. Après avoir analysé le modèle, il corrige une incohérence logique puis procède au développement de la stratégie permettant de remplir le premier objectif. Il procède enfin à la seconde phase de modélisation pour prendre en compte le deuxième objectif. À l'issue de la seconde phase d'analyse, l'attaquant détermine la stratégie grâce à la méthodologie : Désactiver le capteur de niveau d'eau du réservoir.

4.2 Évaluation de la méthodologie et des outils

Cette section évalue les résultats de notre approche. Dans un premier temps, nous dressons le bilan du cas d'étude. Dans un second temps, nous dressons le bilan général de l'approche vis-à-vis des exigences énoncées dans le Chapitre 2.4, nous faisons état des limites rencontrées et nous proposons des pistes d'amélioration.

4.2.1 Bilan du cas d'étude

Le cas d'étude montre comment notre approche est concrètement mise en application avec l'exemple de station de pompage d'eau. Nous suivons la méthodologie phase par phase : spécification, modélisation et analyse.

4.2.1.1 Phase de spécification

La phase de spécification met en évidence le haut niveau d'abstraction de l'Operational Design. Il n'est pas nécessaire que le concepteur ait une expertise dans les différents domaines nécessaires à la conception d'un système complexe. Le concepteur peut modéliser le système à "très gros grain" afin de représenter son fonctionnement de manière abstraite. Notre instance de modèle de données contient un total de 19 instances de données : 1 environnement, 8 éléments, 2 objectifs, 3 obstacles et 5 opportunités. **Le modèle de données est compréhensible et succinct, ce qui permet son utilisation comme support de réflexion et de discussion entre différents agents qui ne sont pas nécessairement experts dans tous les domaines mis en jeu.** Les cinq catégories de données orientent la réflexion du concepteur avec les préoccupations de la mission en termes de système cible, d'objectifs et d'obstacles. Ceci permet de faciliter l'organisation des données recueillies tout en maintenant la clarté du modèle, ce qui facilite les phases de modélisation et d'analyse. La fédération de modèles est essentielle dans notre approche en trois phases. En effet, le modèle de données est relativement simple mais il entraîne une grande complexité dans les phases de modélisation et d'analyse de par les différents modèles mis en jeu. Il est donc primordial de préparer la conceptualisation des modèles dès la phase de spécification avec la fédération.

4.2.1.2 Phase de modélisation

La phase de modélisation montre comment construire le modèle de système pas à pas, d'abord en suivant la spécification, puis en ajoutant les contraintes liées au modèle d'objectifs et d'obstacles. Lors de cette phase, nous avons construit un modèle DyPimca structurel enrichi de comportements dynamiques avec 41 commandes gardées. La fédération de modèles dans l'environnement OpenFlexo permet d'enrichir les concepts inférés dans la phase de spécification en les reliant à des concepts opérationnels. Ces concepts opérationnels sont modélisés dans des langages divers : DyPimca ou commande gardée. La fédération permet de maintenir ces liens et de faciliter la création de nouveau artefacts de modélisation.

Le modèle de comportement du système a été construit à la suite d'une étude préliminaire disponible en Annexe A. Lors de ce travail préparatoire, nous avons modélisé le comportement du système par des automates à états finis en UPPAAL. Cela nous a permis d'identifier les différentes briques comportementales à mettre en place dans ce cas d'étude sous forme de commandes gardées. Nous avons choisi un autre paradigme pour le modèle comportemental, les commandes gardées, car les automates se sont montrés difficiles à maintenir et à étendre, contrairement au formalisme de commandes gardées. Il est vrai

que les automates sont plus expressifs que les commandes gardées. Cependant, notre approche impose de faire évoluer le comportement du modèle en fonction des opportunités identifiées par l'attaquant. De plus, la structure du système en différents éléments structurels et comportementaux demande de prendre en compte la composition de ces éléments afin de faire émerger un comportement global. Ces mécanismes sont lourds à maintenir pour un ensemble d'automates, car chaque nouveau comportement peut induire des incompatibilités. En revanche, les commandes gardées s'adaptent mieux à ces contraintes intrinsèques de notre approche. En effet, les commandes gardées reposent nativement sur une logique de premier ordre, ce qui par nature permet une composabilité.

La seconde itération de modélisation montre les atouts de la fédération de modèles. **Les différents modèles sont facilement extensibles et réutilisables. De plus, la fédération aide à maintenir la cohérence des différents modèles tout au long de l'approche.**

4.2.1.3 Phase d'analyse

Au cours de la phase d'analyse finale, 9365 états et 9364 transitions uniques ont été explorés pour vérifier 44 propriétés en un temps de calcul moyen d'environ 1 minute. La taille de l'espace d'états est très restreinte (environ 10 000) comparée à l'espace d'états d'automates industriels dépassant plusieurs millions d'états. Ceci montre la pertinence du choix du model checking dans notre approche. En effet, le model-checking est une technique d'analyse formelle qui produit des résultats formellement vérifiés. La difficulté principale du model-checking est l'explosion combinatoire qui rend la technique inapplicable sur des espaces d'états trop vastes. Notre utilisation du model-checking se restreint à des espaces d'états très réduits par construction et par volonté de modéliser le système à un haut niveau d'abstraction. L'explosion combinatoire n'est donc pas un problème auquel notre approche se heurte. **Le cas d'étude montre que notre approche produit des résultats formellement vérifiés en des temps avoisinant la minute, ce qui est satisfaisant dans notre contexte.**

Le cas d'étude permet d'illustrer concrètement les avantages de notre approche. D'une part, la simplicité du modèle de données de départ montre qu'il est possible d'obtenir des résultats concrets avec un niveau d'abstraction très élevé. La stratégie opérationnelle obtenue aux termes de notre approche est complexe et prend en compte divers moyens d'actions pour l'attaquant. D'autre part, nous avons modélisé des contraintes de différentes natures (réseau, physique, humaine) à différents niveaux. Ceci démontre que notre approche est adaptée pour différents domaines et différents niveaux d'abstraction. Libre au concepteur de modéliser au niveau qui lui convient. La spécification maintient un niveau de discours compréhensible par tous les acteurs de l'approche tout en facilitant les développements futurs. La modélisation est aisée grâce à la fédération de modèles qui assure la cohérence de l'ensemble des artefacts créés. L'analyse est rapide et formellement vérifiée.

4.2.2 Bilan général

D'un point de vue global, l'approche que nous proposons répond à la question : "Comment l'APT établit-elle sa stratégie ?". Nous montrons comment, à partir d'objectifs de mission, il est possible de construire un modèle de données, puis différents modèles de système, d'objectifs et d'obstacles afin de produire une analyse résultant en une stratégie opérationnelle. L'approche produit un diagnostic du système sous la forme d'une approche opérationnelle qu'adopterait une menace persistante avancée pour s'attaquer au système. Le cas d'étude valide l'exécutabilité de l'approche. Toutefois, le traitement du cas d'étude démontre que notre approche nécessite certaines interventions "à la main" de l'utilisateur,

		Avancée de la thèse
Modèle de systèmes	Méthodologie	✓
	Syntaxe abstraite	✓
	Syntaxe concrète	✓
	Sémantique	✓
Modèle de stratégie	Méthodologie	✓
	Syntaxe abstraite	~✓
	Syntaxe concrète	~✓
	Sémantique	~✓
Interopérabilité	Approche	✓
	Outillage	✓
Analyse	Approche	✓
	Outillage	✓

Table 4.7 – Bilan des exigences de l'approche

ce qui prouve qu'il reste des efforts à fournir en termes d'exécutabilité sans intervention du concepteur. Pour conclure, voici une évaluation globale de l'approche au regard de la validation des exigences que nous avons identifiées. Le Tableau 4.7 résume les différentes exigences introduites auparavant ainsi que leur avancement dans la thèse. L'ensemble positif, sera détaillé dans les paragraphes suivants.

4.2.2.1 Modéliser le système ciblé

La modélisation du système pour les besoins de l'élaboration de stratégie des menaces persistantes avancées consistait en la caractérisation du système d'un point de vue structurel et comportemental selon notre méthodologie. Ces deux aspects ont été modélisés avec succès à l'aide du langage Pimca pour la modélisation statique et structurelle, et à l'aide son extension dynamique DyPimca pour la modélisation dynamique et comportementale.

Nous avons notamment spécifié une syntaxe concrète graphique pour le langage Pimca dans l'environnement OpenFlexo. Quant au langage DyPimca, nous avons spécifié sa syntaxe abstraite, sa syntaxe concrète textuelle et sa sémantique d'exécution afin d'automatiser une partie du diagnostic. **Ceci valide l'intégralité des exigences spécifiées pour la modélisation de systèmes.**

4.2.2.2 Modéliser la stratégie

La modélisation de la stratégie des menaces persistantes avancées consistait en l'adaptation de la méthodologie militaire de l'Operational Design. Nous avons adapté l'Operational Design au contexte des APT en outillant le plus de processus possibles. Ce faisant, nous montrons par l'exemple que l'Operational Design est applicable dans le domaine de la cyber-sécurité contre les APT. Ceci valide les exigences méthodologique et de syntaxe concrète pour la modélisation de la stratégie. La stratégie que nous modélisons est construite comme une succession d'opportunités à exploiter. Notre implémentation est fonctionnelle et répond à nos exigences initiales. Toutefois, pour modéliser une stratégie plus fine, il est nécessaire de prendre du recul sur nos travaux et d'expliciter un langage d'expression de stratégie. La méthodologie prodigue les outils pour qu'un attaquant développe sa stratégie automatiquement, mais il peut être intéressant de prodiguer des patterns pour établir des stratégies plus complexes avec plus de contraintes.

