Maîtrise des correctifs de sécurité pour les systèmes navals

Bastien Sultan

DOCTORAT BRETAGNE LOIRE MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

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

Par

Bastien Sultan

Maîtrise des correctifs de sécurité pour les systèmes navals

Thèse présentée et soutenue à Brest, le 4 décembre 2020
Unité de recherche : Lab-STICC
Thèse N° : 2020IMTA0220

Rapporteurs avant soutenance :

Frédéric Cuppens — Professeur, Université Polytechnique de Montréal
Bouabid El Ouahidi — Professeur, Université Mohammed-V de Rabat

Composition du Jury :

Président : Laurent Nana, Professeur des Universités, Université de Bretagne Occidentale
Examinateurs : David Brosset, Maître de Conférences, Arts et Métiers Sciences et Technologies, École Navale
Frédéric Cuppens, Professeur, Université Polytechnique de Montréal
Bouabid El Ouahidi, Professeur, Université Mohammed-V de Rabat
Joaquín García-Alfaro, Professeur, Télécom SudParis
Dir. de thèse : Yvon Kermarrec, Professeur, IMT Atlantique Bretagne Pays de La Loire

Invité(s)
Patrick Hébrard, Responsable Recherche et Innovation Cyber, Naval Group
Marc Pennamen, Responsable Offres et Projets, Thales SIX GTS France

N° d'ordre : 2020IMTA0220

Sous le sceau de l'Université Bretagne Loire

IMT Atlantique
Bretagne-Pays de la Loire

En accréditation conjointe avec l'Ecole Doctorale MathSTIC

Maîtrise des correctifs de sécurité pour les systèmes navals

Thèse de Doctorat

Spécialité: Informatique

Présentée par **Bastien Sultan**

Département: Logique des Usages, Sciences Sociales et de l'Information

Laboratoire: Lab-STICC – UMR CNRS 3285

Thèse réalisée au sein de la Chaire de cyberdéfense des systèmes navals

Directeur de thèse : Yvon Kermarrec

A soutenir le 4 décembre 2020

Jury :

Frédéric Cuppens, Professeur, École Polytechnique de Montréal (Rapporteur)
Bouabid El Ouahidi, Professeur, Université Mohammed-V de Rabat (Rapporteur)
David Brosset, Maître de Conférences, Arts et Métiers Sciences et Technologies, École Navale (Examinateur)
Joaquín García-Alfaro, Professeur, Télécom SudParis (Examinateur)
Laurent Nana, Professeur, Université de Bretagne Occidentale (Président)
Yvon Kermarrec, Professeur, IMT Atlantique (Directeur de thèse)
Patrick Hébrard, Responsable recherche et innovation cyber, Naval Group (Invité)
Marc Pennamen, Responsable offres et projets, Thales SIX GTS France (Invité)

Il est juste de rapporter les raisons qui m'ont déterminé à suivre une route différente.

Henri-Louis Duhamel du Monceau
Fondateur de l'École des ingénieurs-constructeurs des Vaisseaux royaux – aujourd'hui appelée École Nationale Supérieure de Techniques Avancées
Éléments de l'architecture navale

Table des matières

Acronymes

ANSSI Agence Nationale de la Sécurité des Systèmes d'Information.

ATM Arrêt Technique Majeur.

CERT Computer Emergency Response Team.

Ch Cheval Vapeur.

CODOG COmbined Diesel Or Gas.

COTS Commercial Off-The-Shelf.

CPE Common Platform Enumeration.

CPS Cyber Physical System.

CSV Comma-Separated Values.

CVSS Common Vulnerability Scoring System.

DGA Direction Générale de l'Armement.

ENSTA École Nationale Supérieure de Techniques Avancées.

EVP Équivalent Vingt Pieds.

GNSS Global Navigation Satellite System.

IHM Interface Homme-Machine.

IMT Institut Mines-Télécom.

MCAS Maneuvering Characteristics Augmentation System.

MCO Maintien en Condition Opérationnelle.

MCS Maintien en Condition de Sécurité.

NIST National Institute of Standards and Technology.

NVD National Vulnerability Database.

PID Proportionnel, Intégral, Dérivé.

PLC Programmable Logic Controller.

PSSI Politique de Sécurité des Systèmes d'Information.

rpm Tours par minute.

SCADA Supervisory Control And Data Acquisition.

SNLE Sous-Marin Nucléaire Lanceur d'Engins.

SNLE-NG Sous-Marin Nucléaire Lanceur d'Engins de Nouvelle Génération – Classe Le Triomphant.

SSF Service de Soutien de la Flotte.

VAPT Vulnerability Assessment and Penetration Testing.

XML Extensible Markup Language.

Résumé/*Abstract*

Évoluant dans des environnements contraints et rassemblant dans des espaces réduits des sous-systèmes fortement critiques et hétérogènes, les navires d'aujourd'hui font partie des objets les plus complexes qui soient. À l'heure où croît à leur bord le nombre de systèmes informatiques contrôlant parfois des actionneurs d'importance cruciale, leur maintien en condition de sécurité est une problématique majeure. Les travaux présentés dans cette thèse définissent un processus de gestion des vulnérabilités et des contremesures associées adapté au contexte des systèmes industriels complexes. Ce processus s'appuie sur une méthode et un formalisme de modélisation de ces systèmes permettant d'abstraire leur comportement, discret ou continu, et leurs évolutions éventuelles – apparition d'une vulnérabilité, déploiement d'une contremesure ou survenue d'une attaque – que nous avons définis dans le cadre de ces travaux. Il repose également sur une méthode de calcul des impacts associés aux vulnérabilités, attaques et contremesures aboutissant à leur expression sous la forme d'une métrique adaptée pour la prise de décision. Ce calcul étant permis par la modélisation du système mais aussi par celle des vulnérabilités, attaques et contremesures, nous introduisons par ailleurs une méthode et un formalisme adapté – les mutations d'automates et de réseaux d'automates – permettant leur abstraction. Ces propositions théoriques et méthodologiques sont enfin confrontées à une expérimentation sur un cas fictif représentatif d'un système de propulsion et de gouverne d'un bâtiment de type ferry ou paquebot, permettant de discuter de leur pertinence et de leurs limites ainsi que d'esquisser les perspectives de recherche naissant de nos travaux.

*
* *

Operating in constrained environments and composed of heterogeneous sub-

systems, today's ships are among the most complex objects that exist. Due to the increasing number of cyber assets among their components, patch and vulnerability management applied to naval systems is an essential process. The work detailed in this PhD thesis aims to define such a process tailored to complex cyber-physical systems. This process relies on a modelling method and formalism allowing to depict CPS behaviour and cyber events – a vulnerability discovery, a cyber attack occurrence or a patch deployment. It also relies on an impact assessment method, allowing to compute the effects of cyber events on CPS ability to fulfill their missions. These impacts are expressed through a specially designed metric aiming to help in decision-making. The process, methods, formalisms and metrics we propose in this work are then evaluated through an experimentation based on a fictitious case-study.

Remerciements

Il est de tradition invariable qu'un travail académique respectable commence par une revue de littérature sérieuse du sujet qu'il a l'ambition de traiter. Et comme j'ai l'ambition de la respectabilité, il m'eût semblé absurde de ne point mener une recherche bibliographique approfondie avant d'apporter ma contribution à la science des remerciements.

Ainsi, il ressort de l'étude minutieuse et exhaustive de toutes les thèses écrites depuis les alentours du Moyen-Âge (à dix ans près) que ces pages rituelles suivent en règle générale le plan suivant :

1. témoignage de reconnaissance distinguée aux non-moins distingués membres du jury de soutenance ;
2. témoignage de gratitude sincère aux non-moins sincères camarades de labeur stakhanoviste qui ont partagé les joies profondes de l'auteur durant ses années de recherche (scientifique) ;
3. témoignage d'affection envers les amis dudit auteur ;
4. témoignage d'affection émouvant adressé à sa famille.

Il est à noter que certains s'affranchissent malgré tout des canons de cet exercice et remercient aussi, par exemple, des bateaux [Bro08]. Ces incursions dans le domaine des fulgurances restent malgré tout assez minoritaires et peu d'auteurs investissent le champ du génie à l'exception, peut-être, de quelques théoriciens de la construction ; mais dans ce cas, il s'agit davantage d'une déformation professionnelle. Par conséquent, nous pouvons conclure que les remerciements constituent un exercice normé. Cette norme provient essentiellement – au sens de l'essence, en tant que *norme* – de l'usage social [Bec63] ; et malgré son effet de standardisation quant aux contenus, la qualité des productions qui s'y conforment demeure variable. Nous pouvons trivialement établir une typologie de ces écrits en fonction

des émotions littéraires procurées au lecteur (librement adapté de [Bei90], *cf.* figure 1) : si la quasi-totalité des auteurs aspire à provoquer chez son lectorat une

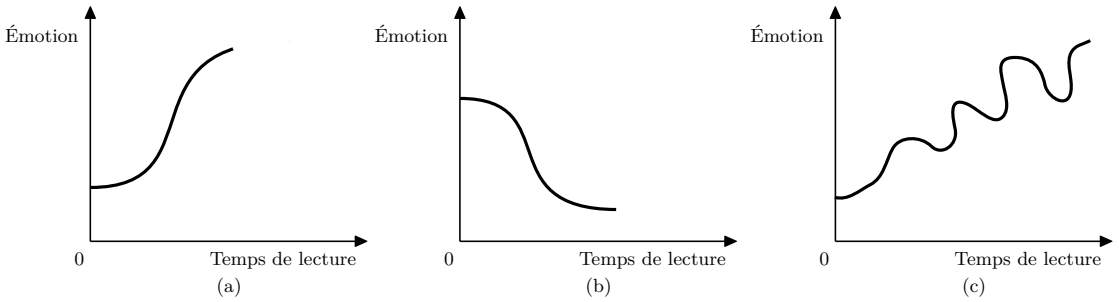


FIGURE 1 – Essai de typologie inspiré de [Bei90].

courbe d'émotion conforme à (a), l'écrasante majorité d'entre eux suscite en réalité un intérêt décroissant (b). L'adhérence à la courbe (a) semblant dès lors utopique, la science nous invite à provoquer des émotions contrastées chez le lecteur afin de maintenir sa curiosité et, *in fine*, un plaisir littéraire croissant (c) : « comme quoi la rigueur mathématique ne messied pas à l'analyse des sentiments. » [Bei90] Et l'académisme ne messied pas aux remerciements.

Ainsi, je souhaite en premier lieu adresser mes remerciements chaleureux et amicaux à Yvon Kermarrec, pour son soutien indéfectible, son écoute remarquable et son aide précieuse. Ces travaux doivent beaucoup à ses conseils et encouragements. Pour avoir accepté des les rapporter, et formulé des avis fort intéressants qui m'ont ouvert des perspectives de réflexion attrayantes, je remercie vivement Frédéric Cuppens et Bouabid El Ouahidi. J'ai été très honoré du temps qu'ils ont accordé à leur évaluation — comme je le suis de pouvoir compter parmi les membres de mon jury de soutenance David Brosset, Joaquín García-Alfaro, Patrick Hébrard, Laurent Nana et Marc Pennamen : l'intérêt que vous avez porté à mes travaux me ravit.

Je ne saurais trop remercier Benoît Clément, Luc Jaulin et Benoît Zerr pour m'avoir transmis leur passion et donné le goût de la « réflexion scientifique » ! Croiser votre route fut une grande chance. Croiser celle de ceux qui, à l'École Navale, à IMT Atlantique ou au détour des colloques, workshops, journées scientifiques et expérimentations m'ont si bien accueilli, fut une expérience non moins enrichissante. Tous vous citer risquerait hélas de métamorphoser ces pages en annuaire... Merci à Caroline, Patrick, Philippe, Fabien, Laura, Joanne, Gaëlle, Sophie, Frank, Mathieu, Gilles ; merci à l'amiral Pagès de son accueil chaleureux et de nos conver-

sations enlevées, à Christophe pour sa disponibilité et sa bienveillance, et à Patrice pour ces moments de rire et ces discussions savoureuses.

J'ai naturellement une pensée toute particulière pour mes chaires (!) camarades ; vous rencontrer fut un vrai plaisir, et c'est sûr d'avoir noué de belles amitiés que j'ai vogué vers d'autres cieux. Alors, j'adresse un grand merci et mes pensées amicales à Benjamin, Étienne, Paul, Pedro, Olivier, Guillaume, Xavier, Douraïd, Clet, Nicolas, Arthur, Thibaud, Mallorie, Mickaël, Maël, ... et à celles et ceux que je n'ai eu la chance de rencontrer qu'épistolairement !

Avant de poursuivre plus avant, je me dois d'exprimer toute ma gratitude à F-GTAT, F-HEPJ et aux salons Air France. Ne serait-ce que pour toutes les lignes que j'y ai écrites ! En revanche, je ne remercie pas le métro. Question de principe ! je ne transigerai pas.

Cher David, je sais ce que cette thèse te doit. Sans ton courageux soutien dans l'affaire des mèches, Dieu seul sait dans quels affres dépressifs j'aurais été plongé ! Je ne te raconte pas le niveau. Merci pour tout — et plus encore, d'ailleurs.

Cher Marc, pour ces déjeuners, ces conversations et ces Nespresso réjouissants, je t'adresse un grand merci et toutes mes amitiés.

Mon cher Vincent, *merci* (avec une emphase sur le « e »). Que paraît longue la route depuis nos étés à Kerskourr, nos tours du lycée qui nous mettaient invariablement en retard au cours de math, ou nos doctes performances d'œnologues distingués ! À force d'errances dans les méandres de l'enseignement supérieur ? Mais dans notre folie du cumul, même après cette thèse, tu gardes une sérieuse avance.

Cher Matthieu, comme tu le sais je m'émerveille désormais bien souvent à la constatation qu'« un étroit fossé sépare dûment l'Ain de Troyes. » Merci pour ta production versifiée où le sérendipe le dispute à la grâce, et pour tout le reste.

Mon cher Pierre-Hugues, pour ces bons moments et dîners partagés entre la Bretagne, Paris, Boulogne et presque Nice, merci.

Cher Benjamin, les fonctions fuchsiennes ont déjà été théorisées ; mais je sais que ce sont des fulgurances plus vives encore que forment ton esprit quand tu montes dans un autobus. Alors, au nom du savoir, des petits-déjeuner de onze heures préparant les déjeuners de onze heures et demi, et de tant d'autres choses : merci. Car c'est une spirale !

Mon cher Paul, je lève mon verre de Moscow Mule à ta santé, avec le plus de

panache dont je suis capable. Merci pour ces discussions aussi passionnantes que peut être solide un œuf d'autruche, et ce n'est pas peu dire.

Cher Étienne, pour ces restaurants dantesques, ces conversations musico-échiquéo-culinaires, pour *Le Daim*, pour ces promenades toujours studieuses et réflexives, merci.

Mon cher Tanguy, je t'adresse des remerciements de confiance. Et je te les adresse... *ici*. Merci pour ces joyeux moments entre l'Irlande et l'Espagne... et tout cela dans une seule rue, de surcroît !

Cher Gaël, toi qui es poursuivi par Paul Baysse, il me tarde que tu reviennes séjourner au bout de la terre, pour tous ces joyeux moments à venir entre l'Irlande et l'Espagne... et tout cela dans plusieurs rues, désormais !