Ceci montre que la méthodologie d'Operational Design est concrètement applicable pour les APT. Toutefois, ces résultats ont été obtenus en partie par des efforts

manuels. Un effort de formalisation du langage de stratégie est requis afin d'obtenir un syntaxe abstraite complète ainsi qu'une sémantique d'exécution bien définie. La sémantique de la stratégie que nous modélisons est une séquence d'opportunités que l'attaquant doit exploiter afin d'accomplir sa mission. Cette sémantique gagnerait à être raffinée mais elle est d'ores et déjà utilisable en l'état.

4.2.2.3 Gérer l'interopérabilité

Les premières exigences introduisent naturellement le problème de l'interopérabilité entre les différents DSML que nous utilisons. Cette problématique inclut deux niveaux, un niveau conceptuel et un niveau technique. Au niveau conceptuel, la fédération nous a permis de construire, dans la phase de spécification, le modèle de données sur lequel se base toute l'approche. Au niveau technique, la fédération nous permet de résoudre le problème de l'interopérabilité et elle nous permettra d'associer ces travaux à d'autres DSML dans le futur. Dans notre domaine en pleine expansion, étant donné les différents formalismes utilisés, **l'approche par fédération semble la mieux indiquée**.

C'est pourquoi nous avons mis en place notre framework au sein de l'environnement de fédération de modèle OpenFlexo. **L'outillage que nous proposons est opérationnel** et nous a permis de développer notre cas d'étude.

4.2.2.4 Analyser les modèles

L'approche que nous préconisons consiste en l'assistance à l'élaboration de stratégies des APT. Pour ce faire, nous avons employé la **technique d'analyse formelle du *model checking* pour garantir un diagnostic fiable et rapide**. Nous avons mis en place cette technique à l'aide du *model checker* OBP2. **L'outillage que nous proposons est opérationnel** et permet de produire automatiquement une stratégie opérationnelle.

4.2.3 Limites de l'approche

Nos travaux connaissent toutefois quelques lacunes, que ce soit au niveau du cas d'étude ou de l'approche globale.

Du point de vue du cas d'étude, il est vrai que nous avons implémenté l'ensemble de notre méthodologie sur une station de pompage d'eau. Ceci démontre la faisabilité de notre approche dans ce contexte. Toutefois, il reste à démontrer que la méthodologie et les outils que nous proposons sont adaptés à la construction de stratégie cyber au sens large. En effet, même si nous avons développé une approche générique et fait en sorte de démontrer la flexibilité de nos outils pour modéliser des objectifs très liés au système cible d'une part, et très méta-cyber (discrétion) d'autre part, nous sommes conscients qu'il existe des systèmes ou des objectifs qui seraient difficiles à modéliser avec cette approche. De plus, la modélisation et l'outillage des stratégies d'attaque est une approche relativement originale qui rend la comparaison avec d'autres solutions non triviale. Ceci demanderait un travail à part entière associant des capacités de modélisation et d'abstraction à des expertises fortement liées au domaine métier.

Plus généralement, l'approche que nous proposons est opérationnelle, comme le montre le cas d'étude. Toutefois, l'outillage requiert plusieurs efforts de modélisation manuelle du fait du manque de syntaxe abstraite dans le modèle de stratégie. La modélisation des obstacles et des opportunités que nos outils offrent est un exercice manuel à l'heure actuelle. De plus, la phase d'analyse se déroule en construisant à la main différentes propriétés mettant en jeu différentes opportunités vis-à-vis de différents objectifs. Pour cela, nous avons commencé à identifier des patterns d'opportunités. En fonction de l'obstacle rencontré par

l'attaquant, il existe plusieurs possibilités d'opportunités-types, comme le fait d'empêcher un comportement de se produire ou le fait d'induire un nouveau comportement. L'introduction de patterns pourrait permettre à la fois d'enrichir notre méthodologie et de constituer une première étape vers la conceptualisation du modèle de stratégie.

4.3 Conclusion

La méthodologie que nous proposons adapte l'Operational Design au contexte de la cyber-sécurité des systèmes industriels pour mieux comprendre comment une APT conçoit sa stratégie d'attaque.

Pour valider notre méthodologie, nous étudions le cas d'étude d'une station de pompage d'eau dans la Section 4.1. Lors de cette étude, nous mettons concrètement en application la méthodologie pour montrer comment un attaquant conçoit sa stratégie d'attaque pour atteindre ses objectifs.

Ensuite nous proposons une évaluation globale de notre méthodologie et de nos outils dans la Section 4.2. Le cas d'étude illustre les différents points forts de la fédération de modèles. Le bilan général est positif puisque la plupart des exigences introduites dans le Chapitre 2 (modélisation de système, de stratégie, gestion de l'interopérabilité et analyse) sont remplies. Il existe toutefois quelques pistes d'améliorations notamment au niveau de la modélisation d'une stratégie plus fine (4.2.3).

Chapitre 5

Conclusion

La place des systèmes industriels et plus généralement cyber-physiques dans la société actuelle est de plus en plus importante. Évoluant rapidement au rythme des avancées technologiques, leur fonctionnement se fait de plus en plus complexe. Notre dépendance à leur égard fait de l'industrie une cible majeure pour les attaquants du monde cyber. La cyber-sécurité des systèmes industriels est donc d'un enjeu capital auquel nous répondons dans ce manuscrit.

Le très vaste domaine de la cyber-sécurité des systèmes industriels est en continuelle expansion. Aussi, nous nous limitons à l'étude d'un type spécifique de menace, les menaces persistantes avancées ou APT. Ces menaces sont des attaquants ayant recours à des stratégies développées et d'énormes moyens. Leur mode opératoire complexe est un challenge pour les acteurs de la cyber-sécurité. Bien qu'il existe de nombreuses solutions pour se défendre contre ce type de menaces, elles ne répondent que partiellement au problème.

Dans ce manuscrit, nous proposons de prendre du recul vis-à-vis des APT en étudiant spécifiquement comment les APT conçoivent leur stratégie. Ce faisant, nous espérons donner un nouveau regard pour mieux comprendre ces menaces et ainsi proposer de nouveaux moyens de défense.

5.1 Objectifs et problématiques

L'état de l'art du Chapitre 2 montre que les menaces persistantes avancées sont un réel danger pour les systèmes industriels. Il est difficile de se prémunir contre les stratégies que ce type d'attaquant peut mettre en place du fait de leur complexité. Bien qu'il existe des solutions de sécurité pour se défendre contre des attaques connues, ces attaquants font preuve de discrétion et emploient des vulnérabilités de type zero-day ainsi que des moyens très divers pour parvenir à leurs objectifs, quitte à entreprendre des campagnes de longue durée.

C'est pourquoi nous proposons de prendre du recul sur les solutions de sécurité existantes. Pour mieux se défendre contre les APT, il est primordial de mieux comprendre leur mode de fonctionnement, autrement dit de comprendre comment les APT structurent leurs attaques. Pour cela, nous abordons deux problématiques dans ce manuscrit.

La première problématique de ce manuscrit est de comprendre le point de vue d'un attaquant de type APT sur le système cible. La modélisation est une approche qui demande un effort supplémentaire de la part du concepteur pour faire ses analyses au lieu de simplement tester et analyser directement le système. Aussi, il est essentiel de justifier l'effort de modélisation en proposant un ensemble d'outils fonctionnels qui permettent l'exécution et l'analyse. On peut parler de "méthode agile" dans la mesure où des objectifs sont identi-

fiés, puis on modélise et on analyse les travaux au regard de ces objectifs tout au long de la démarche. Nous identifions différents objectifs liés à cette problématique :

— une méthodologie, c'est-à-dire une marche à suivre pour capturer le point de vue de l'attaquant sur le système.

— une syntaxe abstraite, c'est-à-dire un ensemble structuré de concepts permettant de modéliser le point de vue de l'attaquant.

— une syntaxe concrète, c'est-à-dire une représentation de ces concepts qui soit manipulable par l'utilisateur.

— une sémantique bien définie, c'est-à-dire le sens qui est donné à chaque concept et l'interprétation qu'il doit avoir pour le concepteur d'une part et pour la machine d'autre part.

La seconde problématique de ce manuscrit est de comprendre la stratégie d'un attaquant de type APT. De même que pour la première problématique, nous répondons à cette problématique avec un ensemble d'outils fonctionnels dans une démarche qui se veut "agile".

Les différents objectifs liés à cette problématique que nous avons identifiés sont identiques aux objectifs de la première problématique à savoir : la définition d'une méthodologie, d'une syntaxe abstraite, d'une syntaxe concrète et d'une sémantique. Pour cet aspect de sémantique, la définition d'une sémantique dynamique requiert une attention particulière.

Dans ce travail, notre solution de modélisation reposant sur la fédération de modèles, nous nous appuyons sur OpenFlexo. Cet environnement permet de fédérer des modèles hétérogènes, ce qui est nécessaire pour mettre en place notre méthodologie qui fait intervenir trois phases (spécification, modélisation et analyse) et de nombreux artefacts de modélisation différents.

Dans la suite de la conclusion, nous récapitulons les différentes contributions apportées pour répondre à nos deux problématiques.

Les travaux exposés dans ce manuscrit ont fait l'objet de trois publications exposant le point de vue de l'APT sur le système [92] et la stratégie de l'attaquant [93, 94].

— [92] Tithnara N. Sun, Bastien Drouot, Fahad Golra, Joël Champeau, Sylvain Guérin, Luka Le Roux, Raúl Mazo, Ciprian Teodorov, Lionel Van Aertyck, and Bernard L'Hostis. *A domain-specific modeling framework for attack surface modeling*. In International Conference on Information Systems Security and Privacy, 2020.

— [93] Tithnara N. Sun, Luka Le Roux, Ciprian Teodorov, and Philippe Dhaussy. *Exploration de scénarios de systèmes cyber-physiques pour l'analyse de la menace*. In Actes des 19èmes journées sur les Approches Formelles dans l'Assistance au Développement de Logiciels, pages 17–24, 2020.

— [94] Tithnara N. Sun, Ciprian Teodorov, and Luka Le Roux. *Operational design for advanced persistent threats*. In the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems : Companion Proceedings, page 1-10, 2020.

5.2 Contributions

Ce manuscrit présente les trois contributions majeures de nos travaux : l'adaptation de l'*Operational Design* au contexte de la cyber-sécurité, la création d'un framework de fédération entièrement outillé pour la modélisation de stratégie d'APT et la création d'un langage de système dynamique pour la modélisation du point de vue de l'attaquant sur le système.

À partir de l'étude des menaces persistantes avancées, nous avons conclu que leur mode opératoire nécessitait une phase de reconnaissance et d'armement préalable [13]. Pour mieux comprendre cette phase, nous avons considéré le point de vue de l'APT sur le système ciblé. Le mode opératoire des APT est analogue à celui des stratégies militaires. C'est pourquoi notre première contribution consiste à adapter la méthodologie militaire de l'*Operational Design* pour adopter le point de vue de l'APT sur sa cible. Présentée dans le Chapitre 3, la méthodologie que nous proposons est centrée autour de la mission de l'APT. Nous définissons la mission de l'APT de la manière suivante : *but à atteindre sur un système cible spécifique comprenant un ou plusieurs objectifs et rencontrant possiblement plusieurs obstacles*.

À partir de cette définition, propre à notre contexte d'étude, nous identifions les trois axes selon lesquels l'APT construit sa stratégie : le système ciblé, les objectifs et les obstacles. Dans la Section 3.1, nous présentons en détail la méthodologie de l'APT pour concevoir sa stratégie. Celle-ci suit trois phases : spécification, modélisation et analyse. Dans la phase de spécification, l'attaquant identifie les éléments de la mission selon les trois axes (système, objectifs et obstacles). Il construit ensuite un modèle de données rassemblant ces éléments reliés à différentes ressources documentaires qui l'aideront dans les phases suivantes. Dans la phase de modélisation, l'attaquant modélise le système ciblé, les objectifs de la mission et les obstacles qui l'entravent. Ces différents processus de modélisation peuvent mettre en jeu différents artefacts de modélisation dont il est important de gérer la synchronisation afin de développer des modèles cohérents entre eux. Dans la phase d'analyse, à partir des modèles créés, l'attaquant conçoit la stratégie opérationnelle qui lui permet de mener à bien sa mission. Nous proposons une méthodologie abstraite qui permet de capturer la stratégie de l'attaquant, répondant ainsi au premier et deuxième objectifs de notre seconde problématique, la méthodologie de modélisation de stratégie et sa syntaxe abstraite. Dans l'esprit, cette méthodologie peut s'adapter à tous les niveaux d'abstraction voulus par l'attaquant. Cependant, la force de la modélisation réside dans son abstraction de la réalité pour simplifier les analyses. Aussi, il est intéressant de garder un haut niveau d'abstraction dans les modèles utilisés dans cette méthodologie.

La méthodologie abstraite que nous proposons n'impose pas de langage de modélisation particulier, cependant nous avons choisi de l'implémenter dans une méthodologie concrète dans différents langages détaillés dans ce manuscrit. Pour cela, nous avons implémenté la fédération de modèles afin de proposer un outillage complet permettant de mener concrètement et entièrement la méthodologie sur des cas d'étude réels. Cette deuxième contribution présentée dans le Chapitre 3 répond au troisième objectif de la seconde problématique, la syntaxe concrète de stratégie. La fédération de modèles est mise en place dans l'environnement OpenFlexo. Les concepts fédérés permettent de maintenir la cohérence entre les différents paradigmes de modélisation choisis. De plus, elle facilite l'extension de l'approche à de nouveaux paradigmes ainsi que la réutilisation des travaux dans d'autres contextes. Le Chapitre 3 détaille l'implémentation de chaque processus de la méthodologie dans le framework. Nous proposons un framework de stratégie de l'APT mettant en œuvre la méthodologie de l'Operational Design dans le contexte des APT.

Ce framework repose essentiellement sur le modèle de système tel qu'il est perçu par l'attaquant. Aussi, le modèle de système ciblé est une contribution majeure de nos travaux. Le langage Pimca modélise les systèmes pour permettre différentes analyses de sécurité, comme l'analyse de surface d'attaque ou le développement de modèle d'attaquant. Cependant, il ne capture pas certains aspects dynamiques nécessaires au développement de stratégie. De plus, il est dépourvu d'un outillage concret adapté à notre contexte de fédération et d'exécution. C'est pourquoi notre troisième contribution, présentée dans la Section 3.3 consiste en la création du langage DyPimca comme extension du langage Pimca pour la

modélisation de systèmes du point de vue d'une menace persistante avancée. DyPimca est un langage prenant en compte l'aspect structurel du système avec les concepts de Pimca et l'aspect comportemental du système avec de nouveaux concepts. Nous définissons la syntaxe abstraite et deux syntaxes concrètes intégrées à notre framework, ainsi qu'une sémantique d'exécution. Cette contribution répond donc intégralement à la première problématique de la thèse, la modélisation du point de vue de l'APT sur le système.

Enfin le Chapitre 4 nous permet de valider notre approche sur un cas d'étude. Nous illustrons l'ensemble de la méthodologie dans ce chapitre en liant toutes nos contributions. La mission est de faire déborder le réservoir d'une station de pompage. Tout d'abord, nous spécifions la mission au regard des trois axes du point de vue de l'attaquant : le système, les objectifs et les obstacles. Ensuite, dans la phase de modélisation, nous construisons le modèle de système conformément aux données rassemblées dans la phase de spécification. Nous construisons conjointement le modèle d'objectifs et d'obstacles afin d'enrichir le modèle de système. A noter que l'objectif d'assurer une discrétion durant l'attaque est assez générique pour être réutilisé dans d'autres cas d'étude. Enfin, la phase d'analyse utilise la technique de vérification formelle de modèles pour aboutir à une stratégie opérationnelle permettant à l'attaquant de remplir sa mission.

5.3 Perspectives

Les travaux présentés dans ce manuscrit répondent aux problématiques identifiées dans l'état de l'art du Chapitre 2, à savoir comprendre le point de vue de l'APT sur le système et l'élaboration de stratégie d'attaque. Nous avons cependant identifié certains aspects qui gagneraient à être approfondis dans des travaux futurs. Ces perspectives d'évolution concernent la compréhension de la stratégie d'une menace persistante avancée et l'évaluation de la méthodologie globale.

Nos travaux permettent de capturer la stratégie d'une APT. Toutefois, à des fins de défense, il semble judicieux de capitaliser davantage sur nos résultats et de poursuivre notre démarche pour mieux comprendre les stratégies d'une APT. Aussi, la création d'un langage de stratégie semble être la prochaine étape naturelle de nos travaux. Les stratégies que notre méthodologie modélisent sont des exploitations successives d'opportunités pour passer au travers des obstacles. C'est pourquoi il peut être intéressant d'identifier des patrons d'opportunités exploitant un obstacle. On constate d'ores et déjà dans le cadre de nos travaux qu'il existe plusieurs moyens de contourner un obstacle. Par exemple, l'attaquant peut induire un comportement non-voulu du système ou empêcher un comportement normal du système de se produire. L'identification de ces patrons amèneraient une compréhension plus fine de la stratégie de l'attaquant, ce qui permettrait à terme de créer un langage de stratégie et donc de modéliser des stratégies plus complexes. Nous avons engagé des travaux [63] dans cette perspective.

Pour poursuivre l'évaluation de notre méthodologie, il semble également naturel de traiter d'autres cas d'étude du domaine. En effet, la méthodologie concrète que nous avons choisie s'adapte tout particulièrement à notre cas d'étude. Il serait intéressant d'évaluer à quel point cette méthodologie concrète s'adapte à d'autres exemples. Il est également crucial d'évaluer dans quelle mesure il est nécessaire de changer de paradigmes de modélisation pour d'autres cas d'études. Ceci permettrait à terme de faire une étude comparative avec d'autres approches plus classiques. En effet, la perspective à très haut niveau d'abstraction de notre approche rend la comparaison avec les travaux existants difficile. D'autant plus que nous nous focalisons exclusivement sur la phase de reconnaissance et d'armement, qui reste très peu étudiée dans la littérature.

L'approche que nous présentons dans ce manuscrit offre potentiellement d'autres perspectives. En effet, la cyber-sécurité contre les menaces persistantes avancées est un domaine en perpétuelle évolution à mesure que les techniques d'attaques se complexifient et que la société dépend de plus en plus des systèmes à défendre. La défense parfaite et exhaustive des systèmes contre ces menaces n'est pas atteignable. C'est pourquoi nous soutenons que la modélisation à haut niveau d'abstraction est un nouvel outil indispensable à la défense de tels systèmes critiques.

Liste des publications

Publications internationales

— [92] Tithnara N. Sun, Bastien Drouot, Fahad Golra, Joël Champeau, Sylvain Guérin, Luka Le Roux, Raúl Mazo, Ciprian Teodorov, Lionel Van Aertyck, and Bernard L'Hostis. *A domain-specific modeling framework for attack surface modeling*. In International Conference on Information Systems Security and Privacy, 2020.

— [94] Tithnara N. Sun, Ciprian Teodorov, and Luka Le Roux. *Operational design for advanced persistent threats*. In the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems : Companion Proceedings, pages 1-10, 2020.

Autres publications

— [93] Tithnara N. Sun, Luka Le Roux, Ciprian Teodorov, and Philippe Dhaussy. *Exploration de scénarios de systèmes cyber-physiques pour l'analyse de la menace*. In Actes des 19èmes journées sur les Approches Formelles dans l'Assistance au Développement de Logiciels, pages 17–24, 2020.