Cher Ronan, je t'exprime une gratitude fort qualitative. Elle eût certes pu être quantitative, mais cela aurait manqué de ce je-ne-sais quoi d'élégance lunaire dont nous raffolons tant.

Cher Adrien, j'attends impatiemment notre prochain déjeuner à la Boule Rouge ! Merci pour ces années de rire. Et cette sortie par la fenêtre, que je ne suis pas près d'oublier !

Mon cher Antoine, merci pour ces beaux moments à Jullouville, Toulon, Brest... Chausey, Le Conquet... diantre, que d'eau ! Merci pour nos conversations, dégustations, représentations, etc. passées et à venir. *Cheers, old chap* !

Cher François, mon cher docteur (Pepper ?), merci pour ces moments et discussions jubilatoires qui, tout en conservant une profondeur remarquable, sont souvent aussi déjantés qu'un concours de lancer de parpaing.

Mon cher Mickaël, *c'est énorme*. Merci pour nos conversations sur la philosophie allemande, dans des pubs irlandais ou des restaurants espagnols — dire que nous incarnons pourtant l'échec de l'enseignement des langues dans ce pays ! Cela vaut bien tous les menuets du monde.

Maintenant, conformément à ce que j'avais annoncé quelques pages auparavant, le moment est venu de remercier ceux que j'ai la chance de pouvoir compter parmi ma famille. Papa, Maman, je n'ai guère besoin de vous écrire ce que je vous dois. Irène-Marie, Lou-Andréa, Jean — enfin, surtout Jean —, quelle belle fortune ai-je de vous connaître ! Me vient également une pensée particulière pour mes grands-parents et pour Pierre-François. Vous tous m'avez donné, entre autres, ce goût des choses de l'esprit qui m'a poussé à mener cette aventure.

Ma chère Loubna, grâce à toi nombre d'aventures me semblent désormais possibles. Sans toi, jamais je n'aurais pu tracer ces lignes, enhardies de l'éclat des hivers auboises et du souffle de Mogador ; des deux Z qui me regardent les écrire et des tilleuls verts de la promenade ! Alors, comme d'autres avant moi, je crois que je vais devoir remercier ce bateau sans qui rien ne serait jamais arrivé.

*
* *

Ces dernières lignes vont à celui que j'associerai toujours, peut-être plus que tout autre, au savoir dans ce qu'il a de plus noble : la liberté, l'humilité, et la transmission. Car ce que tu m'as transmis est le plus précieux et le plus enviable des passeports pour l'existence. Tu expliquais avec la même passion l'histoire que la géométrie anallagmatique, narrais avec la même flamme Vermeer et Villon que Descartes et Teilhard. Et travailler mon arithmétique et ma géométrie sur tes livres aux pages élégamment noircies me rappelait dans un sourire que s'il y a un champ que l'art et l'imagination ne doivent jamais déserter, c'est bien celui des sciences.

*
* *

Bibliographie

[Bei90] Frédéric BEIGBEDER. *Mémoires d'un jeune homme dérangé : roman.* La Table Ronde, 1990. ISBN : 9782710381518

[Bro08] David BROSSET. *Description d'itinéraire en milieu naturel : modèle intégré de description verbale et de représentation spatiale au sein des systèmes d'information géographiques.* Thèse. Arts et Métiers ParisTech, 2008

[Bec63] Howard S. BECKER. *Outsiders.* Free Press, 1963. ISBN : 9781439136362

Introduction générale

Le navire du XXI^{ème} siècle : un système complexe

Les navires – ou *systèmes navals* – d'aujourd'hui font partie des systèmes industriels les plus complexes qui soient [Ste+13]. Systèmes de systèmes évoluant dans un environnement contraint, induisant des risques et des niveaux de fatigue élevés pour les hommes et le matériel (voies d'eau, incendies, isolement à la mer, mouvements de plateforme, ...), ils rassemblent dans des espaces réduits des systèmes variés et critiques. À titre d'exemple, un sous-marin nucléaire lanceur d'engins [Dup19] embarquera entre autres une chaufferie nucléaire intégrée, des systèmes de production d'eau douce et d'oxygène par électrolyse de l'eau de mer, des centrales inertielles, des systèmes de régulation d'assiette et de plongée (ballasts, barres), des plateformes pour assurer le lancement en immersion de missiles balistiques intercontinentaux, un ensemble de capteurs acoustiques, un système d'armes et un système de combat, ... et tout cela confiné dans une plateforme de moins de deux cents mètres de long et évoluant en mission en totale autonomie à plusieurs centaines de mètres de profondeur, avec des exigences de discrétion absolue ! Au-delà des SNLE qui constituent un sommet en la matière, l'ensemble des navires, militaires comme civils, affichent désormais une complexité très forte, à la mesure de leur gigantisme et de leurs capacités qui ne cessent de croître. Le monde de la marine marchande nous fournit à chaque nouveau baptême de porte-conteneurs des exemples éloquents, à l'image du *CMA-CMG Antoine de Saint-Exupéry* qui affiche des chiffres étourdissants : ce bâtiment, long de 400 mètres et large de 59, propulsé par un moteur de 100 000 ch, a une capacité d'emport de 20 600 EVP (équivalent vingt pieds, soit un conteneur d'une quarantaine de mètres cube) [CC18], ce qui représente une caragaison dont la valeur peut avoisiner les 2 milliards d'euros [Pre15].

Il convient également d'ajouter que ces bâtiments sont aujourd'hui opérés par des équipages très réduits : 26 hommes arment l'*Antoine de Saint-Exupéry* [Gar18], ordre de grandeur commun à bord de ces navires géants ; 108 marins opèrent la classe *Aquitaine* de la Marine nationale [Mar18] (contre 244 pour les navires de la génération précédente, la classe *Georges Leygues* [DIC14]. Cette fonte des effectifs du bord est naturellement à mettre en regard d'une numérisation très forte des systèmes, que l'on peut désormais qualifier de « navires numériques ».

À titre d'illustration, le navire armé type du XXI^{ème} siècle voit cohabiter en son sein (sources Naval Group) :

— 2000 applications et logiciels ;
— 400 entrées/sorties et automates ;
— 4 niveaux de confidentialité séparés ;
— 300 calculateurs, dont 30 temps réel ;
— 350 kilomètres de cables ;
— 150 équipements réseau ;
— un système de combat dont la part logicielle est codée sur 20 millions de lignes ;
— un système de gestion de plateforme dont la part logicielle est codée sur 2 millions de lignes.

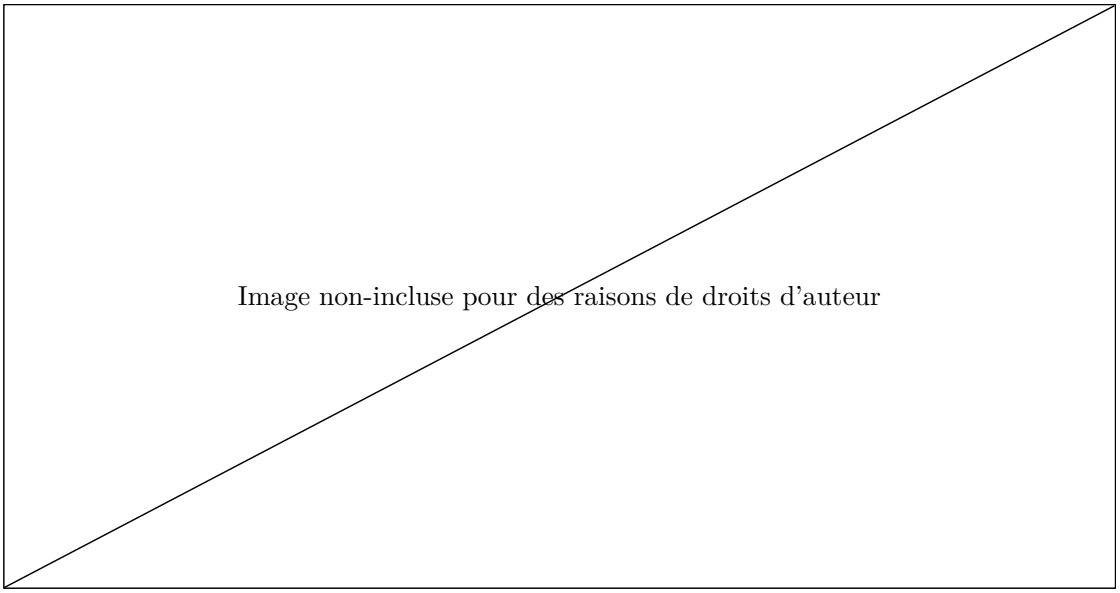


FIGURE 2 – Le navire armé numérique type. Source : Naval Group

Gigantisme, complexité, technologies de pointe, numérisation : tel est désor-

mais le visage des systèmes navals, fruit de plusieurs siècles d'innovations et qui ont façonné l'industrie navale d'aujourd'hui et permis des gains importants en matière de capacités tout en maîtrisant, à chaque fois, les risques induits par l'introduction de ruptures technologiques. Tout au long de son histoire, la construction navale a vu ses acteurs rechercher l'équilibre optimal entre capacités et maîtrise des risques, nécessairement forts du fait de l'environnement d'évolution des navires et souvent accrus par l'émergence de technologies nouvelles, afin d'optimiser au mieux les capacités des systèmes, leur sûreté, leur sécurité et celle des équipages.

Les innovations de rupture de la construction navale : des gains capacitaires à la maîtrise des risques

Si l'importance des enjeux économiques, scientifiques et territoriaux liés à la maîtrise de la mer ont, depuis l'Antiquité, poussé l'homme à se tourner vers les océans et la construction de vaisseaux permettant de les parcourir, l'ingénierie navale a connu une évolution relativement lente jusqu'à la fin de l'époque moderne. L'état des connaissances scientifiques en matière d'architecture navale et d'hydrodynamique est en effet resté assez figé jusqu'à l'orée du dix-huitième siècle, où les travaux de Bernoulli [Ber38], d'Alembert [LRd70] et Euler [Bou60] ouvrirent la voie à une étude rigoureuse de la mécanique des fluides et à ses applications techniques. Quelques décennies plus tôt, sous l'impulsion de Colbert, secrétaire d'État à la Marine entre 1669 et 1683, la France s'était engagée dans l'acquisition de moyens navals importants dans le but de combler l'écart qui s'était creusé avec les principales marines militaires ou commerciales européennes, au premier rang desquelles les flottes anglaise et hollandaise. Combinée au progrès des sciences marines, cette volonté politique initiera un âge d'or pour l'ingénierie navale et donnera naissance à de nombreuses innovations techniques.

C'est en effet au début du dix-huitième siècle, avec la fondation en 1741 par Duhamel du Monceau de l'école des ingénieurs-constructeurs des vaisseaux royaux, que la construction navale de l'âge de la voile sera consacrée en tant que science. Elle atteindra son paroxysme avec Jacques-Noël Sané (1740-1831), ancien élève de Duhamel du Monceau, inventeur de la standardisation des vaisseaux [Wie+18] – la fameuse notion de *classe* de bâtiments – en 1782 avec la conception du *Téméraire* [WR15], vaisseau de 74 canons à la conception innovante tant en termes de navigabilité que d'armement. C'est à cette période que l'industrialisation de la

construction navale deviendra effective, reléguant au rang du passé des méthodes souvent empiriques qui ne répondaient plus à l'exigence d'efficacité et de puissance requises par la compétition croissante entre les États afin d'obtenir la maîtrise des océans. Les innovations de Sané, notamment en termes de manœuvrabilité, se traduisirent par un gain pur en matière de capacités des systèmes. Il convient de noter qu'il s'agissait essentiellement d'innovations de continuité (amélioration des formes des coques et des gréements, de la répartition des armements, etc.) et que les ruptures introduites par Sané concernaient avant tout les méthodes de construction. En revanche, la construction navale intégrera par la suite des innovations de rupture qui, bien que bénéfiques sur le plan capacitaire, se feront rarement sans l'introduction de nouveaux risques – comme nous le verrons à travers trois exemples significatifs.

Au cours du dix-neuvième siècle, la machine à vapeur issue de la première révolution industrielle verra sa généralisation à bord des navires, militaires ou commerciaux [GL92]. Cette intégration permettra des gains phénoménaux en termes de puissance et de vitesse, réduisant les temps de traversée et permettant d'augmenter significativement les volumes des bâtiments, mais ses contreparties sont majeures : risques d'incendie, autonomie réduite du fait du mauvais rendement des premières machines imposant d'embarquer d'importantes quantités de charbon – au détriment de l'espace alloué aux passagers ou au fret dans le cas des marines commerciales, ou à l'armement pour les marines militaires –, risque de perte totale de manœuvrabilité en cas de panne des moteurs, fragilité des roues à aubes utilisées pour la propulsion, etc. Les premiers bâtiments mus par la vapeur furent d'ailleurs des navires à propulsion mixte – à vapeur et vélique –, solution de contournement imaginée par les ingénieurs navals en attendant le développement de technologies permettant une maîtrise satisfaisante des risques induits. Malgré la maîtrise ultérieure de ces risques, le passage à la vapeur induira une perte de discrétion des bâtiments : par l'augmentation de leurs dimensions bien sûr, mais avant tout par la génération des panaches de fumées d'échappement aisément repérables et l'augmentation du bruit rayonné. La machine à vapeur sera petit à petit abandonnée et d'autres technologies de propulsion verront le jour : les premiers moteurs thermiques seront expérimentés dans les années 1880 et commenceront à propulser les navires à l'aube du vingtième siècle [Smi10], les turbines à gaz, offrant des capacités d'accélération inédites, seront intégrées aux bâtiments militaires peu après la seconde guerre mondiale [WF04] ; mais les risques induits par l'intégration de ces modes de propulsion recoupaient dans une large mesure ceux que les ingénieurs avaient déjà résolu suite à la généralisation de la vapeur.

En matière de propulsion navale, la grande rupture technologique du vingtième siècle viendra de l'utilisation de l'énergie nucléaire. Intégrée dès 1955 sur l'*USS Nautilus* [ZM18], elle offre des avantages considérables : une autonomie très fortement accrue – nettement supérieure aux besoins des missions puisqu'elle se compte en années –, un bruit rayonné nettement inférieur à celui des moteurs thermiques, un remarquable gain de puissance, etc. La réaction nucléaire est par ailleurs une source d'énergie anaérobie, ce qui constitue un avantage précieux dans le cas des sous-marins qui n'ont de ce fait aucun besoin de diminuer périodiquement leur immersion afin d'alimenter leurs moteurs en air via leurs schnorchels. Naturellement, l'utilisation d'une telle technologie à des fins de propulsion navale a nécessité la résolution de défis inédits [Fri], outre ceux de la résolution des risques classiques en matière d'exploitation nucléaire : aménagement de chaufferies dans des espaces réduits, conception de solutions de rechargement du combustible à l'occasion des arrêts techniques des bâtiments, spécification des réacteurs afin qu'ils puissent fonctionner dans un environnement naval avec les mouvements de plate-forme que cela implique (accélérations, roulis, tangage, etc.), exigence de disponibilité constante de la propulsion, exigences de sûreté spécifiques, ... Nous pouvons également mentionner l'intégration d'exigences drastiques en matière de bruit rayonné – les sytèmes conçus dans le cadre du programme SNLE-NG (classe *Le Triomphant*) constituent en la matière un exemple remarquable du fait des niveaux visés en terme de discrétion acoustique [Fri]. La propulsion nucléaire navale est aujourd'hui utilisée sur 150 bâtiments environ, très majoritairement militaires et sous-marins.