— Poster. Tithnara N. Sun, Ciprian Teodorov, Joël Champeau, and Philippe Dhaussy. *Dynamic system modeling for threat analysis*. Séminaire MOCS, 2018.

— Poster. Tithnara N. Sun, Bastien Drouot, Fahad Golra, Joël Champeau, Sylvain Guérin, Luka Le Roux, Raúl Mazo, Ciprian Teodorov, Lionel Van Aertyck, and Bernard L'Hostis. *A domain-specific modeling framework for attack surface modeling*. In International Conference on Information Systems Security and Privacy, 2020.

— Présentation. Tithnara N. Sun. *Pimca, modélisation système pour la cybersécurité*. 14/05/2020.

Bibliographie

[1] Élicitation. URL https://www.lalanguefrancaise.com/dictionnaire/definition/elicitation. Consulté le 08/07/2021.

[2] Information technology—security techniques—information security management systems—overview and vocabulary. *ISO/IEC 27000 :2018*, pages 1–38, février 2018.

[3] Martín ABADI et Leslie LAMPORT : The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991. ISSN 0304-3975.

[4] Walid AL-AHMAD : A detailed strategy for managing corporation cyber war security. *International Journal of Cyber-Security and Digital Forensics*, 2(4):1–9, 2013.

[5] Annie AUTHOSSERRE-CAVARERO, Frédéric BERTRAND, Mireille FORNARINO, Philippe COLLET, Hubert DUBOIS, Stéphane DUCASSE, Sophie DUPUY-CHESSA, Catherine FARON ZUCKER, Cyril FAUCHER, Jean-Yves LAFAYE, Philippe LAHIRE, Olivier LE GOAER, Johan MONTAGNAT et Anne-Marie PINNA-DÉRY : Ingénierie dirigée par les modèles : quels supports à l'interopérabilité des systèmes d'information ? *Revue des Sciences et Technologies de l'Information*. URL https://hal.inria.fr/hal-00813675.

[6] Marco AUTILI, Paola INVERARDI et Patrizio PELLICCIONE : Graphical scenarios for specifying temporal properties : An automated approach. *Autom. Softw. Eng.*, 14:293–340, septembre 2007.

[7] Sean BECHHOFER, Frank van HARMELEN, Jim HENDLER, Ian HORROCKS, Deborah MCGUINNESS, Peter PATEL-SCHNEIJDER et Lynn Andrea STEIN : OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium (W3C), février 2004. URL http://www.w3.org/TR/owl-ref/.

[8] Sabri BENDRISS, Abdelatif BENABDELHAFID, Jaouad BOUKACHOUR et Dalila BOUDEBOUS : Meta-modèle de référence holonique pour la gestion de la traçabilité du produit dans la chaîne logistique. *In 5ème Colloque International Conception et Production Intégrées CPI*, Rabat, Maroc, 2007.

[9] Valentin BESNARD : *EMI - Une approche pour unifier l'analyse et l'exécution embarquée à l'aide d'un interpréteur de modèles pilotable : application aux modèles UML des systèmes embarqués*. Thèse de doctorat, ENSTA Bretagne - École nationale supérieure de techniques avancées Bretagne, décembre 2020. URL https://tel.archives-ouvertes.fr/tel-03371484.

[10] Ross BREWER : Advanced persistent threats : minimising the damage. *Network security*, 2014(4):5–9, 2014.

[11] Mihal BRUMBULLI, Emmanuel GAUDIN et Ciprian TEODOROV : Automatic Verification of BPMN Models. *In 10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, 2020. URL https://hal.archives-ouvertes.fr/hal-02441878.

[12] CANADIAN JOINT OPERATIONS HEADQUARTERS : Systemic operational design : Freeing operational planning from the shackles of linearity. *Canadian military journal*, 9(4), 2009.

[13] Ping CHEN, Lieven DESMET et Christophe HUYGENS : A Study on Advanced Persistent Threats. *In* Bart DECKER et André ZÚQUETE, éditeurs : *15th IFIP International Conference on Communications and Multimedia Security (CMS)*, volume LNCS-8735 de *Communications and Multimedia Security*, pages 63–72, Aveiro, Portugal, septembre 2014. Springer. URL https://hal.inria.fr/hal-01404186. Part 2 : Work in Progress.

[14] Kim-Kwang Raymond CHOO : Organised crime groups in cyberspace : A typology. trends in organized crime, 11, 270-295. *Trends in Organized Crime*, 11:270–295, septembre 2008.

[15] Edmund M. CLARKE, Jr., Orna GRUMBERG, Daniel KROENING, Doron PELED et Helmut VEITH : *Model checking*. MIT press, 2018.

[16] Benoît COMBEMALE : Ingénierie Dirigée par les Modèles (IDM) – État de l'art. URL https://hal.archives-ouvertes.fr/hal-00371565. 2008.

[17] Stephen A. COOK : The complexity of theorem-proving procedures. *In Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[18] Nancy J. COOKE : Knowledge elicitation. *Handbook of applied cognition*, pages 479–509, 1999.

[19] Patrick COUSOT et Radhia COUSOT : Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints. *In Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252, 1977.

[20] Michael K. DALY : Advanced persistent threat. *Usenix, Nov*, 4(4):2013–2016, 2009.

[21] Direction Interministérielle des Systèmes d'Information et de COMMUNICATION : Référentiel général d'interopérabilité version 2.0. Rapport technique, Secrétariat général pour la modernisation de l'action publique, 2015. URL https://references.modernisation.gouv.fr/sites/default/files/Referentiel_General_Interoperabilite_V2.pdf.

[22] Edsger W. DIJKSTRA : Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, 1975. ISSN 0001-0782.

[23] Zdena DOBESOVA : Using the physic of notation to analyse modelbuilder diagrams. volume 1, pages 595–602, juin 2013.

[24] Bastien DROUOT : *Fédération de modèles pour l'analyse de cybersécurité du point de vue d'un attaquant.* Thèse de doctorat, École Nationale Supérieure de Techniques Avancées Bretagne, avril 2021. URL https://hal.archives-ouvertes.fr/tel-03275242.

[25] Bastien DROUOT et Joël CHAMPEAU : Model federation based on role modeling. *In MODELSWARD*, pages 72–83, 2019.

[26] Dale C. EIKMEIER : Redefining the center of gravity. Rapport technique, NATIONAL DEFENSE UNIV WASHINGTON DC, 2010.

[27] E. Allen EMERSON : Temporal and modal logic. *In Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.

[28] Anna FENSEL et Uwe KELLER : Choosing an ontology language. pages 47–50, janvier 2005.

[29] Florent FORTAT, Maryline LAURENT et Michel SIMATIC : Games based on active nfc objects : Model and security requirements. *In Proceedings of the 2015 International Workshop on Network and Systems Support for Games*, NetGames '15. IEEE Press, 2015. ISBN 9781509000685.

[30] Ivo FRIEDBERG, Florian SKOPIK, Giuseppe SETTANNI et Roman FIEDLER : Combating advanced persistent threats : From network event correlation to incident detection. *Computers & Security*, 48:35–57, 2015.

[31] Ibrahim GHAFIR et Vaclav PRENOSIL : Advanced persistent threat and spear phishing emails. *In Proceedings of International Conference on Distance Learning, Simulation and Communication. University of Defence*, pages 34–41, 2015.

[32] Andrew GINTER : The Top 20 Cyberattacks on Industrial Control Systems. Rapport technique, Waterfall Security Solutions, 2017.

[33] Michael GLASSMAN et Min Ju KANG : Intelligence in the internet age : The emergence and evolution of open source intelligence (osint). *Computers in Human Behavior*, 28(2):673–682, 2012.

[34] Fahad GOLRA, Joël CHAMPEAU et Ciprian TEODOROV : *Early Validation Framework for Critical and Complex Process-Centric Systems*. janvier 2019.

[35] Fahad R. GOLRA, Antoine BEUGNARD, Fabien DAGNAT, Sylvain GUERIN et Christophe GUYCHARD : Addressing modularity for heterogeneous multi-model systems using model federation. MODULARITY Companion 2016, pages 206–211, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340335.

[36] Fahad R. GOLRA, Antoine BEUGNARD, Fabien DAGNAT, Sylvain GUERIN et Christophe GUYCHARD : Addressing modularity for heterogeneous multi-model systems using model federation. *In Companion Proceedings of the 15th International Conference on Modularity*, pages 206–211, 2016.

[37] Fahad R. GOLRA, Antoine BEUGNARD, Fabien DAGNAT, Sylvain GUERIN et Christophe GUYCHARD : Using free modeling as an agile method for developing domain specific modeling languages. *In Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pages 24–34, 2016.

[38] Fahad Rafique GOLRA, Fabien DAGNAT, Jeanine SOUQUIÈRES, Imen SAYAR et Sylvain GUÉRIN : Bridging the Gap Between Informal Requirements and Formal Specifications Using Model Federation. *In* Ina Schaefer EINAR BROCH JOHNSEN, éditeur : *16th International Conference on Software Engineering and Formal Methods*

(SEFM 2018), volume 10886 de *Software Engineering and Formal Methods 16th International Conference, SEFM 2018, Held as Part of STAF 2018, Toulouse, France, June 27–29, 2018, Proceedings*, pages 54–69, Toulouse, France, juin 2018. Springer. URL https://hal.archives-ouvertes.fr/hal-01853610.

[39] Kerim GOZTEPE : Designing fuzzy rule based expert system for cyber security. *International Journal of Information Security Science*, 1:13–19, janvier 2012.

[40] Kerim GOZTEPE, Recep KILIC et Alper KAYAALP : Cyber defense in depth : Designing cyber security agency organization for turkey. *Journal of Naval Science and Engineering*, 10:1–24, novembre 2014.

[41] Thomas GRAVES et Bruce E. STANLEY : Design and operational art : A practical approach to teaching the army design methodology. *Military Review*, 93(4):53, 2013.