La dernière rupture technologique de l'industrie navale que nous citerons est celle induite par l'utilisation de l'électronique à des fins d'automatisation des processus industriels. La théorie de la commande des systèmes dynamiques, dont l'asservissement [1] qui en est l'application la plus répandue, une science ancienne : dès 1788, James Watt mit au point un régulateur entièrement mécanique destiné à réguler la vitesse de rotation des machines à vapeur [LL04] ; le problème

1. En automatique, l'*asservissement* d'un système est un procédé de commande en boucle fermée consistant à faire adhérer le plus possible la sortie du système à la consigne fixée par un opérateur sans aucune intervention humaine. Outre la consigne, ce procédé prend en compte afin d'élaborer sa commande l'*observation* (mesure) de la sortie et la mesure des perturbations extérieures éventuelles. Considérons par exemple le safran d'un navire. Nous notons $\theta(t)$ l'angle entre le safran et l'axe de la coque : $\theta(t)$ est la sortie du système. Le système sera « asservi en angle » à la consigne $\theta_c(t)$ s'il existe un procédé automatique permettant de faire correspondre au mieux $\theta(t)$ à $\theta_c(t)$ en s'appuyant sur la mesure de $\theta(t)$ et des perturbations environnementales – dans le cas d'espèce, le courant marin.

de l'évolution et de la commande des systèmes dynamiques fut ensuite formalisé mathématiquement en 1840 par Airy [Air40] puis en 1868 par Maxwell dans son article *on governors* [Max68]. Les premiers mécanismes de commande asservie qui en découlèrent furent des régulateurs pneumatiques et hydrauliques. Au gré des avancées théoriques et techniques, ces systèmes se perfectionnèrent : à la fin des années 1960, le premier automate programmable industriel (*programmable logic controller*, ou PLC) fut conçu par General Motors [Bis09], synthèse des mécanismes de commande industrielle existant jusqu'alors et des possibilités ouvertes par l'essor de l'informatique et de l'électronique – un tel automate étant un ordinateur dédié à la commande d'un système, disposant de ports d'entrée (permettant la remontée d'informations des capteurs, la communication des consignes par les opérateurs, ...) et de ports de sortie (auxquels peuvent être branchés des actionneurs, des systèmes d'alarme, ...). Ces PLC, une fois convenablement paramétrés, permettent donc de réaliser intégralement les asservissements et autres bouclages pour lesquels ils ont été programmés. Leurs avantages sont nombreux : ce sont des composants génériques, reprogrammables et aisément remplaçables, faciles d'utilisation, à l'encombrement très réduit, et livrés avec des logiciels de supervision dédiés (les fameux SCADA). De ce fait, les PLC sont de nos jours largement utilisés dans les systèmes industriels complexes, au nombre desquels les navires. En effet, leurs applications sont nombreuses à bord des bâtiments : commande des systèmes de propulsion et de manœuvre (moteurs Diesel, turbines à gaz, safrans, ...), des systèmes de tranquilisation de plate-forme – stabilisation en roulis, tangage, lacet, embardée, ... – et de compensation de gîte, systèmes de gestion de l'énergie, systèmes de sécurité (« *alarmes incendie ou de voie d'eau* »), etc. [LD94 ; Gro17 ; LV17 ; Pau+17]. Le bénéfice de l'intégration de tels dispositifs est évident – outre les avantages propres aux PLC que nous avons évoqués, une automatisation rigoureuse des processus industriels au sein des systèmes critiques induit avant tout des gains de fiabilité et de rapidité d'exécution des processus, de supervision des systèmes, une diminution de la pénibilité des métiers du bord, un bénéfice économique, etc. Mais une fois encore, les risques associés sont non nuls : l'un des plus notables est celui lié à la remontée d'information erronnées, étant donné que les commandes sont élaborées d'après des informations fournies par une série de capteurs qui peuvent apporter aux automates des informations corrompues [Cos18]. Nous pouvons à ce titre citer l'exemple du vol 501 d'Ariane 5, où les sytèmes chargés d'établir les consignes de pilotage du lanceur ont ordonné une série d'actions de correction de trajectoire ayant fait violemment dévier la fusée de sa trajectoire, et détériorant sa structure (désolidarisation des boosters, ...) jusqu'à ce que la fusée s'autodétruise, suite à

des remontées d'informations erronées liées à deux défaillances identiques sur le système de guidage inertiel principal et le système de secours qui avait pris le relais [Lio96]. Plus récemment, les exemples tragiques des vols JT610 de Lion Air et ET302 d'Ethiopian Airlines, impliquant tous deux des Boeing 737-MAX 8, illustrent tous deux ce risque : la probable cause des deux accidents est une défaillance du MCAS, système mis au point par Boeing et destiné à prévenir un décrochage de l'aéronef en rétablissant, par des actions sur l'empennage, un angle d'attaque « sûr » lorsque l'appareil s'approche de son incidence de décrochage ; dans ces deux cas, une sonde d'incidence, l'un des capteurs fournissant des informations au MCAS, a fait remonter des données aberrantes qui ont poussé le système à effectuer une succession d'actions à piquer dont nous connaissons l'issue [Mui19].

Chacune de ces défaillances, occasionnant ou non un accident, amène les concepteurs des systèmes à améliorer la maîtrise des risques liés à leur automatisation grandissante afin d'atteindre des niveaux de sûreté très élevés : une fois encore, le gain capacitaire induit par la rupture technologique a appelé une réflexion autour des risques qu'elle génère. Or si nous avons évoqué l'écueil de la remontée d'informations erronées, un autre danger lié à l'utilisation des PLC à des fins d'automatisation des processus existe et fonde le travail de recherche présenté dans ce mémoire. Spécifiés en premier lieu pour fonctionner de manière sûre, ces ordinateurs n'intégraient pas, dans leurs premières versions, de mécanismes de sécurité informatique [ML17]. Il convient d'ajouter à cela des faiblesses récurrentes dans les architectures les intégrant : comme le souligne la note stratégique *Cybersécurité dans le milieu maritime* publiée par la société CEIS en 2017 [Pau+17], les systèmes de contrôle industriel s'appuyant sur ces PLC ont souvent été mis en réseau pour permettre leur opération depuis la passerelle du navire afin que les équipages, de plus en plus réduits, aient une capacité de supervision et de contrôle de l'ensemble des systèmes industriels du navire depuis un point d'entrée unique. Par ailleurs, il arrive également que « *pour répondre aux besoins de contrôle et de maintenance distante par les armateurs et organismes de soutien, ces systèmes [de contrôle industriel soient] connectés aux réseaux de communication qui relient le navire à la terre* » [Pau+17] – réseaux qui, comme le souligne la même note, servent parfois à des usages extrêmement divers dont les besoins de sécurité sous-jacents le sont tout autant (« communications professionnelles du navire », accès Internet de l'équipage ou des passagers, remontée d'informations des conteneurs connectés, etc.). Cette fusion des réseaux et des usages ouvre naturellement de nouveaux chemins d'attaque, élargissant notablement la surface d'exposition des navires à des attaques opérées à distance [Pau+17]. Des vulnérabilités ainsi ouvertes pourraient

permettre des actions malveillantes ciblant les informations des capteurs, les commandes du bâtiment elles-mêmes, ses informations opérationnelles, etc. et dont les conséquences sont aisément imaginables : *in fine*, nous voyons donc se dessiner pour les systèmes industriels les contours d'un nouveau risque qu'il convient de prendre en compte et de maîtriser.

La Chaire de Cyberdéfense des Systèmes Navals

« Répondant à l'objectif d'acquisition de compétences et de technologies en cybersécurité prônée par le Livre Blanc sur la Défense et la Sécurité Nationale de 2013, la Chaire de Cyberdéfense des Systèmes Navals mutualise autour d'un projet de recherche les ressources humaines, scienfitiques et techniques issues de la coopération académique et industrielle entre l'École Navale, ENSTA Bretagne, IMT Atlantique, Naval Group et Thales » [Sul+16]. À travers une production scientifique menée entre autres dans le cadre de doctorats – quatorze sujets ont ainsi été explorés depuis 2014, quatre d'entre eux ayant déjà donné lieu à des soutenances de thèse –, cette chaire industrielle participe à la réflexion et à la prospective en matière de cybersécurité des systèmes industriels, couvrant un large spectre de sa composante navale.

Comme nous l'avons évoqué précédemment, « l'application de la recherche en cybersécurité au domaine naval est, en effet, particulièrement importante à l'heure où les bâtiments – civils ou militaires – embarquent un grand nombre de systèmes informatiques contrôlant notamment des actionneurs mécaniques d'importance critique, ou permettant au navire de communiquer, se localiser, percevoir son environnement opérationnel. Toute bénéfique qu'elle soit en termes d'efficacité, de précision et de sûreté, la présence de ces systèmes informatiques peut ouvrir, nous l'avons vu, des brèches à un attaquant sachant les exploiter. Ces failles peuvent être logicielles, matérielles, organisationnelles ou humaines, et posent le problème de leur détection et de leur comblement. Véritable processus critique [du maintien en condition opérationnelle des systèmes navals], la maîtrise des correctifs remédiant à ces failles » est au cœur des travaux de recherche dont la synthèse est exposée dans le présent mémoire [Sul+16].

Objectifs des travaux et questions de recherche

L'objectif final des travaux que nous nous attacherons à exposer est d'établir un processus et un ensemble d'outils, métriques et modèles permettant une gestion maîtrisée des correctifs (ou contremesures) de sécurité appliqués à un système naval.

Cette problématique est complexe à plusieurs titres. Elle l'est, tout d'abord, dans la mesure où si le déploiement d'une contremesure vise à maîtriser au mieux le risque lié aux vulnérabilités d'un système, il peut lui-même induire des risques intolérables – par exemple, le déploiement d'un correctif logiciel peut imposer une indisponibilité temporaire de certains sous-systèmes critiques ; dans le cadre d'un navire, il n'est pas acceptable de perdre la propulsion pendant un déploiement du bâtiment à la mer. Par cette illustration, nous voyons apparaître un second facteur de complexité : l'impact d'une contremesure peut varier en fonction de l'activité en cours du système. Une gestion des correctifs rigoureuse nécessite donc une mesure des impacts que peuvent avoir une vulnérabilité, son exploitation, et une série de correctifs : cette mesure doit être comparable et pertinente ; une quantification de l'impact peut être, par exemple, économique : cette question a déjà été abordée à plusieurs reprises, notamment dans [Lin10 ; Pat+08] ; si cette approche est appropriée dans le cas d'un navire (notamment un porte-conteneurs), il semble encore plus pertinent de quantifier l'impact que peuvent avoir vulnérabilités et correctifs sur les missions du système. La complexité de la problématique, ensuite, est inhérente aux systèmes étudiés : un bâtiment est « un système complexe composé d'un ensemble de sous-systèmes, dont des systèmes cyber-physiques (CPS) [2], opérés par un équipage dans le but d'accomplir un ensemble de missions, lesquelles reposent sur des fonctions impliquant de nombreux acteurs ; par exemple, la navigation s'appuie sur des capteurs GNSS et/ou des centrales inertielles, des systèmes de traitement et d'analyse de leurs signaux, le choix de la route, l'opération des systèmes de propulsion » et de gouverne, une série d'actionneurs et d'automates élaborant les commandes afférentes, etc. [Sul+16] Il est donc nécessaire, afin de rendre compte au mieux des impacts sur les missions du navire, d'établir une mesure et une méthodologie de calcul d'impact les plus exhaustives que possible, mettant en évidence les phases des missions, les fonctions, les composants du système, ... qui peuvent être affectés.

2. On appelle *système cyber-physique* un système, souvent complexe, percevant et agissant sur son environnement par le biais de capteurs et d'actionneurs, dont la supervision et la commande s'effectuent numériquement.

De ce qui précède nous pouvons déduire plusieurs besoins, et questions de recherche, relatifs à la problématique traitée :

QR1 : Un besoin méthodologique : comment architecturer un processus de gestion de correctifs adapté ?

QR2 : Un besoin de modélisation : comment modéliser un système aussi complexe et aussi hétérogène qu'un navire ?

QR3 : Un besoin de conception de métrique : comment exprimer l'impact d'une vulnérabilité, de l'exploitation d'une vulnérabilité, d'une contremesure, sous forme de mesure comparable contenant suffisamment d'information ?

QR4 : Un besoin méthodologique subsidiaire : comment mener les calculs aboutissant à ces mesures ?

QR5 : Un besoin de modélisation subsidiaire : quelle modélisation des vulnérabilités, des attaques et des contremesures est appropriée dans la visée de QR3 et QR4 ?

Réponses apportées et plan du mémoire

L'apport des travaux présentés dans ce document se divise en deux axes principaux : d'une part, un apport méthodologique par l'introduction d'un processus de gestion de correctifs adapté au contexte des systèmes navals et d'une démarche de modélisation afférente ; d'autre part, la conception d'une mesure de l'impact comportemental d'une vulnérabilité, de son exploitation, et d'une contremesure. Nos différents chapitres s'attacheront à détailler ces apports en suivant la chronologie des tâches successives de notre processus : nous exposerons ainsi dans un premier temps l'architecture du processus proposé, sa motivation et ses présupposés technologiques, documentaires et scientifiques, avant d'évoquer la démarche de modélisation des systèmes navals retenue – ce chapitre nous permettra de faire le lien avec la partie suivante du mémoire, qui détaillera la méthodologie de calcul d'impact et la mesure associée que nous avons imaginée, avant de conclure par un chapitre de synthèse des expérimentations permettant d'éprouver ces apports sur des cas d'études réels ou fictifs. Nos contributions se répartissant de fait entre plusieurs champs de la littérature scientifique, nous proposerons, pour chacun des trois premiers chapitres, une revue de littérature spécifique [3] nous permettant de

3. Ces revues de littérature sont largemement issues des états de l'art que nous avons précédemment publiés dans [Sul+16] et [Sul+18].

les situer vis-à-vis des travaux proches des nôtres et concernés par la thématique du chapitre.

	QR1	**QR2**	**QR3**	**QR4**	**QR5**
Chapitre 1	■				
Chapitre 2		■			■
Chapitre 3			■	■	
Chapitre 4	*chapitre de synthèse des résultats*				

TABLE 1 – Matrice chapitres/questions de recherche

Étant donné cette partition de notre mémoire, nous pouvons établir une matrice illustrant, pour chacune de nos questions de recherche, les chapitres y apportant une réponse (*cf.* table 1).

Chapitre 1

Un processus de gestion des correctifs

Table des matières

La réponse méthodologique que nos travaux apportent à la problématique posée réside dans l'architecture d'un processus de gestion de correctifs, se déroulant sur l'ensemble du cycle de vie du navire et ayant pour but premier de permettre, en réponse à la découverte d'une vulnérabilité, le choix et le déploiement de contremesures en fournissant un ensemble d'indicateurs donnant une estimation de leur

innocuité et leur efficacité. Il ne s'agit pas ici d'évaluer ces deux caractéristiques sur le seul composant où seront éventuellement déployées ces contre-mesures, mais bien de les estimer par rapport au système dans son entièreté, et plus particulièrement de les estimer dans une optique fonctionnelle : quelles fonctionnalités le navire va-t-il perdre lors de la modification induite par la contre-mesure ? Sera-t-il toujours en mesure d'accomplir ses missions, partiellement ou totalement ?

Ce chapitre nous permettra d'exposer, après une revue de littérature des processus de *patch management* existants, le principe général du processus, de détailler ses différentes tâches, et de passer en revue les besoins technologiques afférents. Ainsi, nous expliquerons l'architecture globale du processus dans un premier temps ; cela nous permettra de mettre en exergue sa structure en deux phases inscrites dans des durées différentes – le temps « long » de la vie du système pour la première, celui, rapide, de la réaction à la découverte d'une vulnérabilité pour la seconde. Ensuite, nous nous attacherons à présenter dans le détail les différentes tâches constituant ces deux phases : l'analyse de la seconde sera également l'occasion de présenter un outil de veille, développé dans le cadre de nos travaux, permettant de détecter les vulnérabilités éventuelles affectant les composants et les fonctions du navire. Enfin, en nous appuyant sur les principes méthodologiques exposés tout au long du chapitre, nous mènerons un inventaire des technologies nécessaires à la bonne conduite de notre processus de gestion de correctifs.

Définitions liminaires

En préambule à ce chapitre, il semble opportun de définir plusieurs termes qui sont au centre de nos travaux et qui apparaîtront donc de manière récurrente dans ces pages. Nous proposons à cet égard les quatre définitions suivantes, que nous avons précédemment proposées dans [Sul+18] :

Définition 1 (**Système**). *Un système est une entité composée d'opérateurs, de procédures, d'équipements matériels et de logiciels. Les composants de cette entité œuvrent conjointement dans leur environnement opérationnel dans le but de fournir des fonctions permettant d'accomplir un ensemble de missions donné* [DoD95]. *Un système possède une politique de sécurité* [AEK09] *garantissant notamment la sûreté de fonctionnement de ses éléments.*

Définition 2 (**Vulnérabilité**). *Une vulnérabilité, ou faille, est une faiblesse dans la conception, la configuration, l'opération ou la maintenance d'un système, qui*

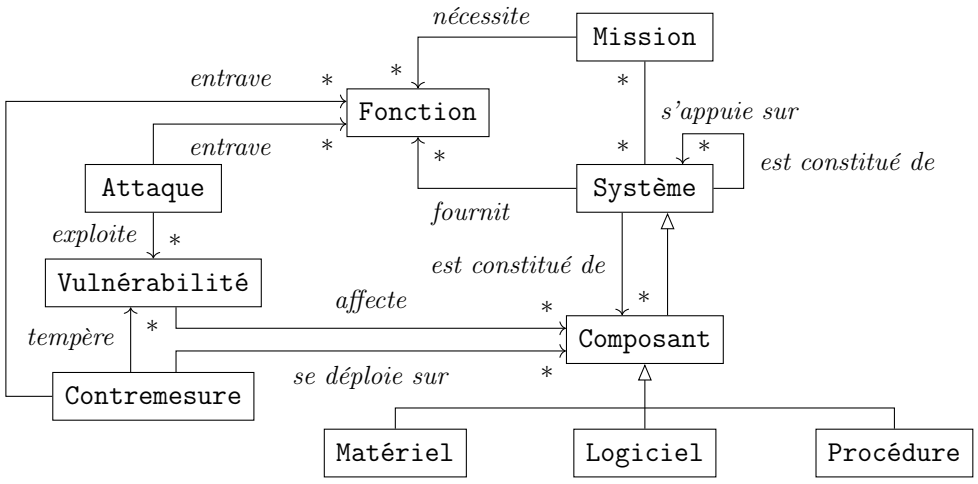


FIGURE 1.1 – Métamodèle d'un système naval (précédemment proposé dans [Sul+16 ; Sul+18]).

peut être exploitée pour contrevenir à sa politique de sécurité [Jaw93] *dans le but d'entraver ses fonctions et ses missions.*

Définition 3 (Contremesure – ou correctif). *Un correctif est une mesure, ou un ensemble de mesures (action, dispositif matériel, logiciel, procédure, ...) qui réduit la possibilité d'exploiter une vulnérabilité donnée* [Jaw93].

Les liens entre ces trois premières définitions sont illustrés sur le métamodèle que nous proposons en figure 1.1.

Définition 4 (Impact). *L'impact d'une vulnérabilité (resp. d'un correctif, d'une attaque) est l'ensemble des retombées d'une vulnérabilité (resp. d'un correctif, d'une attaque) sur la capacité du système à accomplir ses missions de manière sûre. Ces impacts permettent de comparer les conséquences de deux contremesures, d'une contremesure et d'une vulnérabilité, etc. Par conséquent, il s'agit d'un élément-clé afin de choisir la contremesure la plus efficace et la moins néfaste pour atténuer une vulnérabilité* [Sul+18].

1.1 Revue de littérature

La problématique méthodologique de la gestion des correctifs de sécurité, ou *patch management*, a été abordée dans la littérature à plusieurs reprises, notam-

ment par [BS03], [Cha+05], [Div08], [Liu+09] et [Gau+17] [1].

Bill Brykczynski and Robert A. Small introduisent dans [BS03] huit pratiques clés étayant un processus de gestion de correctifs efficace, à savoir :

1. *Establish Policies, Procedures, and Responsibilities*, recouvrant les activités d'adaptation des pratiques suivantes au contexte de la structure déployant un processus de gestion de correctifs (identification des acteurs impliqués, fréquence de veille des vulnérabilités, etc.) ;
2. *Maintain Awareness of IT Infrastructure*, recouvrant les activités de gestion de configuration ;
3. *Maintain Vulnerability Alert Resources*, correspondant à la constitution et à l'actualisation d'un ensemble de sources de collecte des alertes de sécurité relatives aux logiciels déployés au sein de la structure ;
4. *Monitor Vulnerability Alerts*, recouvrant les opérations d'analyse des alertes collectées en vertu du point précédent ;
5. *Assess and Respond to Vulnerability Alerts*, pour évaluer l'impact potentiel des attaques exploitant les vulnérabilités découvertes, identifier les correctifs logiciels disponibles et dans la négative, concevoir des mesures organisationnelles de contournement temporaire dans l'attente de sa publication, etc. ;
6. *Test and Evaluate Patches*, correspondant aux opérations de test sur une plateforme d'intégration afin de répondre aux deux questions suivantes : les correctifs sont-ils efficaces ? Vont-ils introduire des effets de bord intolérables ? ;
7. *Install Patches*, recouvrant les opérations de déploiement des correctifs sur les systèmes en production ;
8. *Measure and Improve the Process*, afin d'ajuster le processus tout au long de son cycle de vie.

Nous pouvons trouver la mise en application de pratiques similaires dans le processus proposé par Chuan-Wen Chang *et al.* dans [Cha+05] (*cf.* figure 1.2), dont la méthode sous-jacente s'appuie sur cinq étapes consécutives : suite à la découverte d'une vulnérabilité, la tâche de *réception d'informations relatives à la vulnérabilité (1)* précède l'*analyse d'impact (2)*, la *planification des tests et du*

1. L'essentiel de cette revue de littérature a fait l'objet d'une publication dans [Sul+18] et reprend le contenu – traduit – de cette publication.

déploiement (3) de la contremesure candidate à la correction de la vulnérabilité découverte ; les étapes de *test (4)* de la contremesure et de *déploiement automatisé et audit du correctif (5)* venant conclure le processus. Il est à noter que la tâche d'*analyse d'impact (2)* consiste principalement à mener une analyse périmétrique de la vulnérabilité, permettant d'identifier les composants qu'elle affecte. Par ailleurs, l'impact des contremesures est évalué par le biais des tâches (3) et (4), à travers des opérations de test.

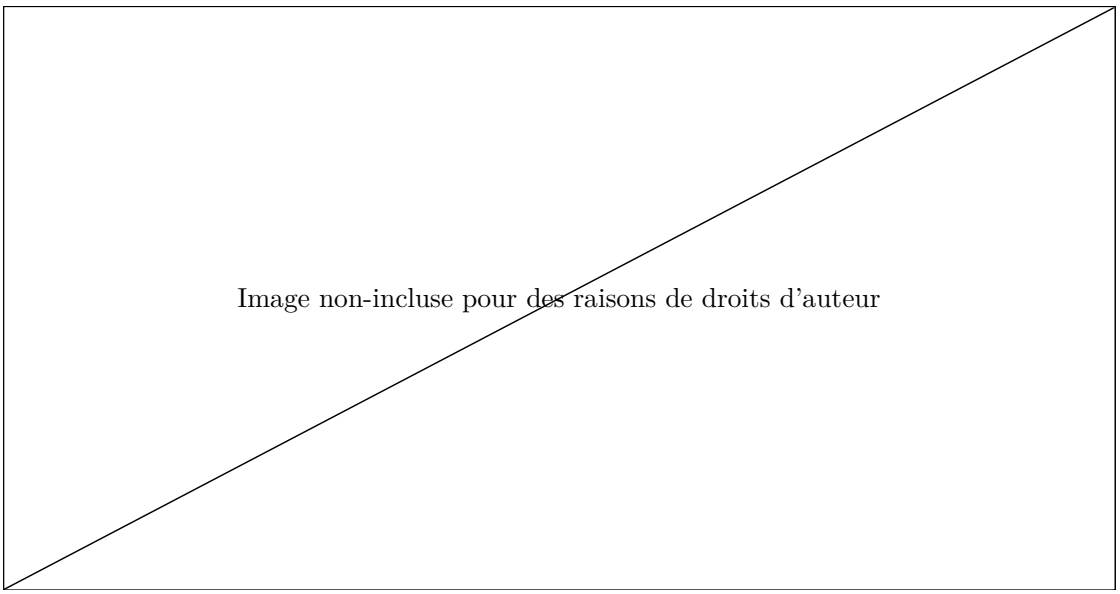


FIGURE 1.2 – Le processus proposé dans [Cha+05]. ©IEEE 2005

L'approche proposée par Simon Liu *et al.* dans [Liu+09] se fonde également sur des étapes similaires, soit :

1. *Establish Timely and Practical Alerts*, consistant à veiller les vulnérabilités logicielles applicables à la structure en fonction d'une gestion de configuration périodiquement mise à jour ;
2. *Receive Notification of Patches or Discover Them*, étape où la recherche des correctifs logiciels disponibles est conduite ;
3. *Download Patches and Documentation*, recouvrant les opérations de téléchargement et de vérification de l'intégrité de la contremesure logicielle ;
4. *Assess and Prioritize Vulnerabilities* : cette évaluation des vulnérabilités s'appuie sur CVSS , une métrique exprimant leur impact sur une échelle de 0 à 10 en synthétisant trois « scores » (score de base, intrinsèque à

la vulnérabilité, score « temporel », score « environnemental ») calculés d'après une évaluation qualitative d'une série de paramètres [2] ;

5. *Perform Testing*, étape où le correctif sera évalué une fois encore par son déploiement sur des environnements de test ;
6. *Deploy Patches* ;
7. *Audit and Assessment*, correspondant à la vérification de l'efficacité de la contremesure une fois déployée dans son environnement.

Plus récemment, Adam Gauci *et al.* [Gau+17] ont proposé une méthode de *patch management* adaptée aux réseaux de distribution d'énergie, s'appuyant également sur les étapes proposées par Bill Brykczynski and Robert A. Small [BS03] et insistant sur la nécessité de disposer d'un environnement de test, idéalement sous la forme de jumeau numérique du système, et d'un inventaire rigoureux des composants du système qui, corrélé à leur configuration logicielle ainsi qu'à la liste des contremesures disponibles, permet d'obtenir à tout moment une vision de l'écart entre l'état de sécurité du système et les recommandations des éditeurs des logiciels qu'il opère.

Si *Recommanded Practices for Patch Management of Control Systems* [Div08], établissant des pratiques clés d'un processus de gestion de correctifs appliqué à des systèmes de contrôle industriels (dont les PLC) insiste sur certaines actions déjà recommandées par les articles précédemment cités – en insistant notamment, avec un niveau de détail supplémentaire, sur les pratiques de gestion de configuration ou sur les tests d'intégration des contremesures logicielles, menées sur des environnements de test qui « simulent au plus près l'environnement opérationnel » auquel elles sont destinées –, de nouvelles notions sont introduites comme la mise en place d'un plan d'archivage et de sauvegarde du système, ou d'actions récurrentes d'audit des vulnérabilités non publiées par les éditeurs des COTS utilisés. Ce guide propose par ailleurs l'évaluation des vulnérabilités par le biais d'une empreinte constituée de quatre critères subjectifs (impact – au sens d'impact sur le composant affecté – exposition – au sens de chemin d'attaque – , simplicité d'exploitation, et déploiement – la version du logiciel affecté par la vulnérabilité est-elle déployée ?). Il s'agit toutefois d'une évaluation qualitative, et aucune méthodologie d'évaluation en amont de l'impact d'une contremesure n'est par ailleurs proposée : les éventuels problèmes sont une fois encore détectés lors de son intégration dans ses environnement successifs (de test puis de production).

2. La spécification de CVSS dans son dernier standard est disponible à l'adresse suivante : https://www.first.org/cvss/specification-document

De manière générale, la littérature existante dont ces quatre articles insiste sur l'importance primordiale du test des contremesures en amont ou en parallèle de leur déploiement, dans la visée d'assurer leur efficacité et leur innocuité ; aucun ne propose toutefois une méthodologie pratique d'évaluation des contremesures ou des vulnérabilités (à l'exception, pour ces dernières, de l'évaluation qualitative de [Div08]), et les métriques proposées pour l'expression des impacts des vulnérabilités découvertes manquent soit de nuance (évaluation qualitative reposant sur trois niveaux (*low*, *medium* et *high* dans [Div08]), soit d'exhaustivité (CVSS dans [Liu+09], qui synthétise les informations sous la forme d'un unique nombre) ni d'analyse d'impact des contremesures en-dehors de leur constatation directe lors d'un déploiement sur des environnements de test.

Par ailleurs, ces processus s'attachent à l'analyse de vulnérabilités et de contremesures uniquement logicielles, et ne permettent donc pas la prise en compte de faiblesses architecturales ou organisationnelles qui entrent dans le cadre de notre définition d'une vulnérabilité. Nous proposons donc, afin de répondre à notre première question de recherche QR1, un processus original – bien qu'inspiré des pratiques clés identifiées dans cette revue de littérature – dont la finalité est une prise de décision, suite à la découverte d'une vulnérabilité telle que nous la définissons, orientée par une analyse comportementale de cette vulnérabilité, des attaques pouvant l'exploiter et des contremesures candidates à sa correction.

1.2 Présentation générale du processus

Le processus que nous proposons[3] ayant pour tâche décisive l'analyse de l'impact fonctionnel, en amont de leur déploiement éventuel, des contre-mesures à disposition des organismes chargés du maintien en condition de sécurité du système étudié, il est nécessaire d'en construire une abstraction rendant compte au plus près de son comportement, afin de permettre la déduction des éléments nécessaires à cette analyse sans avoir besoin de tester chaque contre-mesure sur le système lui-même. Cette abstraction, pour être construite, nécessite une connaissance approfondie du système qu'elle modélise : il est donc nécessaire, en premier lieu, de collecter un ensemble de modèles du système. Ces modèles peuvent être des documents issus de la conception du bâtiment – architectures fonctionnelle,

3. La présente section se fonde partiellement sur le processus que nous avons défini dans [Sul+18] et les explications fournies dans le même article.

physique, topologique, etc. –, de ses utilisateurs (équipages, armateurs ou états-major) – description des missions, doctrine d'emploi, règlements, procédures, etc. –, ou encore des organismes chargés de son maintien en condition opérationnelle – notamment les documents liés à la gestion de configuration.