[42] GROUPE DE TRAVAIL INTEROPÉRABILITÉ DE L'ASSOCIATION FRANCOPHONE DES UTILISATEURS DE LOGICIELS LIBRES (AFUL) : Définition de l'Interopérabilité, 2010. URL http://definition-interoperabilite.info/.

[43] Christophe GUYCHARD, Sylvain GUERIN, Ali KOUDRI, Antoine BEUGNARD et Fabien DAGNAT : Conceptual interoperability through models federation. *In Semantic Information Federation Community Workshop*, page 23, 2013.

[44] Thoufique HAQ, Jinjian ZHAI et Vinay K PIDATHALA : Advanced persistent threat (APT) detection center, avril 2017. US Patent 9,628,507.

[45] Hiba HNAINI, Luka LE ROUX, Joël CHAMPEAU et Ciprian TEODOROV : Security property modeling. *In 7th International Conference on Information Systems Security and Privacy ICISSP*, pages 694–701, 2021.

[46] Eric M. HUTCHINS, Michael J. CLOPPERT et Rohan M. AMIN : Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[47] John HUTCHINSON, Jon WHITTLE, Mark ROUNCEFIELD et Steinar KRISTOFFERSEN : Empirical assessment of mde in industry. *In Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 471–480, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450304450.

[48] Vinay M. IGURE, Sean A. LAUGHTER et Ronald D. WILLIAMS : Security issues in SCADA networks. *Computers & Security*, 25(7):498–506, octobre 2006.

[49] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO) : ISO 9001 :2015 – Systèmes de management de la qualité – Exigences. Standard, International Organization for Standardization, 2015.

[50] ISACA : Advanced persistent threat awareness study results. Rapport technique, Trend Micro, 2013.

[51] Isabelle JEUGE-MAYNART : Larousse, novembre 2021. URL https://www.larousse.fr/.

[52] Muhammer KARAMAN, Hayrettin CATALKAYA, Ahmet Zeki GEREHAN et Kerim GOZTEPE : Cyber operation planning and operational design. *International Journal of Cyber-Security and Digital Forensics*, 5:21+, 2020/4/22/ 2016.

[53] KASPERSKY : The icefog APT : A tale of cloak and three daggers. Rapport technique, Kaspersky, 2013.

[54] David KENNEDY, Jim O'GORMAN, Devon KEARNS et Mati AHARONI : *Metasploit : the penetration tester's guide*. No Starch Press, 2011.

[55] KEYENCE : La « traçabilité » de A à Z ! Les bases de la traçabilité. URL https://www.keyence.fr/ss/products/marking/traceability/basic_about.jsp.

[56] Moonzoo KIM, Mahesh VISWANATHAN, Sampath KANNAN, Insup LEE et Oleg SOKOLSKY : Java-mac : A run-time assurance approach for java programs. *Formal methods in system design*, 24(2):129–155, 2004.

[57] Kenneth J. KNAPP, Thomas E. MARSHALL, R. Kelly RAINER et F. Nelson FORD : Information security : management's effect on culture and policy. *Information Management & Computer Security*, 2006.

[58] Barbara KORDY, Ludovic PIÈTRE-CAMBACÉDÈS et Patrick SCHWEITZER : DAG-based attack and defense modeling : Don't miss the forest for the attack trees. *Computer science review*, 13:1–38, 2014.

[59] Ralph LANGNER : Stuxnet : Dissecting a cyberwarfare weapon. *IEEE Security Privacy*, 9(3):49–51, 2011.

[60] Jean-Claude LAPRIE : Dependability : Basic concepts and terminology. *In Dependability : Basic Concepts and Terminology*, pages 3–245. Springer, 1992.

[61] Luka LE ROUX et Ciprian TEODOROV : Partially Bounded Context-Aware Verification. *In 17th International Conference on Software Engineering and Formal Methods, SEFM 2019*, volume LNCS, pages 532–548, Oslo, Norway, septembre 2019. URL https://hal.archives-ouvertes.fr/hal-02434620.

[62] Jay LEE, Behrad BAGHERI et Hung-An KAO : A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.

[63] Vincent LEILDÉ : *Aide au diagnostic de vérification formelle de systèmes*. Thèse de doctorat, ENSTA Bretagne - École nationale supérieure de techniques avancées Bretagne, novembre 2019. URL https://tel.archives-ouvertes.fr/tel-03172212.

[64] Frankie LI, Anthony LAI et DDL : Evidence of advanced persistent threat : A case study of malware for political espionage. *2011 6th International Conference on Malicious and Unwanted Software*, pages 102–109, 2011.

[65] Qiao LIANG et Wang XIANGSUI : *Unrestricted warfare*. Foreign Broadcast Information Service, 1999.

[66] Hung-Jen LIAO, Chun-Hung Richard LIN, Ying-Chih LIN et Kuang-Yuan TUNG : Intrusion detection system : A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[67] Simon LIU et Rick KUHN : Data loss prevention. *IT professional*, 12(2):10–13, 2010.

[68] Peri LOUCOPOULOS et Roberto ZICARI : *Conceptual Modeling, Databases, and Case : An Integrated View of Information Systems Development*. John Wiley and Sons, Inc., USA, 1992. ISBN 0471554626.

[69] Pratyusa K. MANADHATA et Jeannette M. WING : An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, 2011.

[70] MANDIANT, A FIREEYE COMPANY : M-Trends 2015. Rapport technique, Mandiant Consulting, 2015.

[71] Marvin MINSKY : Matter, mind and models. 1965.

[72] Mark A. MUSEN : The protégé project : a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.

[73] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) : Managing information security risk organization, mission, and information system view, mars 2011.

[74] Jörg NIEMÖLLER, Leonid MOKRUSHIN, Konstantinos VANDIKAS, Stefan AVESAND et Lars ANGELIN : Model federation and probabilistic analysis for advanced oss and bss. *In 2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies*, pages 122–129. IEEE, 2013.

[75] National Institute of STANDARDS et Technology (NIST) : Information technology security training requirements : A role- and performance-based model, 1998.

[76] OMG : Business process model and notation (BPMN) version 2.0.2. Standard, Object Management Group (OMG), décembre 2013. URL https://www.omg.org/spec/BPMN/2.0.2/PDF.

[77] Janis OSIS et Erika NAZARUKA : *Model-driven domain analysis and software development : Architectures and functions*. janvier 2010.

[78] Ing J. F. OVERBEEK : Meta object facility (MOF) : investigation of the state of the art. Mémoire de D.E.A., Citeseer, 2006.

[79] Celia PAULSEN et Robert D. BYERS : Glossary of key information security terms, 2019.

[80] Elisha PETERSON : Dagger : Modeling and visualization for mission impact situation awareness. *In MILCOM 2016-2016 IEEE Military Communications Conference*, pages 25–30. IEEE, 2016.

[81] Gary L. PETERSON : Myths about the mutual exclusion problem. 1981.

[82] Sophie PINCHINAT, Mathieu ACHER et Didier VOJTISEK : ATSyRa : an integrated environment for synthesizing attack trees. *In International Workshop on Graphical Models for Security*, pages 97–101. Springer, 2015.

[83] M. Zubair RAFIQUE, Ping CHEN, Christophe HUYGENS et Wouter JOOSEN : Evolutionary algorithms for classification of malware families through different network behaviors. *In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 1167–1174, 2014.

[84] Stefan RASS, Sandra KÖNIG et Stefan SCHAUER : Defending against advanced persistent threats using game-theory. *PloS One*, 12(1), 2017.

[85] Maud RIO : *A l'interface de l'ingénierie et de l'analyse environnementale, fédération pour une éco-conception proactive*. Thèse de doctorat, Centre de Recherches et d'Etudes Interdisciplinaires sur le Développement Durable (CREIDD), 2012.

[86] Marco ROCCHETTO et Nils Ole TIPPENHAUER : CPDY : extending the dolev-yao attacker with physical-layer interactions. *CoRR*, abs/1607.02562, 2016. URL http://arxiv.org/abs/1607.02562.

[87] Matthew SCHMID, Frank HILL et Anup K. GHOSH : Protecting data from malicious software. *In 18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pages 199–208. IEEE, 2002.

[88] Jean-Philippe SCHNEIDER : *Les rôles : médiateurs dynamiques entre modèles système et modèles de simulation*. Thèse de doctorat, Université de Bretagne occidentale - Brest, novembre 2015. URL https://tel.archives-ouvertes.fr/tel-01986294.

[89] Rupam Kumar SHARMA, Hemanta Kumar KALITA et Biju ISSAC : Different firewall techniques : A survey. *In Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2014.

[90] Anthony J. H. SIMONS : The theory of classification part 1 : perspectives on type compatibility. *Journal of Object Technology*, 1(1):55–61, 2002.

[91] Bastien SULTAN, Fabien DAGNAT et Caroline FONTAINE : A methodology to assess vulnerabilities and countermeasures impact on the missions of a naval system. *In Computer Security*, pages 63–76. Springer, 2017.

[92] Tithnara N. SUN, Bastien DROUOT, Joël CHAMPEAU, Fahad R. GOLRA, Sylvain GUÉRIN, Luka LE ROUX, Raul MAZO, Ciprian TEODOROV, Lionel VAN AERTRYCK et Bernard L'HOSTIS : A Domain-Specific Modeling Framework for Attack Surface Modeling. *In Proceedings of the 6th International Conference on Information Systems Security and Privacy - Volume 1 : ICISSP,*, pages 341–348. INSTICC, SciTePress, 2020. ISBN 978-989-758-399-5.

[93] Tithnara N. SUN, Luka LE ROUX, Ciprian TEODOROV et Philippe DHAUSSY : Exploration de scénarios de systemes cyber-physiques pour l'analyse de la menace. *In Actes des 19èmes journées sur les Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL 2020)*, pages 17–24, 2020.