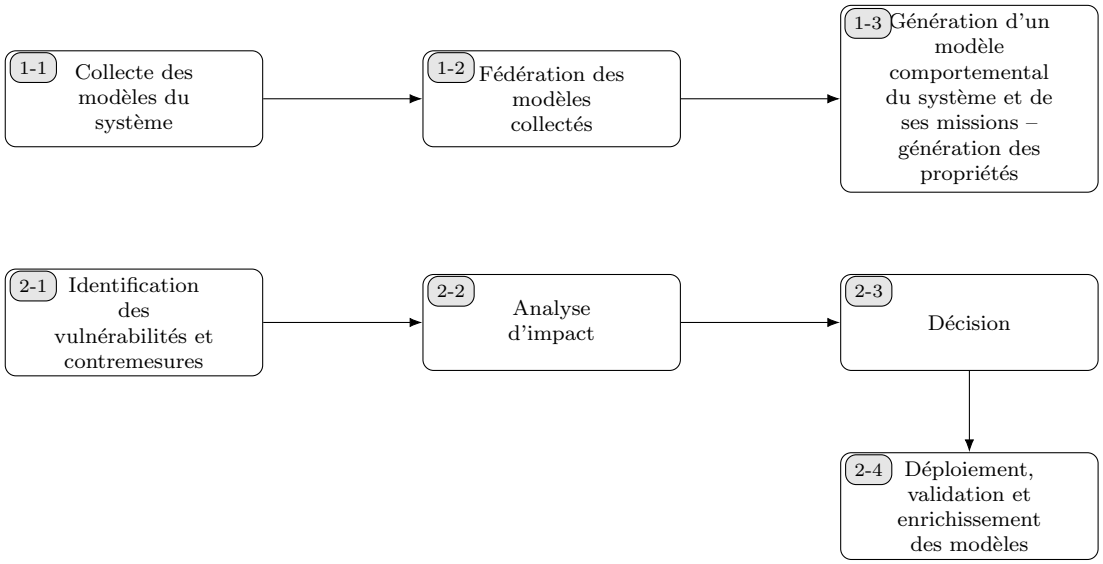


FIGURE 1.3 – Le processus de gestion de correctifs proposé [Sul+16 ; Sul+18].

Ces modèles collectés seront ensuite fédérés afin d'en extraire un ensemble de vues permettant de faire converger tous les éléments pertinents pour l'établissement de l'abstraction comportementale, qui en sera subséquemment déduite. Les tâches 1-1 à 1-3 du processus que nous proposons (*cf.* figure 1.3) répondent à cette nécessité de collecte, de fédération et de construction d'un modèle comportemental du système. Elles répondent également à l'objectif d'établissement d'une modélisation des différentes missions du système et d'un ensemble de propriétés de sûreté, nécessaires comme nous le verrons ultérieurement à l'estimation de l'impact des contre-mesures et des vulnérabilités déployées et affectant le navire. Ces tâches 1-1 à 1-3 constituent la première phase de notre processus, que nous désignerons par la suite comme la *phase de collecte, de modélisation et de maintien des connaissances*. Cette phase a vocation à s'étendre sur l'ensemble du cycle de vie du système : du fait de la durée de vie importante des systèmes industriels complexes, ces derniers sont en évolution constante – à titre d'exemple, la vie « opérationnelle » du porte-avions Charles de Gaulle est censée durer quarante ans et des chantiers de grande envergure sont régulièrement menés afin de moderniser les systèmes du bâtiment,

le dernier en date étant l'arrêt technique majeur (ATM-2) ou refonte à mi-vie de 2017, s'étant étendu sur dix-huit mois et ayant permis des modifications profondes des systèmes de combat et de plateforme, le remplacement d'automates et de capteurs, la modernisation des réseaux, etc. Il convient donc, à chaque mise à niveau technologique ou suite à toute autre modification du système étudié, de tenir à jour son modèle comportemental ainsi que ses propriétés de sûreté.

Cette phase « support » est le préalable à la seconde phase du processus, qui vise à détecter les vulnérabilités affectant le navire et fournir des indications utiles au choix de la réaction appropriée à travers une série d'analyses d'impact des différentes options disponibles. Cette seconde phase, que nous désignerons par la suite comme la *phase de veille et de réaction*, est composée des tâches 2-1 à 2-4 du processus de gestion de correctifs s'articulant entre elles de la manière suivante : la tâche 2-1 d'*identification des vulnérabilités et contremesures* est exécutée en permanence grâce entre autres à un outil de remontée et de filtrage d'alertes de sécurité que nous exposerons par la suite dans ce chapitre ; dès qu'une vulnérabilité affectant le système est découverte, une recherche des correctifs existants et/ou la conception de contremesures (qui peuvent être des mesures temporaires d'urgence) sont menées ; chacune de ces contremesures sera ensuite analysée au cours de la tâche 2-2 à l'issue de laquelle leurs impacts fonctionnels, ainsi que ceux de la vulnérabilité et des attaques susceptibles de l'exploiter, seront exprimés sous forme d'une mesure permettant leur comparaison, permettant ensuite la tâche 2-3 de décision où le choix de réaction face à la vulnérabilité sera arrêté ; ensuite, au cours de la tâche 2-4, les contremesures choisies seront déployées puis validées par une série de tests menés sur le système ; leur intégration sera enfin prise en compte dans la gestion de configuration du navire dont la modélisation comportementale sera enrichie en conséquence.

La suite du chapitre nous permettra d'exposer dans le détail ces sept tâches, leur fonctionnement et leurs prérequis documentaires et technologiques.

1.3 Phase de collecte, de modélisation et de maintien des connaissances

La première phase de notre processus, comme nous l'avons précédemment évoqué, est un ensemble d'opérations ayant pour produits un modèle comportemental du système, un modèle de ses missions et un ensemble de propriétés, et

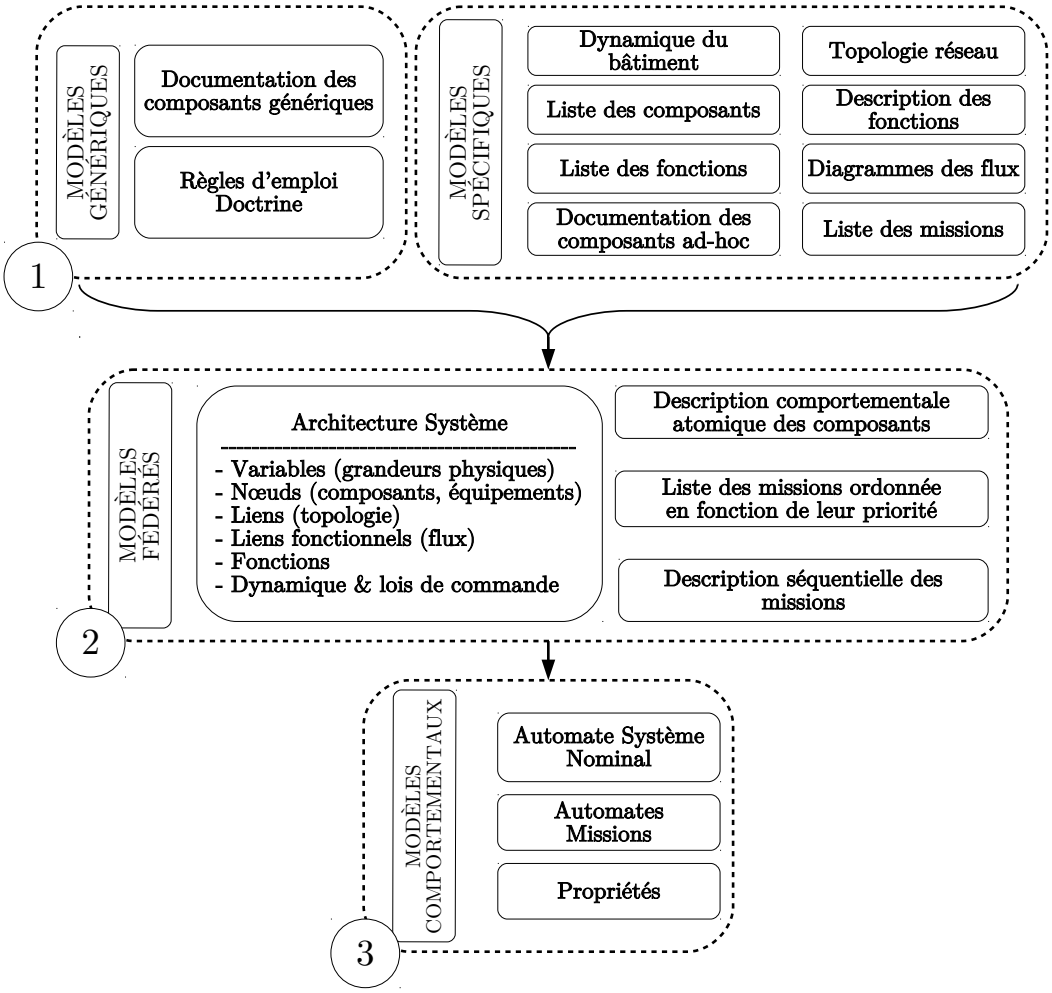


FIGURE 1.4 – Les corpus de modèles analysés et/ou générés durant la première phase du processus

pour substrat un ensemble de modèles d'entrée. Ces opérations peuvent être réparties en trois tâches successives, chacune aboutissant à un corpus de modèles issu de la collecte ou de la transformation de modèles d'entrée.

La figure 1.4 présente ces trois corpus, ainsi que les liens les unissant. La tâche 1-1 aboutit à la génération du corpus ① suite à une collecte de modèles ; la tâche 1-2 débouche sur la génération du corpus ② après transformation du corpus ① ; tandis que la tâche 1-3, en transformant le corpus ②, permet d'obtenir le corpus ③, qui est le produit final de cette première phase.

Nous nous attacherons dans les paragraphes qui vont suivre à présenter une méthodologie permettant ces générations successives de corpus de modèles, ainsi que les outils permettant d'automatiser les opérations pouvant l'être [4]. La présentation de chaque tâche suivra la même structure : ses pré-requis seront exposés dans un premier temps, avant une explication des opérations qui y sont rattachées et une présentation des éventuels outils associés, à laquelle succédera une synthèse des données de sortie.

1.3.1 Collecte des modèles du système (1-1)

1.3.1.1 Pré-requis

— Données d'entrée : identification du bâtiment ou de la classe de bâtiments étudié(e), des concepteurs, armateurs ou états-major, ainsi que des organismes chargés du maintien en conditions opérationnelles (MCO) du bâtiment ou de la classe de bâtiments ;
— Savoirs-faire : ingénierie des systèmes navals.

1.3.1.2 Opérations

Les opérations constituant cette tâche doivent être menées lors de la première application du processus à un système donné, mais également de manière périodique, tout au long de son cycle de vie, afin de maintenir une vue du système aussi actuelle que possible. Elles consistent principalement en une série d'entretiens avec les acteurs suivants :

— les concepteurs du système (nous appelons « concepteurs » les acteurs institutionnels ou industriels ayant pris part à la spécification, l'architecture et la réalisation du système – dans le cadre d'un bâtiment de la Marine nationale, il pourra s'agir de la DGA et de ses différents fournisseurs ; dans le cadre d'un bâtiment d'une compagnie d'armateurs, du bureau d'études des chantiers navals l'ayant conçu et de ses différents fournisseurs) ;
— l'autorité d'emploi : l'armateur dans le cas d'un bâtiment civil, l'état-major dans le cas d'un bâtiment militaire ;
— les organismes chargés du MCO (par exemple, le service de soutien de la flotte – SSF – pour la Marine nationale, les sociétés – filiales ou sous-

4. La justification des formalismes retenus sera détaillée au chapitre 2.

traitantes – chargées de la gestion des flottes pour les compagnies d'armateurs ou les affréteurs – CMA Ships, Genavir, etc.) ;
— l'équipage du bâtiment étudié.

Ces entretiens ont pour but de collecter un ensemble de modèles aussi complet et aussi pertinent que possible : à cet égard, les entretiens doivent s'attacher à dépasser la collecte de documents pré-existants, étant donné que ces derniers ne sont pas toujours réalistes (il peut exister des variations entre le fonctionnement prévu d'un système et son fonctionnement constaté), et à formaliser ces écarts (par leur transcription dans les modèles existants ou par la création de nouveaux modèles à l'issue des entretiens) qui revêtent souvent la forme de pratiques non formalisées (« on branche directement telle baie réseau à telle autre », « on débranche tel équipement sur telle plage horaire », etc.).

Au-delà et en complément des entretiens avec les différents acteurs identifiés ci-dessus, des opérations de recherche documentaire peuvent également être menées dans le cadre de cette première phase afin de collecter tout modèle utile et non-spécifique au bâtiment étudié (documentation d'un composant générique, etc.).

1.3.1.3 Sortie

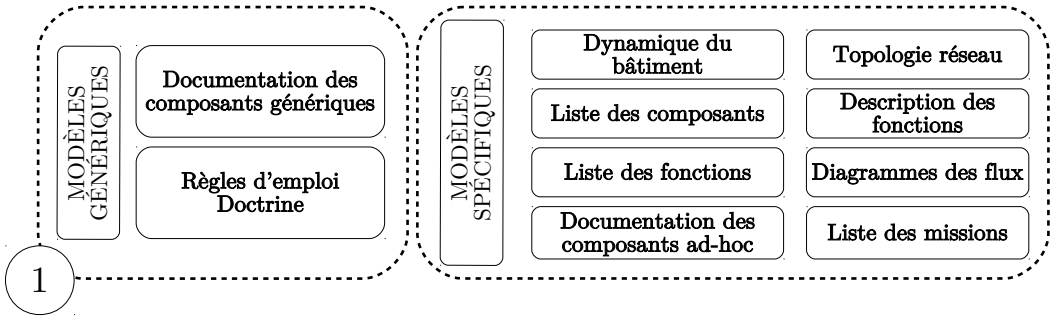


FIGURE 1.5 – Le corpus de modèles issu de la tâche 1-1

À l'issue de cette première tâche, un premier corpus de modèles (*cf.* figure 1.5) est constitué. Il est divisé en deux classes : les modèles « génériques », applicables à différents systèmes, et les modèles « spécifiques », propres au bâtiment (ou à la classe de bâtiments) étudié [Sul+18]. Les informations collectées au cours de cette

tâche incluent *a minima* les suivantes [Sul+18] :

- **Modèles génériques**
  - La documentation des composants génériques (COTS) utilisés ;
  - Les éléments de doctrine navale, ou de règles d'emploi, relatifs au déploiement des moyens de la marine militaire ou de l'armateur concerné ;
- **Modèles spécifiques**
  - La description de la dynamique du système (les équations différentielles décrivant l'évolution de ses grandeurs physiques – position, cap, vitesse, etc.) ;
  - La liste des composants du système, incluant le cas échéant l'identification des logiciels déployés sur chaque composant et leurs numéros de version ;
  - La documentation des composants spécifiques utilisés ;
  - La topologie réseau ;
  - Les diagrammes de flux, mettant en évidence les flux entre les différents composants du système lors de l'exécution des fonctions auxquels ils concourent ;
  - La liste des fonctions du système ;
  - La description des fonctions listées, incluant notamment leur décomposition fonctionnelle, l'identification des sous-systèmes et composants concourant à leur exécution, et les modifications des grandeurs physiques liées aux actionneurs du système et impliquées par leur exécution (par exemple la loi de commande liant la consigne de cap de l'opérateur et les angles de barres réalisés) ;
  - La liste des missions du système.