[94] Tithnara N. SUN, Ciprian TEODOROV et Luka LE ROUX : Operational design for advanced persistent threats. *In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems : Companion Proceedings*, pages 1–10, 2020.

[95] Colin TANKARD : Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

[96] Ciprian TEODOROV, Philippe DHAUSSY et Luka LE ROUX : Environment-driven reachability for timed systems. *International Journal on Software Tools for Technology Transfer*, pages 1–17, avril 2017.

[97] Ciprian TEODOROV, Luka LE ROUX, Zoé DREY et Philippe DHAUSSY : Past-free [ze] reachability analysis : reaching further with dag-directed exhaustive state-space analysis. *Software Testing, Verification and Reliability*, 26(7):516–542, 2016.

[98] Christopher THEISEN, Nuthan MUNAIAH, Mahran AL-ZYOUD, Jeffrey C. CARVER, Andrew MENEELY et Laurie WILLIAMS : Attack surface definitions : A systematic literature review. *Information and Software Technology*, 104:94–103, 2018.

[99] US AIR FORCE : Annex 3-0 operations and planning, novembre 2016.

[100] US DEPARTMENT OF THE ARMY : TRADOC Pamphlet 525-7-8 Cyberspace operations concept capability plan 2016-2028, février 2010. URL https://fas.org/irp/doddir/army/pam525-7-8.pdf.

[101] US DEPARTMENT OF THE ARMY : ATP 5-0.1 Army design methodology, juillet 2015. URL https://armypubs.army.mil/epubs/DR_pubs/DR_a/pdf/web/atp5_0x1.pdf.

[102] US JOINT OPERATION PLANNING : Joint publication (JP) 5-0, joint planning. *Washington, DC : CJCS*, 26, 2006.

[103] US JOINT OPERATION PLANNING : Joint publication (JP) 2-01.3 joint intelligence preparation of the operational environment (JIPOE), 2014.

[104] US JOINT OPERATION PLANNING : Joint publication (JP) 1, doctrine for the armed forces of the united states. *Washington, DC : CJCS*, 2017.

[105] US JOINT STAFF : 7. planner's handbook for operational design, 2011.

[106] Parvathy VINOD, Vijay LAXMI et Manoj S. GAUR : Survey on malware detection methods. *In Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09)*, pages 74–79, 2009.

[107] Nikos VIRVILIS et Dimitris GRITZALIS : The big four - what we did wrong in advanced persistent threat detection? *In 2013 International Conference on Availability, Reliability and Security*, pages 248–254, 2013.

[108] Nikos VIRVILIS, Dimitris GRITZALIS et Theodoros APOSTOLOPOULOS : Trusted computing vs. advanced persistent threats : Can a defender win this game? *In 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, pages 396–403. IEEE, 2013.

[109] Peter WEGNER : Interoperability. *ACM Computing Surveys (CSUR)*, 28(1):285–287, 1996.

[110] Brett T. WILLIAMS : The joint force commander's guide to cyberspace operations. *Joint Force Quarterly*, 73(2):12–19, 2014.

[111] Spyros XANTHAKIS, Pascal RÉGNIER et Constantin KARAPOULIOS : *Le test des logiciels*. Hermes Science Publications, 2000.

Annexe A

Station de pompe à eau traitée en UPPAAL

A.1 Réservoir d'eau

A.2 PLC

A.3 Pompe

A.4 Vanne d'entrée

A.5 Capteur

A.6 SCADA

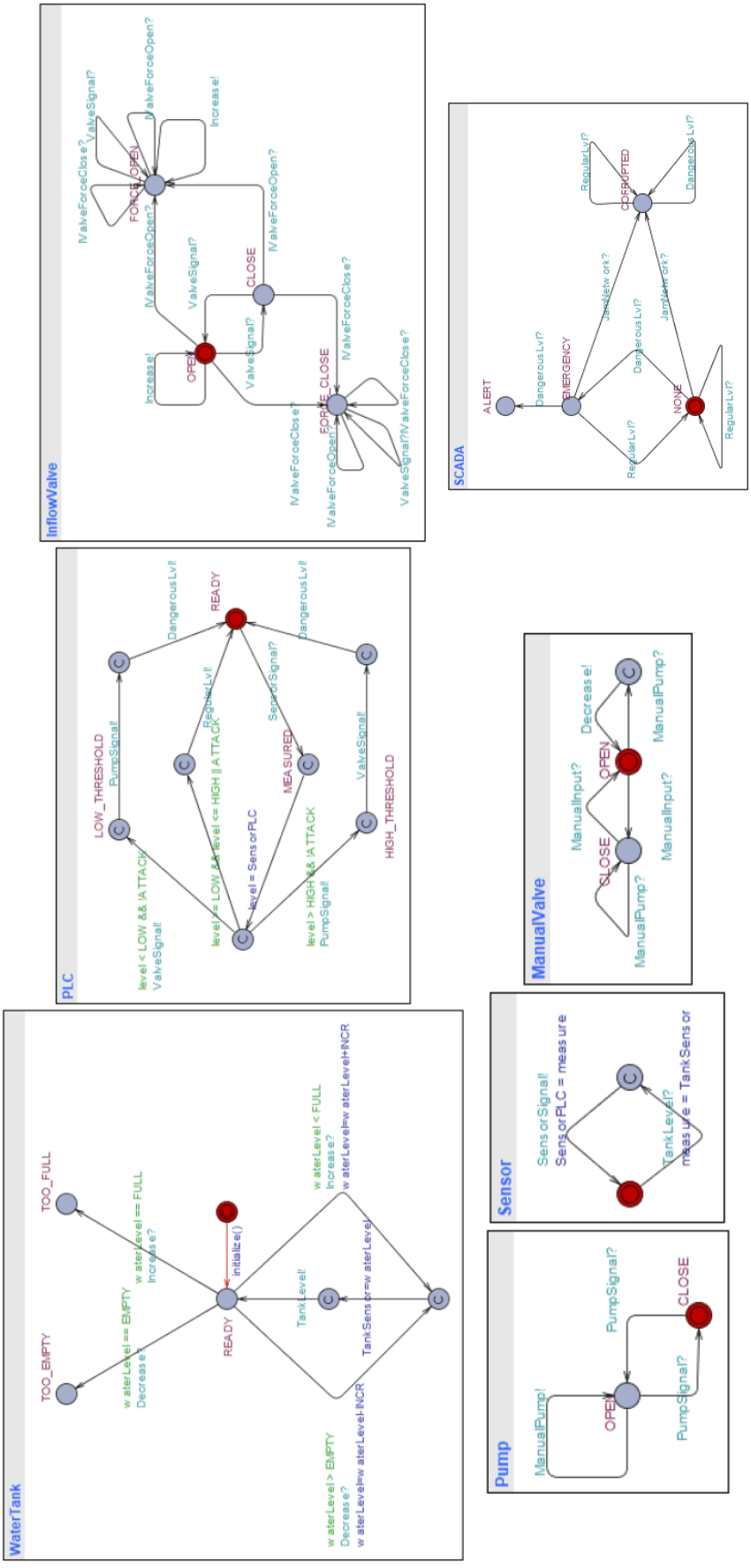


FIGURE A.1 – Ensemble des automates UPPAAL représentant le comportement de la station de pompe à eau

Annexe B

Diagramme de séquences de la méthodologie

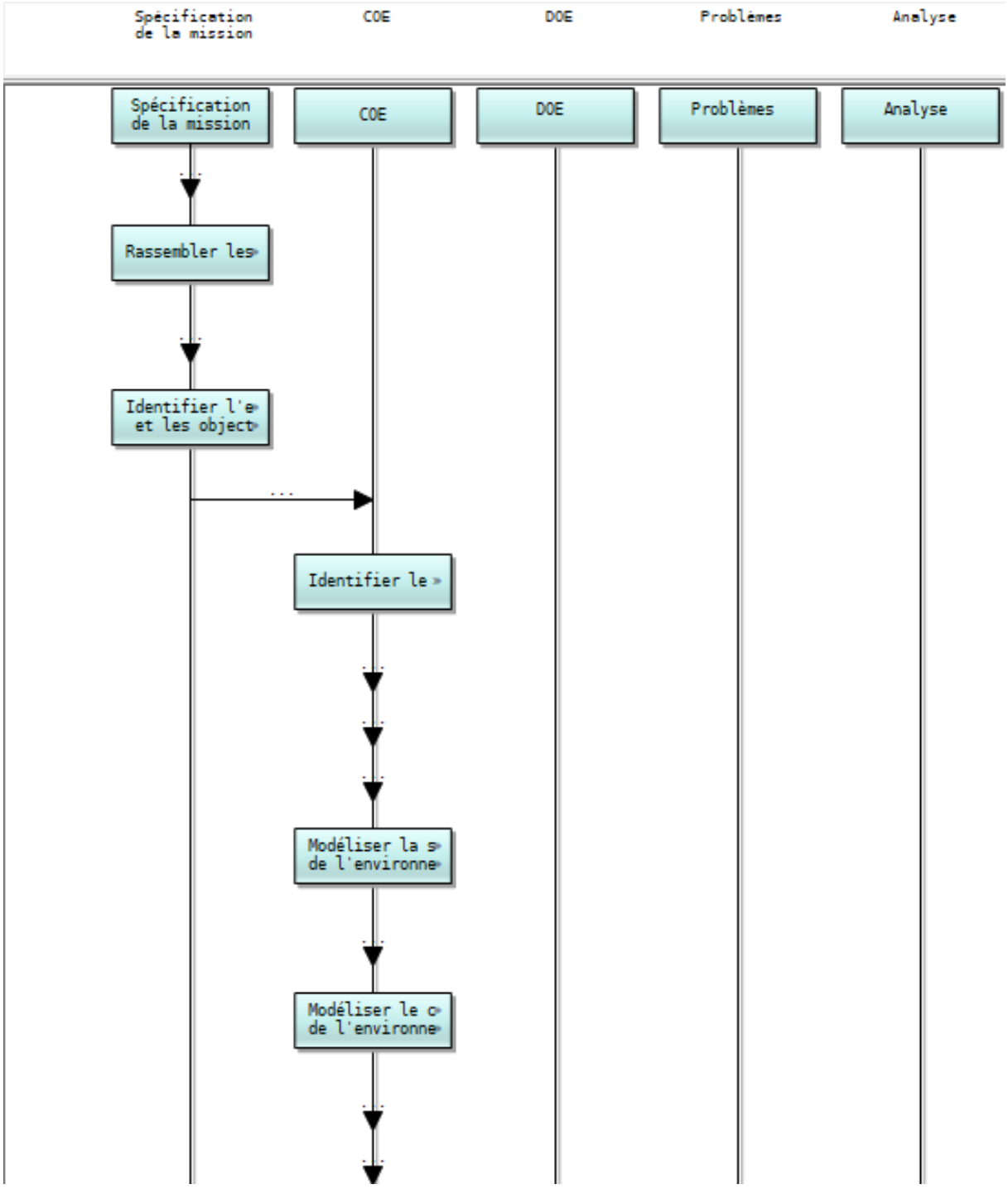


FIGURE B.1 – Diagramme de séquence d'une exécution du modèle BPMN global (1/5)

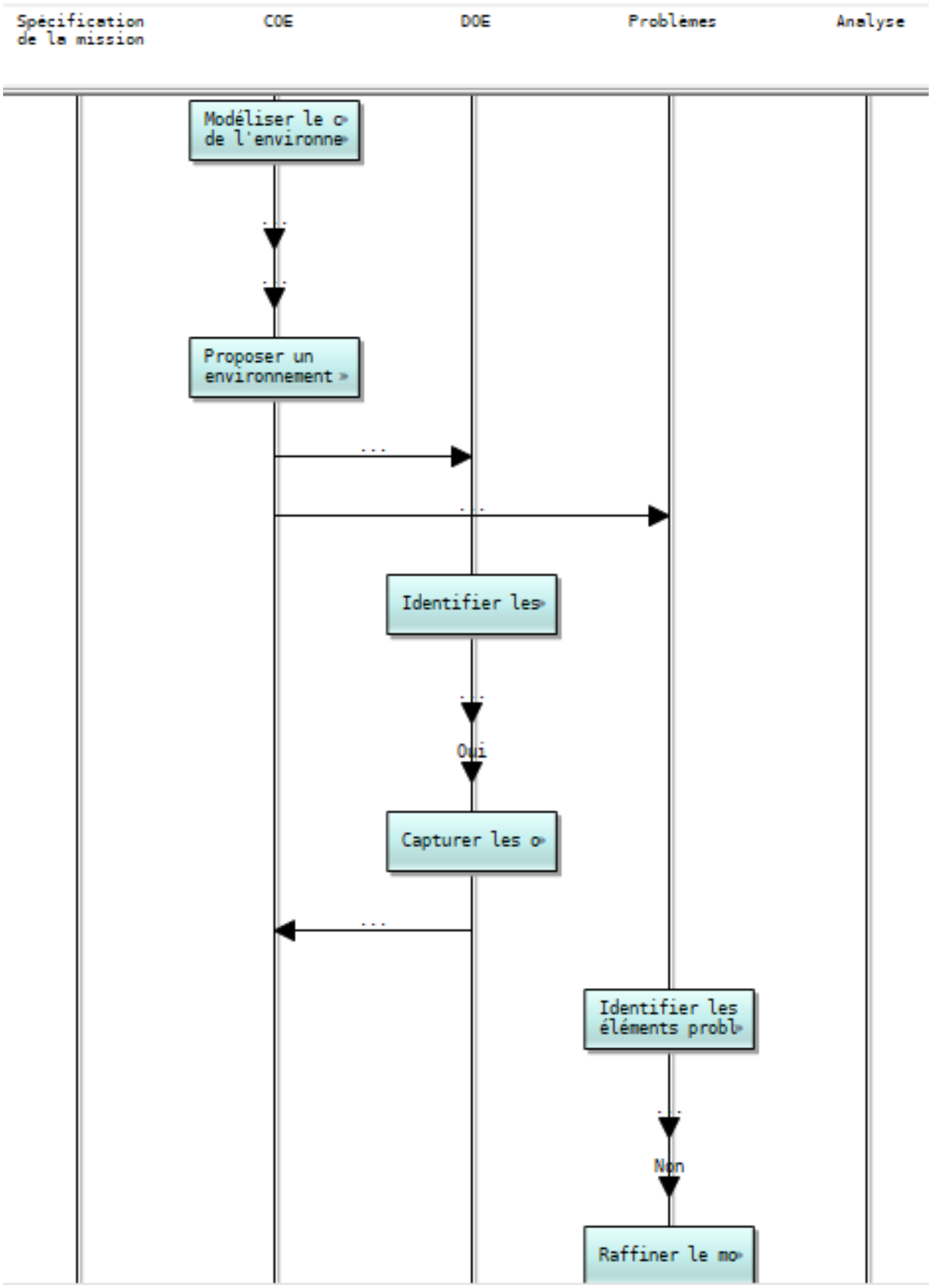


FIGURE B.2 – Diagramme de séquence d'une exécution du modèle BPMN global (2/5)

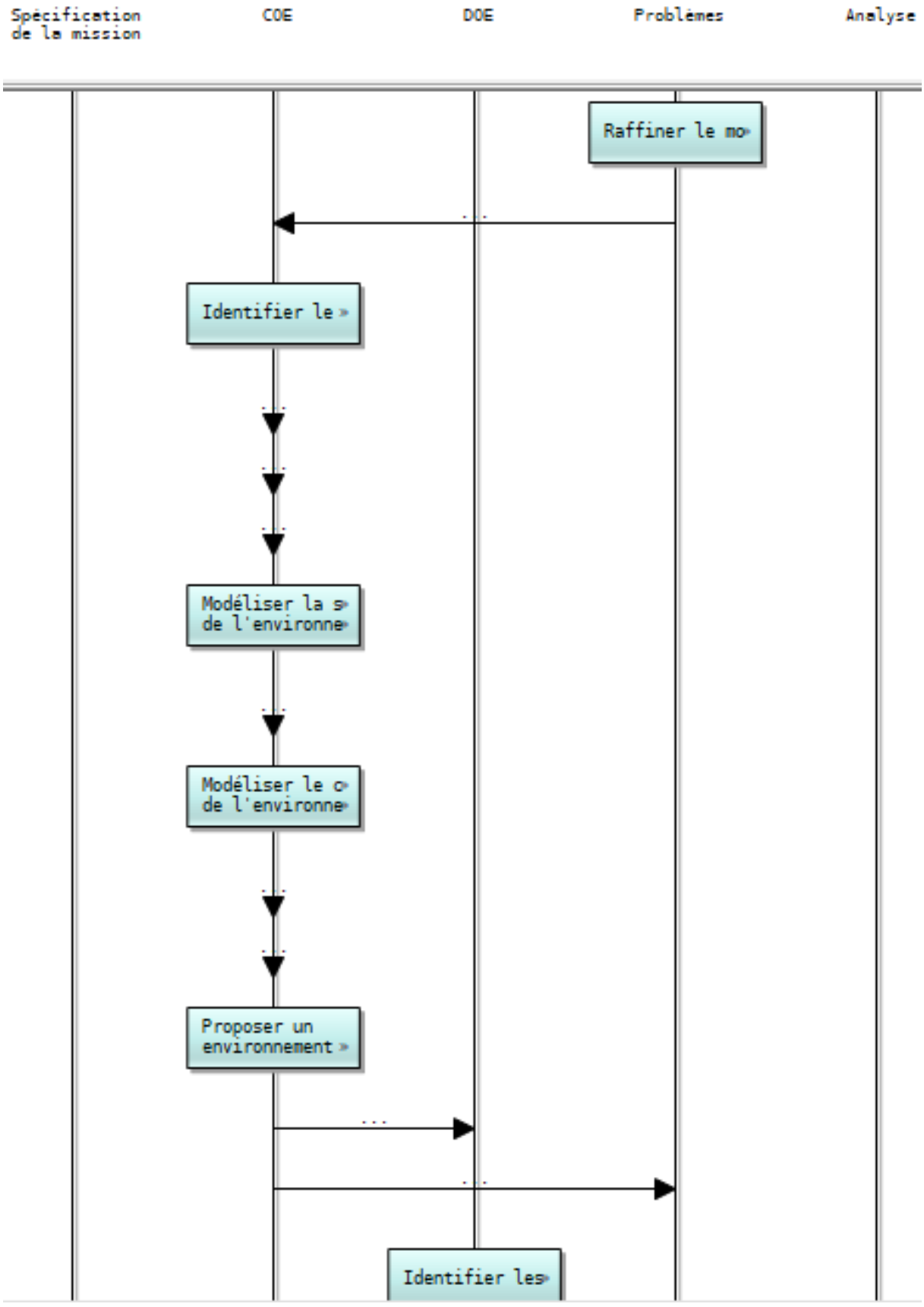


FIGURE B.3 – Diagramme de séquence d'une exécution du modèle BPMN global (3/5)