1.3.2 Fédération des modèles collectés (1-2)

1.3.2.1 Pré-requis

- Données d'entrée : le corpus de modèles ① obtenu à l'issue de la tâche 1-1 ;
- Savoirs-faire : ingénierie des systèmes navals, modélisation des systèmes sous forme d'automates finis temporisés.

1.3.2.2 Opérations

La visée de cette tâche est d'analyser le corpus de modèles initial afin d'en extraire les informations nécessaires à l'établissement du modèle comportemental du système, des modèles des missions et des propriétés nécessaires au calcul d'impact. En effet, le corpus ① étant composé de modèles exprimés dans des formalismes hétérogènes – et variables d'un bâtiment étudié à un autre –, son exploitation dans sa forme « brute » ajouterait une complexité forte aux opérations de génération des modèles finaux. Nous proposons donc la construction d'un corpus intermédiaire, regroupant à travers quatre classes de modèles les données nécessaires à ces opérations de génération. Ces quatre classes, ou *vues* [Guy+13] de notre corpus de modèles ①, sont :

- Le **modèle d'architecture système**, synthétisant les informations suivantes :
  - Les variables (grandeurs physiques liées au système – i.e. cap, vitesse, ..., à ses actionneurs – i.e. angle de barre, vitesse de rotation des moteurs, ..., aux consignes – i.e. cap désiré, vitesse désirée, ...) ;
  - Les équations liant l'évolution de ces variables entre elles lors de l'exécution des fonctions du système (lois de commande et équations descriptives de la dynamique du navire) ;
  - Les nœuds (composants et équipements réseau) ;
  - Les liens topologiques entre les différents composants ;
  - Les liens fonctionnels (les flux) entre les différents composants ;
  - La séquence des flux lors de l'exécution des fonctions du système ;
- Les **modèles comportementaux de chaque composant**, exprimés sous la forme d'automates temporisés [5] et dans un formalisme standard rendant compte, pour chaque automate, des informations suivantes :
  - L'ensemble de ses états ;
  - Son état initial ;
  - L'ensemble de ses horloges ;
  - L'ensemble de ses invariants ;
  - L'ensemble de ses transitions, chacune incluant les éventuelles gardes, mises à jour et action associées ;
- Une **description séquentielle de chaque mission** ;
- Une **liste des missions ordonnées selon leur criticité**.

5. *cf.* chapitre 2, section 2.2.

L'expression de ces quatre classes devra suivre des règles syntaxiques précises, exposées en annexe A, afin de permettre les opérations de génération de la tâche 1-3.

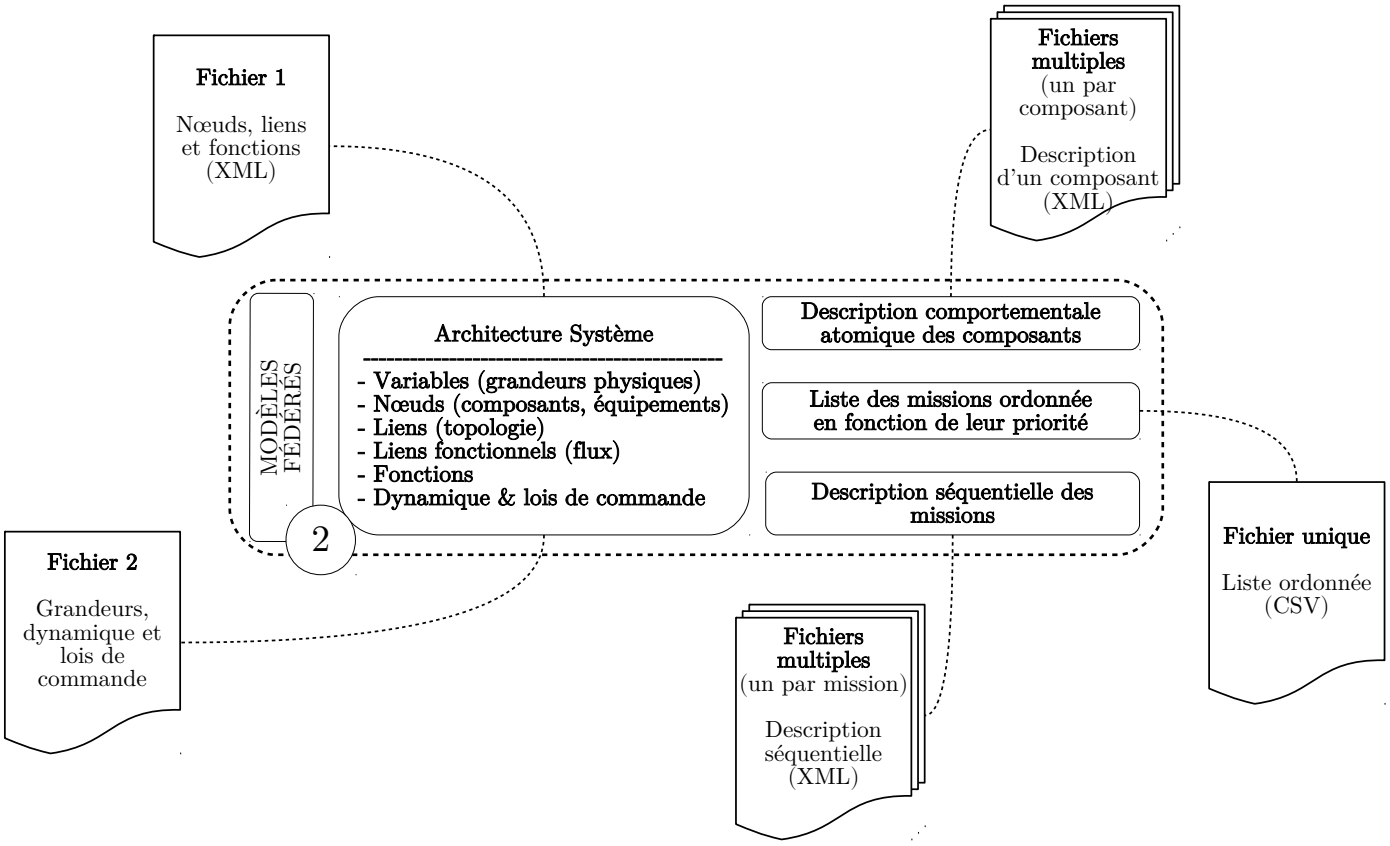


FIGURE 1.6 – Formalisme d'expression des vues extraites de la fédération de modèles (pour plus de détails, se reporter à l'annexe A).

1.3.2.3 Sortie

À l'issue de la tâche 1-2, le corpus ② de modèles intermédiaire est constitué (*cf.* figure 1.7). Ces quatre types de modèles, correspondant à autant de vues distinctes de l'ensemble des modèles fédérés, constituera le substrat de la génération du modèle comportemental exécutable que nous allons à présent exposer.

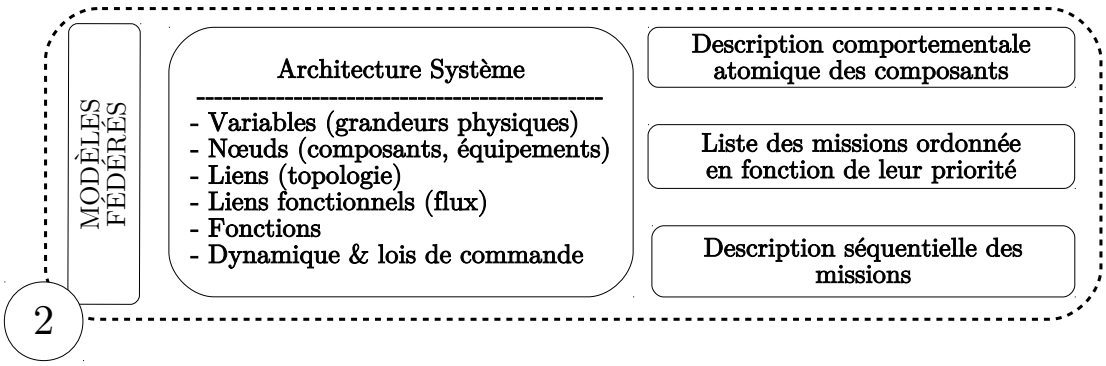


FIGURE 1.7 – Le corpus de modèles issu de la tâche 1-2

1.3.3 Génération d'un modèle comportemental du système, des ses missions et des propriétés

1.3.3.1 Pré-requis

— Données d'entrée : le corpus de modèles ②, issu de la tâche 1-2 ;
— Outils : une implémentation des chaînes de compilation permettant d'automatiser partiellement la génération du modèle comportemental, adaptée au formalisme retenu pour son expression ;
— Savoirs-faire : ingénierie des systèmes navals, modélisation des systèmes sous forme d'automates finis temporisés, *model-checking*.

1.3.3.2 Opérations

Le but de la tâche 1-3 est la génération d'un ensemble de modèles comportementaux du système, à partir des vues extraites de la fédération des modèles du corpus ①, et fournies au sein du corpus ②. Cette modélisation comportementale est composée des trois classes suivantes :

— l'Automate Système Nominal ;
— les Automates Mission ;
— les Propriétés.

Ces trois classes sont exprimées dans des formalismes laissés à la discrétion des organismes chargés de la mise en œvre du processus ; leur choix devra toutefois se conformer aux contraintes suivantes :

— l'Automate Système Nominal et les Automates Mission doivent être générés dans le même langage ;

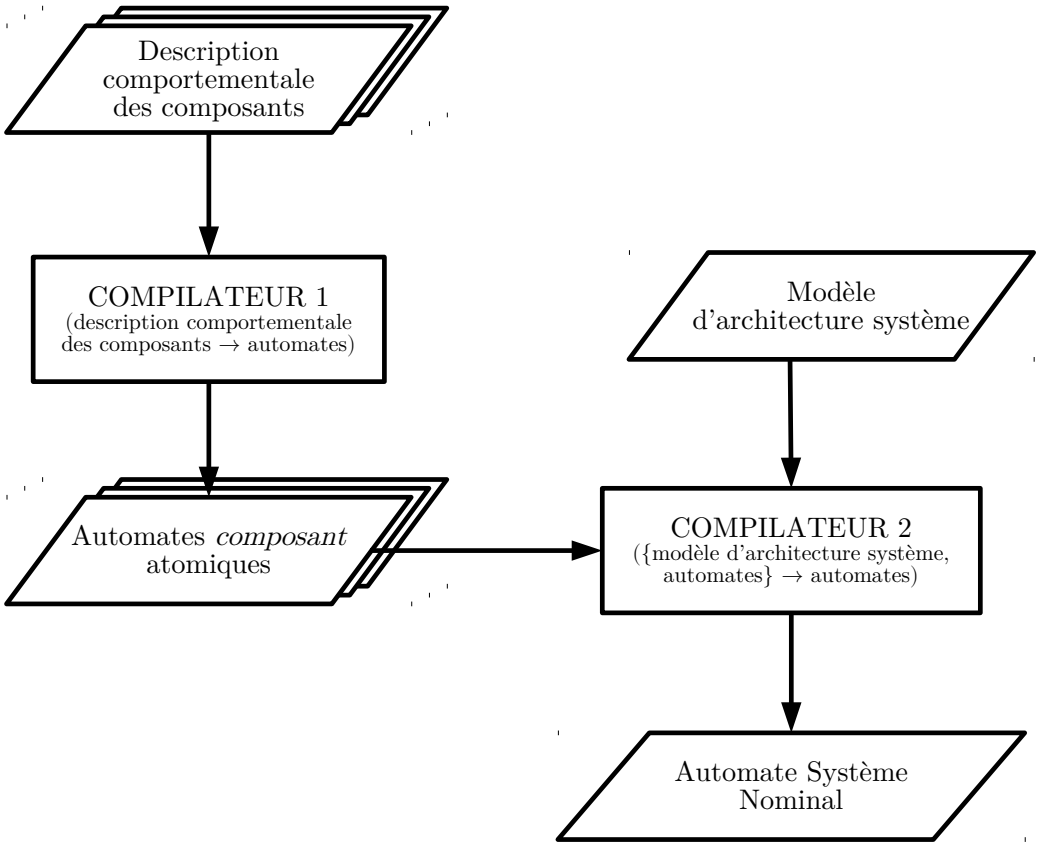


FIGURE 1.8 – Vue générale de la chaîne de compilation du modèle exécutable

— le langage pré-cité doit pouvoir décrire des automates finis temporisés permettant l'utilisation de variables, tels que formalisés dans [Beh+04] ;
— le langage d'expression des propriétés doit être compatible avec un moteur de vérification formelle pouvant calculer leur validité sur l'Automate Système Nominal exprimé dans le langage retenu ;
— il doit par ailleurs permettre la vérification des différents types de propriétés sur lesquels nous reviendrons ultérieurement ; notons ici qu'il devra permettre la vérification de propriétés traduisant des contraintes temporelles ainsi que de propriétés ayant trait à des tests sur les variables associées aux automates testés.

Génération de l'Automate Système Nominal

La génération de l'Automate Système Nominal nécessite une synthèse des infor-

mations contenues dans le modèle d'architecture système d'une part, et d'autre part de celles fournies par la description comportementale des composants.

Étant donné les formalismes de modélisation proposés en amont, il est possible d'automatiser cette tâche dans une large mesure en concevant un outil basé sur la chaîne de compilation présentée en figure 1.8), mobilisant deux compilateurs intervenant séquentiellement à partir des deux classes de modèles d'entrée nécessaires à la génération de l'Automate Système Nominal. L'implémentation de cette chaîne permet :

— de transcrire la description comportementale des composants en autant d'automates *composant* exécutables (un par composant modélisé) ;
— de générer les automates exécutables intermédiaires figurant la topologie réseau (automates simples liant deux automates *composant* entre eux dans le cadre d'un échange réseau en modélisant un lien bi ou monodirectionnel) ;
— de calculer et générer les automates *processus* exécutables (automates de niveau intermédiaire permettant la synchronisation des Automates Mission et des automates *composants*) ;
— de calculer et générer le modèle permettant la composition de ces automates en un automate hiérarchique exécutable, l'*Automate Système Nominal*.

Notons qu'étant donné la complexité des modèles d'entrée et la simplicité de la chaîne de compilation qui demeure un prototype de validation du concept d'automatisation, ces opérations de génération automatisée sont plus à concevoir comme une aide à la construction et une fois l'Automate Système Nominal compilé, il conviendra dans tous les cas de le tester (en confrontant notamment les mises à jour des variables globales éventuellement définies dans les transitions des automates *composant* avec les lois de commande définies dans le second volet du modèle d'architecture système issu du corpus ②), et éventuellement de le compléter.

Toutes les fonctionnalités offertes par les langages de modélisation des automates finis temporisés ne sont en effet pas prises en compte par la chaîne de compilation présentée : possibilité peut être par exemple laissée à l'utilisateur de programmer des fonctions affectant les variables et les horloges par effet de bord lors de l'exécution d'une transition. Cela peut être utile entre autres afin de calculer la variable globale entière *rudder_ angle* correspondant à l'angle de barre réalisé par l'appareil à gouverner du bâtiment, modifiée lors de l'exécution des automates

impliqués dans la fonction de gouverne. Une fonction permettant de modifier cette variable globale pourra donc être développée et associée à une transition semblant pertinente à l'utilisateur : par exemple, celle liant *receivingSensorData* et *waiting-Setpoint* au sein de l'automate décrivant le fonctionnement du PLC présenté plus tôt. Le second volet du modèle d'architecture système du corpus ② servira à nouveau à ce stade, à des fins cette fois-ci de génération de ces fonctions de mise à jour dans l'éventualité où certaines modifications des grandeurs physiques du système telles que décrites dans ce document sont trop compliquées pour figurer au sein des transitions des automates *composants* sous la forme d'une simple formule insérée en tant qu'élément de la liste des mises à jour des variables. Cette liste contiendra donc après modification, à la place d'une formule, l'appel de la fonction ainsi développée.

Il est à noter par ailleurs que les outils développés pour valider le concept de la chaîne de compilation génèrent actuellement des modèles exécutables dans le formalisme UPPAAL [Beh+04], mais qu'elle peut bien entendu être adaptée à tout langage de modélisation d'automates finis temporisés, afin de répondre aux attentes et besoins des analystes chargés de la mise en œuvre du processus.

Génération des Automates Mission

La génération des Automates Mission (*cf.* figure 1.9) est une opération beaucoup plus simple que celle ayant trait à la génération de l'Automate Système Nominal. En effet, il suffit à cette étape de transcrire les descriptions séquentielles des missions issues de la tâche 1-2 en autant d'automates exécutables exprimés dans le langage retenu.

Comme expliqué précédemment, les outils développés afin de valider le concept de compilation des Automates Mission permettent la génération d'automates dans le formalisme UPPAAL mais ils sont aisément adaptables en ce que le principe demeure le même quel que soit le langage de modélisation choisi.

Génération des Propriétés

Les propriétés sont conçues sur la base d'une analyse humaine. Elles servent de base à la vérification, dans le cadre du calcul d'impact mené à la tâche 2-2 et s'appuyant comme nous le verrons par la suite sur des opérations de *model-checking*,

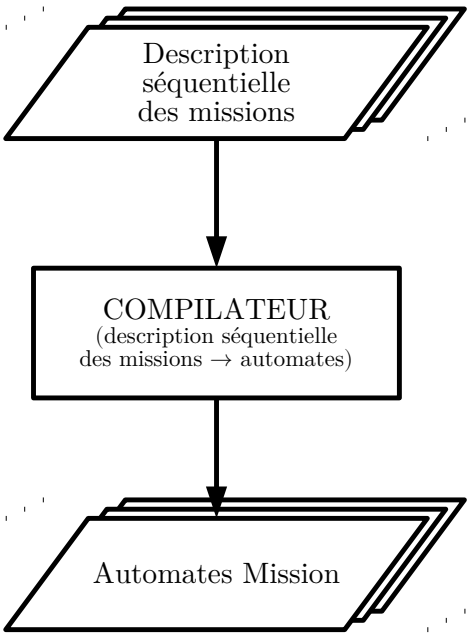


FIGURE 1.9 – Vue générale de la chaîne de compilation des Automates Mission

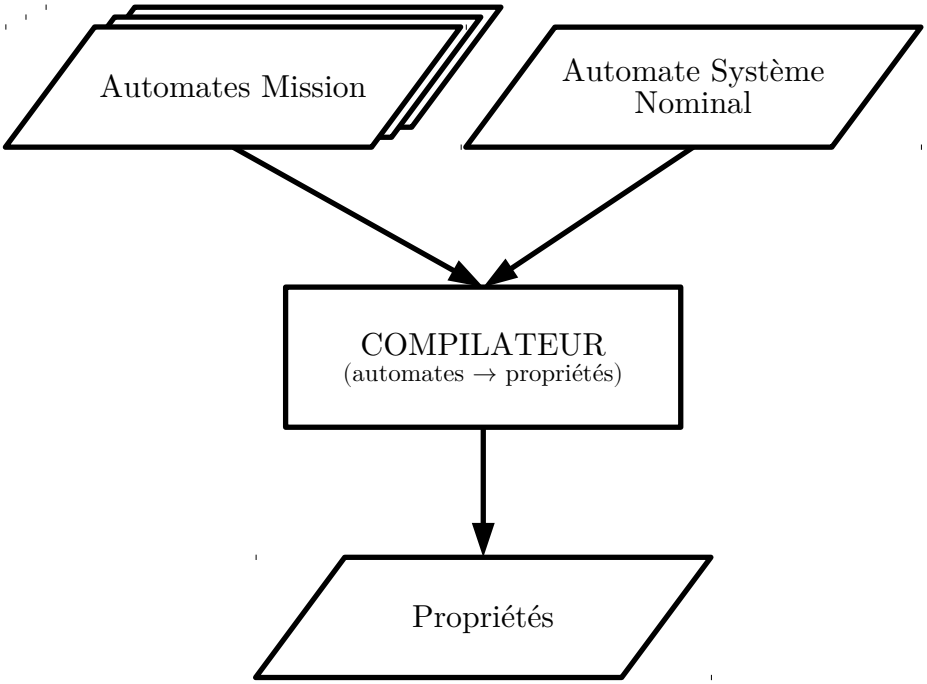


FIGURE 1.10 – Vue générale de la chaîne de compilation partielle des Propriétés

d'exigences de sûreté de fonctionnement ou de sécurité permettant d'évaluer en tout premier lieu la capacité du système à accomplir ses missions, mais aussi l'intégrité de ses composants, la conformité du comportement de ces derniers à leur spécification, la capacité du système à accomplir ses missions dans un délai précis ou la confidentialité des données qu'il traite. Elles sont choisies par les opérateurs du processus en fonction de la granularité de vérification souhaitée, mais doivent a minima permettre de vérifier l'atteignabilité de l'état final de chaque automate mission. Certaines d'entre elles peuvent toutefois être générées automatiquement, notamment celles ayant trait à la conformité des composants à leur spécification dans le contexte d'une mission (la propriété à vérifier étant que pour un automate composant donné, chaque état nécessaire à l'exécution de la mission auquel cet automate concourt est atteignable), ou à la capacité du système à accomplir une mission donnée (il suffit de vérifier que l'état final de l'automate mission afférent peut être atteint).

À ces causes, nous proposons une chaîne de compilation permettant une génération de ces propriétés afin d'alléger le travail d'analyse (*cf.* figure 1.10). Son principe est une fois encore très simple : il suffit d'extraire de l'Automate Système Nominal le nom et le nom des états de chaque automate *composant* puis de générer la propriété de conformité dans le langage de vérification retenu ; et d'extraire des Automates Mission le nom de chacun d'entre eux puis de générer les propriétés d'atteignabilité correspondantes dans ce même langage de vérification.

Nous obtenons, à l'issue de ces opérations de génération des propriétés, un ensemble de $\mathcal{P}$ qui peut se décomposer en $n+1$ sous-ensembles, n étant le nombre de missions du système : n sous-ensembles de propriétés dont l'expression est liée à chaque mission (regroupant les propriétés traduisant la capacité du système à accomplir la mission, à l'accomplir dans un délai précis, et les propriétés de conformité du comportement de chaque composant dans le contexte de la mission), et un sous-ensemble de propriétés dont l'expression est indépendante des missions (regroupant les propriétés ayant trait à l'intégrité de chaque composant et à la confidentialité des données traitées par le système).

1.3.3.3 Sortie

Cette tâche 1-3 concluant la première phase de notre processus résulte en un dernier corpus ③ de modèles comportementaux, correspondant au « produit » final des trois tâches de modélisation successive. Ces modèles serviront de base aux

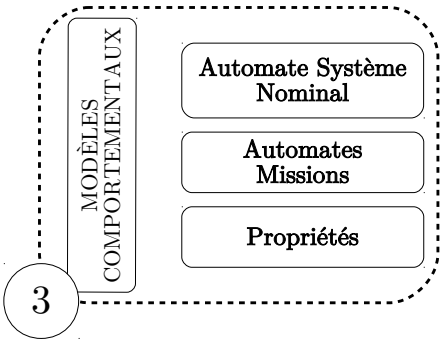


FIGURE 1.11 – Le corpus de modèles issu de la tâche 1-3

opérations de calcul d'impact subséquentes à la découverte d'une vulnérabilité et qui s'intègrent dans la deuxième phase du processus de gestion de correctifs, que nous nous attacherons à présenter dans la section qui va suivre.

1.4 Phase de veille et de réaction

Si les trois tâches précédentes s'inscrivaient dans le temps long de la durée de vie d'un système industriel, la seconde phase du processus proposé se définit comme une séquence double, où cohabitent une tâche (2-1) de veille des vulnérabilités affectant le système, exécutée en permanence tout au long de sa vie, et une suite d'opérations de réponse à la découverte d'une de ces vulnérabilités (2-2 à 2-4) – exécutée en un temps nécessairement plus contenu.

Symétriquement à la phase de modélisation où les corpus ①, ② et ③ sont construits, ces opérations mènent chacune à un enrichissement de cette base de modélisation par l'introduction et la prise en compte de nouvelles connaissances ayant trait aux vulnérabilités et attaques (données apparaissant en rouge sur la figure 1.12) et aux contremesures (éléments figurant en vert sur le même schéma). En effet, la tâche 2-1 consistant à veiller les vulnérabilités (et suite à la découverte de l'une d'entre elles, les possibles attaques et contremesures associées) apporte au corpus ① six nouvelles classes de modèles. La tâche 2-2, dans sa visée de calculer l'impact d'une vulnérabilité découverte, de ses attaques et contremesures afférentes, enrichit le corpus ② et le corpus ③ de respectivement trois et quatre nouveaux ensembles de modèles. Si la tâche 2-3, bornée à la prise de décision subséquente à la découverte d'une vulnérabilité sur la base des résultats du calcul des différents impacts, n'entraîne pas de modification de notre base de modélisation,

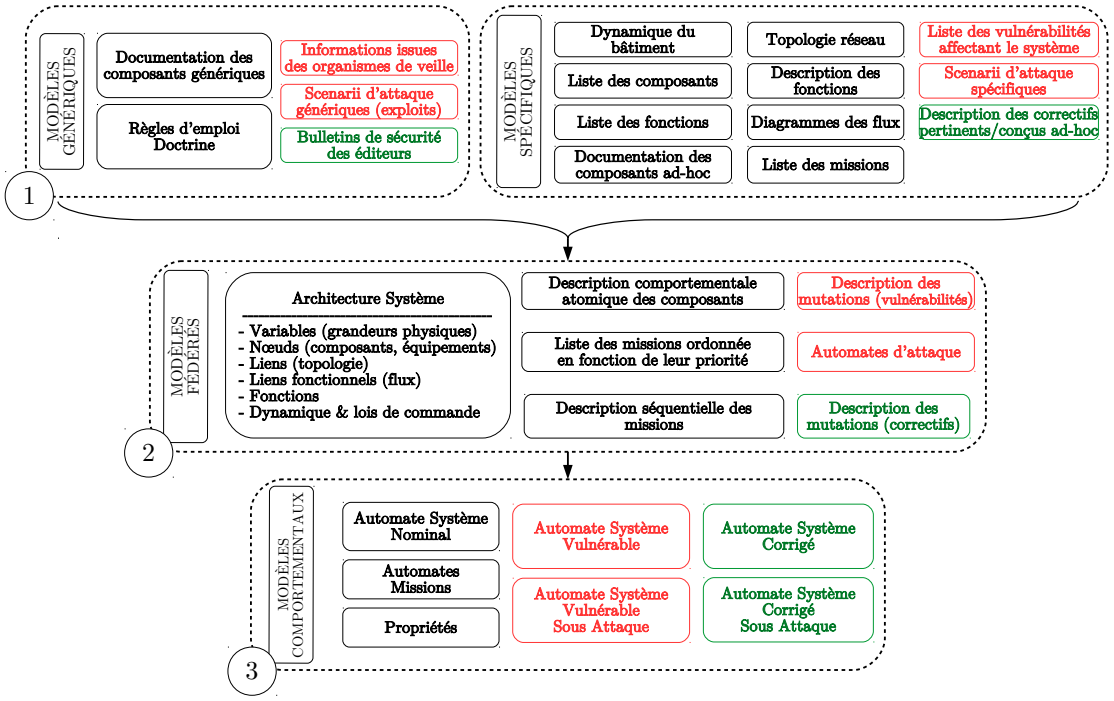


FIGURE 1.12 – Les différentes classes de modèles analysées et générées (en rouge et vert sur la figure) durant la phase de veille et de réaction.

la tâche 2-4 l'enrichira potentiellement en ce qu'elle provoque, dans l'hypothèse où une contremesure est déployée sur le système, une mise à jour du corpus ①.

1.4.1 Identification des vulnérabilités et contremesures

La première tâche de la phase de veille et de réaction s'ancre dans le cadre de la première de ses finalités : la veille des vulnérabilités affectant le système. À cet égard, ses opérations permettent, tout au long du cycle de vie du bâtiment, d'agréger et d'analyser un maximum d'informations pertinentes dans l'optique de déceler ces faiblesses. Entrent également dans le cadre de la tâche 2-1 l'identification des contremesures relatives aux failles révélées : collecte des mesures correctives proposées par les éditeurs de COTS et conception éventuelle de contremesures *ad-hoc* complètent donc le panel d'actions menées au cours de cette tâche.

1.4.1.1 Pré-requis

— Données d'entrée : architecture fonctionnelle du système (**liste des fonctions** et **description des fonctions** du corpus ①), éléments de gestion de configuration (logiciels et versions utilisés, déploiement de ces derniers composant par composant – ces informations sont fournies par la **liste des composants** du corpus ①) ;
— Outils : logiciel de veille VulneRobot, outils d'audit de code et d'audits techniques divers (analyse réseau, scan de vulnérabilités, etc.) ;
— Savoirs-faire : veille et analyse des avis de sécurité issus des bases de données publiques et utilisation d'outils associés, audits organisationnels et techniques et utilisation des outils associés ; dans le cadre de la conception de contremesures : ingénierie des systèmes cyber-physiques, connaissance du contexte opérationnel du système.

1.4.1.2 Opérations

Comme évoqué précédemment, les opérations de la tâche 2-1 peuvent se répartir en deux catégories : les opérations de veille des vulnérabilités d'une part, et celles de collecte et/ou d'élaboration des contremesures d'autre part.

Veille des vulnérabilités
La veille des vulnérabilités s'opère, dans le cadre du processus proposé, à travers deux approches. La première consiste à agréger continuellement, par le biais des bulletins de sécurité appropriés (publiés par l'ANSSI, le NIST, ...), les informations issues des concepteurs des COTS exécutés par les composants du système et à lever une alerte dès qu'une version déployée de l'un d'entre eux apparaît vulnérable ; ces opérations, fastidieuses par nature, ont fait l'objet d'une automatisation grâce au développement d'un outil de collecte et de synthèse. La seconde approche impliquée dans la tâche de veille mobilise par essence des opérations plus diverses en ce qu'elle vise à découvrir les faiblesses inhérentes à l'architecture du système, à ses procédures opérationnelles et à ses logiciels conçus spécifiquement. Elle implique la réalisation régulière de tests de pénétration du bâtiment, d'audits organisationnels et techniques et résulte également – en ce qui intéresse la tâche de veille du moins – dans la levée d'alertes à la découverte d'une vulnérabilité affectant le système.