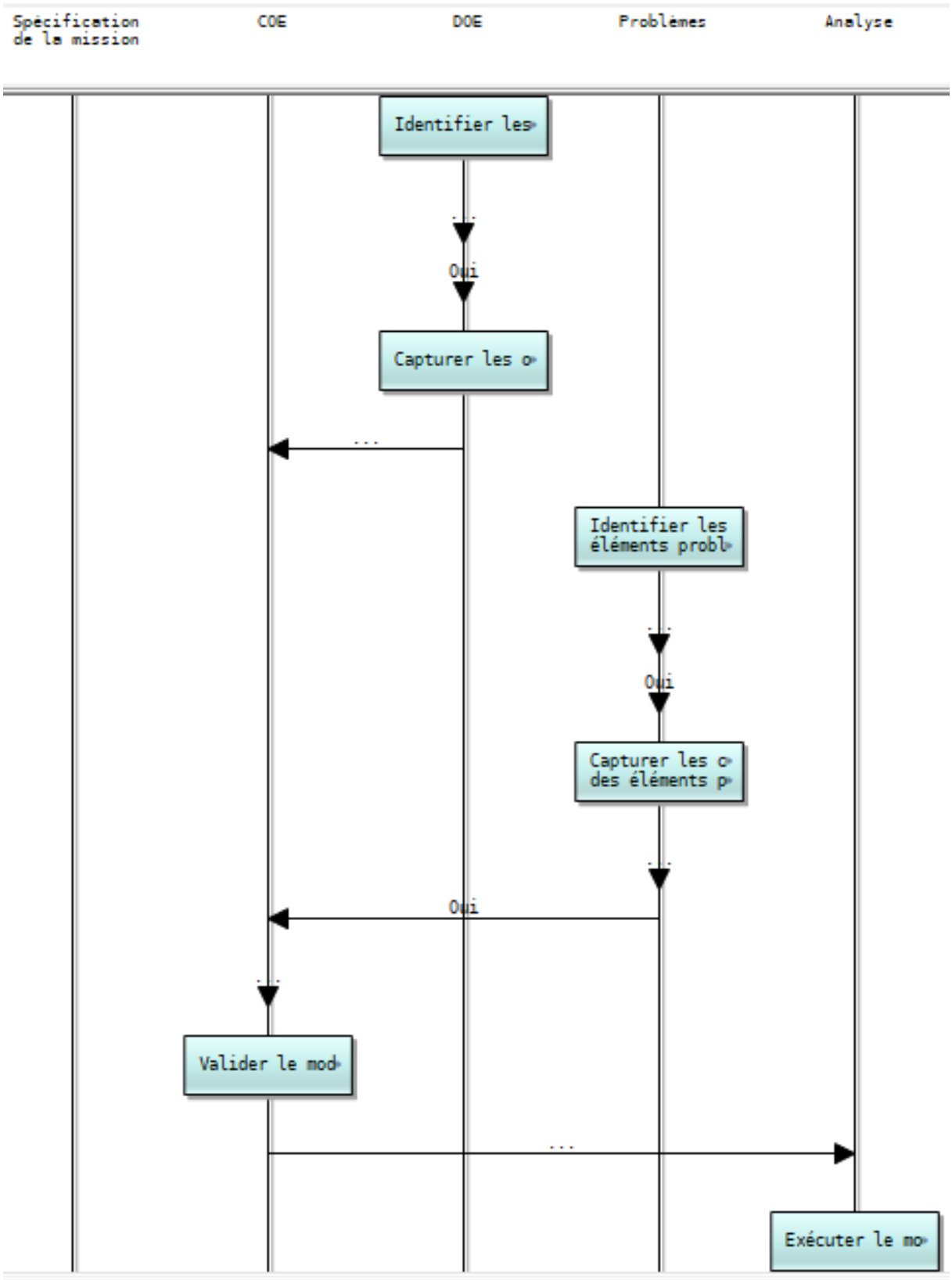


FIGURE B.4 – Diagramme de séquence d'une exécution du modèle BPMN global (4/5)

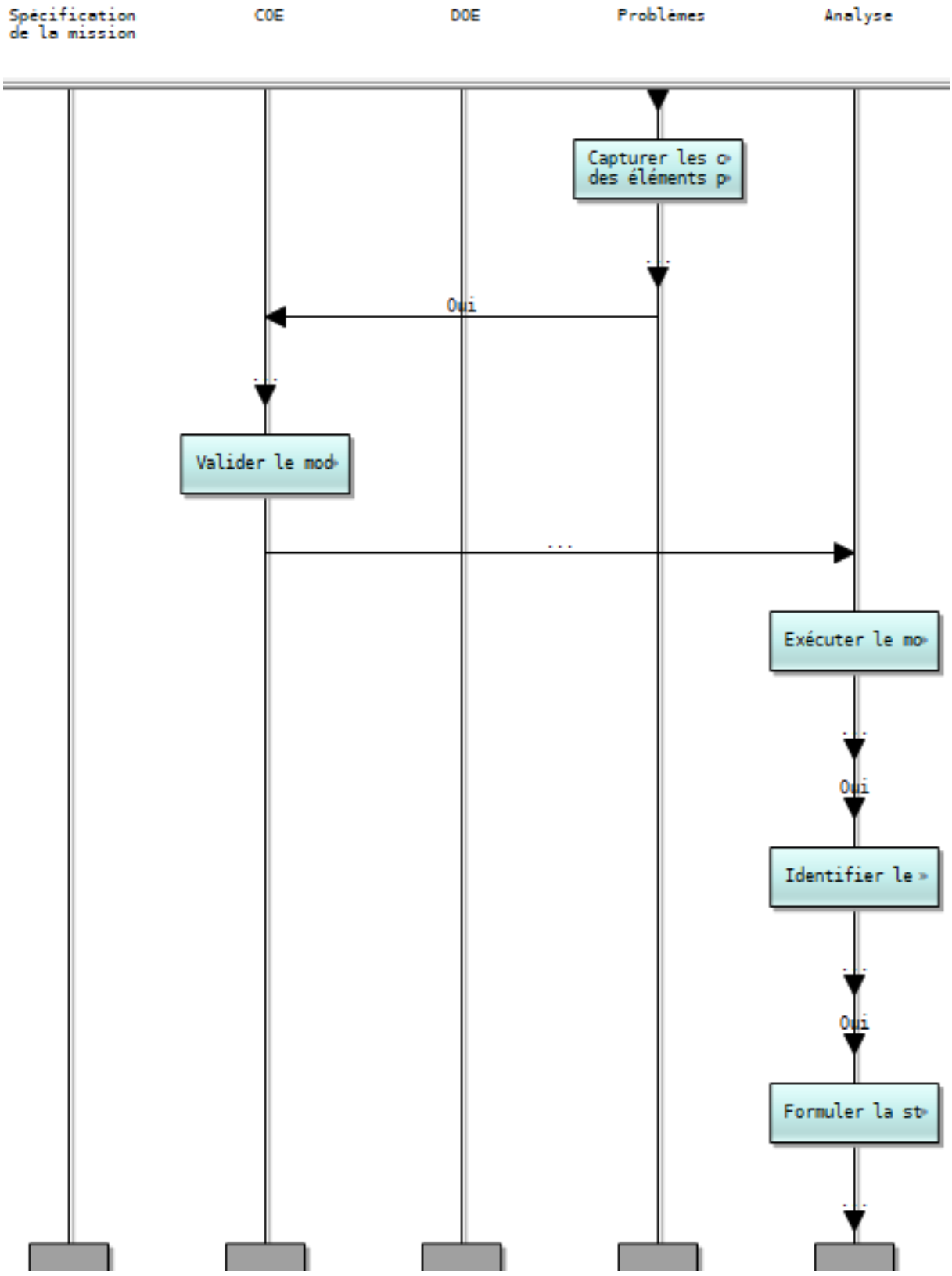


FIGURE B.5 – Diagramme de séquence d'une exécution du modèle BPMN global (5/5)

DOCTORAT BRETAGNE LOIRE / MATHSTIC

ENSTA BRETAGNE

Titre : Modélisation et Analyse Formelle de Modèles Système pour les Menaces Persistantes Avancées

Mot clés : Cyber-sécurité, Modélisation, Fédération de Modèles, Analyse Formelle, Model-Checking, Advanced Persistent Threat

Résumé : La criticité croissante des systèmes industriels les expose davantage aux menaces du monde cyber. En particulier les *menaces persistantes avancées* ou *Advanced Persistent Threats* (APT) sont des attaquants sophistiqués dotés de ressources conséquentes et ciblant spécifiquement les systèmes critiques. Les méthodologies de cyber-défense actuelles permettent de protéger les systèmes contre les cyber-menaces classiques mais elles peinent à contrer efficacement les APT. En effet, les APT usent de stratégies complexes et de tactiques de dissimulation qui les rendent difficile à contrecarrer. Pour répondre à ce besoin, la méthodologie d'*Operational Design* tirée des stratégies militaires permet de mieux comprendre l'établissement de stratégie de ces attaquants sophistiqués. Cette méthodologie axée sur la mission et adaptée au contexte des APT repose sur la fédération de plusieurs processus de spécification, de modélisation et d'analyse pour produire une stratégie opérationnelle. Pour évaluer cette approche, un outillage fédéré complet a été conçu et appliqué à un cas d'étude d'une mission d'attaque de système de pompage d'eau.

Title: Systems Modeling and Formal Analysis for Advanced Persistent Threats

Keywords: Cyber-security, Modeling, Model Federation, Formal Analysis, Model-Checking, Advanced Persistent Threat

Abstract: Critical industrial systems are prime targets of cyber threats. In particular the *Advanced Persistent Threats* (APT) are sophisticated and well-resourced attacks targeting valuable assets. For APTs both the attack and the defense require advanced planning and strategies similar to military operations. The existing cyber-security-aware methodologies achieve valuable results for regular cyber-threats, however they fail to adequately address APTs due to their refined strategies and evasive tactics. The *Operational Design* methodology of military forces helps in better understanding how APTs devise their strategies. This mission-driven methodology adapted to the APT context relies on the federation of several processes of specification, modeling and analysis in order to produce an operational strategy. To evaluate this approach, a complete federation framework has been developed and applied to the case study of a mission of attack on a water pumping station.