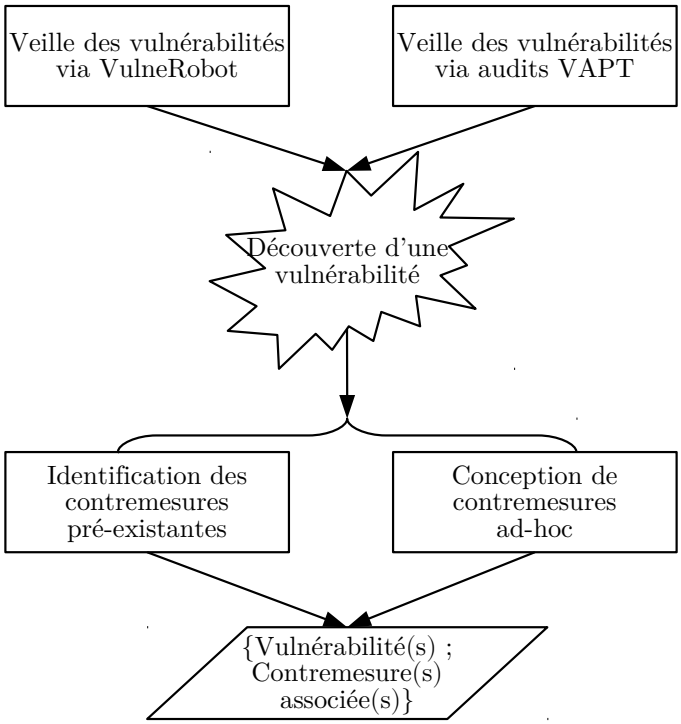


FIGURE 1.13 – Séquence des opérations de la tâche 2-1

i. Veille des vulnérabilités via VulneRobot

La première approche de veille, comme évoqué précédemment, s'appuie sur un logiciel dédié permettant de collecter périodiquement les avis de sécurité faisant état des vulnérabilités relatives à la configuration logicielle du système [GM17a]. Cet outil, que nous nommerons VulneRobot et qui a été conçu ad-hoc pour les besoins de ces travaux[6], a pour « moteur » un robot de collecte (*cf.* figure 1.14) parcourant un ensemble pré-défini de sites web mettant à disposition les avis de sécurité pertinents eu égard à la configuration logicielle du système. Étant donné que chacun de ces sites utilise un formalisme spécifique, VulneRobot dispose d'autant de plugins permettant l'analyse syntaxique des données qui y figurent ; cette architecture autorise une plus grande modularité en ce qu'il suffit de développer un nouveau plugin afin d'intégrer une source d'information supplémentaire à l'en-

6. NB : le développement de VulneRobot a été mené par Antoine Girard et Cyril Manda dans le cadre d'un projet de troisième année d'IMT Atlantique, et son code source ainsi que de plus amples explications sur son fonctionnement peuvent être trouvées au sein de leur répertoire GitHub et de leur rapport, accessibles via : https://github.com/linuxisnotunix/Vulnerobot.

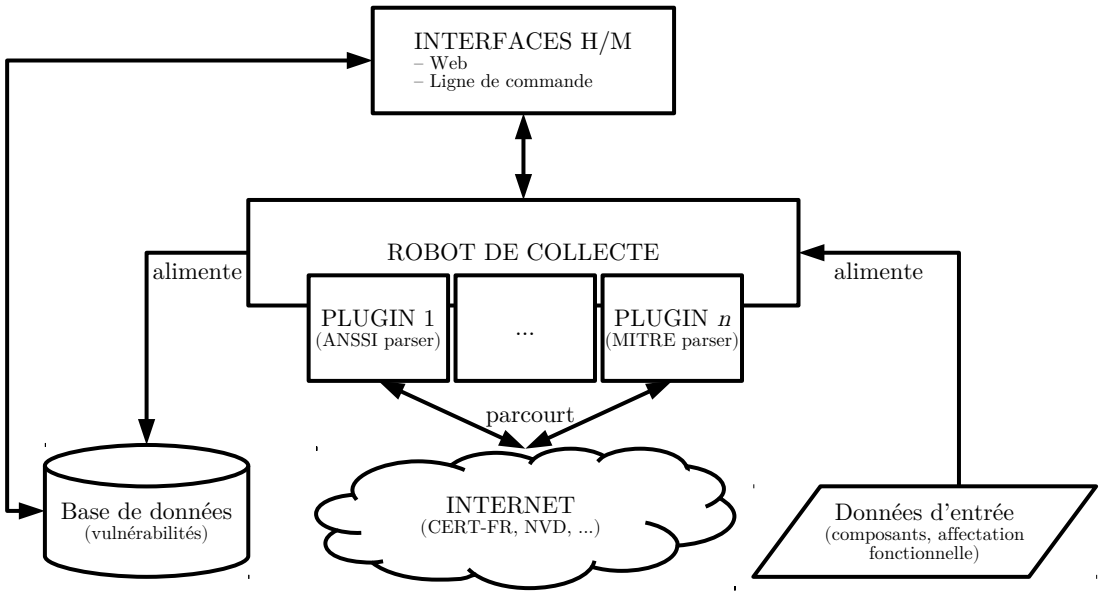
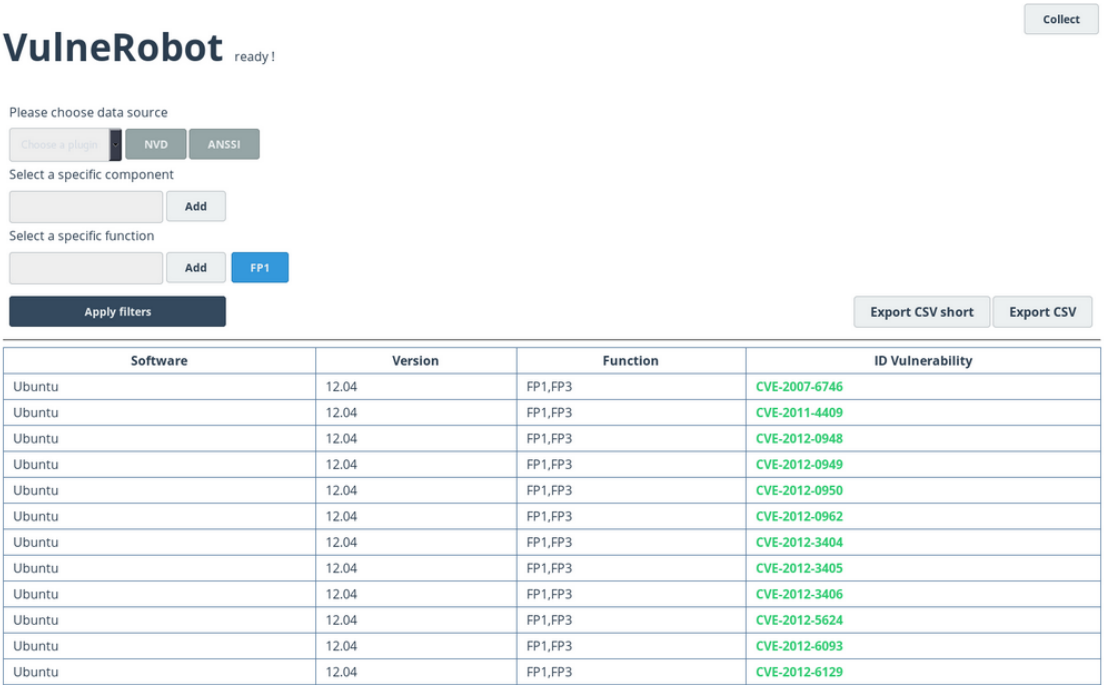


FIGURE 1.14 – Architecture du logiciel de veille VulneRobot

semble des sites de référence, ou de modifier les plugins existants si une évolution du formalisme des sites de référence survient. Dans l'état actuel des choses, VulneRobot dispose de deux de ces modules, un permettant la prise en compte des avis de sécurité du CERT-FR [7] (centre gouvernemental de veille, d'alerte et de réponse aux attaques informatiques mis en œuvre par l'ANSSI) et l'autre assurant l'interface avec la base NVD (*National Vulnerability Database*) [8] documentée par le gouvernement des États-Unis et diffusée par le NIST.

À partir de ces éléments collectés et analysés ainsi que du fichier décrivant la configuration logicielle du système et fourni par l'utilisateur, VulneRobot alimente une base de données où les vulnérabilités pertinentes sont enregistrées. Cette base de données peut être consultée par le biais de deux interfaces utilisateur, une interface web (*cf.* figure 1.15) ainsi qu'une interface en ligne de commande, qui permettent par ailleurs l'interaction avec le robot de collecte (déclenchement d'un « scan » des sites de référence, modification du fichier de configuration). Il est à noter qu'outre des options d'export sous forme de fichiers CSV de la base de données de VulneRobot, l'interface web fournit des options de filtrage des données permettant la visualisation des vulnérabilités chaîne fonctionnelle par chaîne fonctionnelle et composant logiciel par composant logiciel.

7. https://www.cert.ssi.gouv.fr/
8. https://nvd.nist.gov/



Software	Version	Function	ID Vulnerability
Ubuntu	12.04	FP1,FP3	CVE-2007-6746
Ubuntu	12.04	FP1,FP3	CVE-2011-4409
Ubuntu	12.04	FP1,FP3	CVE-2012-0948
Ubuntu	12.04	FP1,FP3	CVE-2012-0949
Ubuntu	12.04	FP1,FP3	CVE-2012-0950
Ubuntu	12.04	FP1,FP3	CVE-2012-0962
Ubuntu	12.04	FP1,FP3	CVE-2012-3404
Ubuntu	12.04	FP1,FP3	CVE-2012-3405
Ubuntu	12.04	FP1,FP3	CVE-2012-3406
Ubuntu	12.04	FP1,FP3	CVE-2012-5624
Ubuntu	12.04	FP1,FP3	CVE-2012-6093
Ubuntu	12.04	FP1,FP3	CVE-2012-6129

FIGURE 1.15 – Interface Web du logiciel de veille VulneRobot [GM17a].

Afin que VulneRobot puisse opérer, le fichier décrivant la configuration logicielle du système mentionné ci-dessus doit se conformer au formalisme suivant [GM17b] :

$$COTS_i\_name; COTS_i\_version; function_j\_name; \ldots; function_k\_name; COTS_i\_cpe$$

Où $function_{j\ldots k}$ sont les fonctions auxquelles le logiciel $COTS_i$ contribue et $COTS_i\_cpe$ est l'identifiant CPE [9] synthétisant les informations ayant trait à l'éditeur, au nom et au numéro de version de ce même logiciel. À titre d'exemple, le fichier de configuration correspondant à un sous-système bureautique de bord s'appuyant sur un ordinateur de bureau opérant le système d'exploitation Debian dans sa version 10, Wget dans sa version 1.7.1 et Adobe Acrobat Reader dans sa version 9.3.4 afin d'exécuter une fonction de téléchargement et de consultation de rapports de mission (dans le cadre d'un partage de données opérationnelles entre plusieurs bâtiments d'une même flotte, par exemple) que nous nommerons *read_reports* pourra donc prendre la forme suivante :

9. *Common Platform Enumeration*, selon le formalisme décrit par le NIST à l'adresse suivante : https://nvd.nist.gov/products/cpe.

```
1 Debian ; 10 ; read_reports ; cpe:/o:debian:debian_linux
    :10.0
2 GNU Wget ; 1.7.1 ; read_reports ; cpe:/a:gnu:wget:1.7.1
3 Adobe Acrobat Reader ; 9.3.4 ; read_reports ; cpe:/a:adobe:
    acrobat_reader:9.3.4
```

L'utilisation de VulneRobot par les organismes en charge de l'opération du processus de gestion de correctifs assure ainsi une part majeure du travail de veille, permettant de collecter tout au long du cycle de vie du système les alertes de sécurité, et donc de documenter les vulnérabilités afférentes, publiées pour les logiciels génériques déployés sur ses composants. Il convient toutefois de noter que VulneRobot ne documente pas exhaustivement, en l'état, les failles découvertes : il incombe toujours à l'utilisateur de consulter le bulletin de sécurité relatif afin d'obtenir une description détaillée et précise de la vulnérabilité et des scénarii d'attaque l'exploitant. Par ailleurs, cet outil et la démarche de veille qu'il appuie traitant exclusivement, par essence, des logiciels pour lesquels les éditeurs assurent une communication des vulnérabilités, il est nécessaire de mener des opérations de veille complémentaires afin d'obtenir une vision exhaustive de la sécurité du système.

ii. Veille des vulnérabilités via des méthodes d'audit

Ces opérations complémentaires, couramment synthétisées en tant que *vulnerability assessment and penetration testing* (VAPT), visent à évaluer la sécurité globale du système et recouvrent différents types d'audits pouvant être menés afin de déceler les vulnérabilités affectant le système, et notamment celles qui sont inhérentes à son architecture et à ses règles d'opération – que la première approche de veille n'est pas en mesure de révéler. Des processus et outils associés aux opérations de VAPT ont été exposés et proposés à de nombreuses reprises : le lecteur pourra utilement se référer aux synthèses des méthodes et outils opérées par [SM14] et [SB18], et à la revue de littérature très complète proposée par [SA16].

Dans le contexte de notre processus, nous pouvons mettre en exergue les trois grandes catégories d'opérations suivantes :

— les opérations d'audit organisationnel : il s'agit d'évaluer la pertinence et la robustesse de la politique de sécurité (et notamment de la PSSI) interne au bâtiment et à son écosystème ;
— les opérations d'audit technique, qui recouvrent essentiellement trois va-

riétés d'analyses : l'audit du code source des logiciels déployés sur les composants du système, l'audit de son architecture technique (infrastructure réseau, etc.), et l'évaluation de la robustesse des mécanismes de défense implémentés (choix, soin de l'implémentation et paramétrage des algorithmes cryptographiques, gestion des droits liés aux fichiers, etc.) ;
— les tests d'intrusion permettant de valider les informations collectées lors des différents audits mentionnés aux deux points précédents, et d'évaluer l'exploitabilté des vulnérabilités en situation réelle.

Combinée à la veille des bulletins de sécurité, l'exécution régulière de ces différents types d'audits permet d'obtenir une connaissance aussi complète que possible des différentes vulnérabilités affectant le système (et notamment de les documenter par la construction de scénarii d'attaque spécifiques au système). Cette connaissance permet de satisfaire la seconde finalité de la tâche 2-1, à savoir l'identification et/ou la conception ad-hoc de contremesures pertinentes.

Identification et/ou conception des contremesures

Une fois une vulnérabilité découverte, trois options s'ouvrent quant à l'identification des contremesures appropriées : soit des contremesures satisfaisantes existent et ont été publiées, soit ces contremesures ne sont pas pertinentes et il paraît intéressant d'en développer pour tempérer la vulnérabilité, soit aucune contremesure n'existe et le développement d'une ou plusieurs d'entre elles est indispensable.

i. Identification des contremesures pré-existantes

Les bulletins de sécurité issus des organismes de veille comportent dans leur très large majorité un champ ayant trait aux conseils, outils et solutions proposés par les éditeurs ou les organismes eux-mêmes. Il s'agit parfois de contremesures non-techniques – à titre d'exemple, la NVD fournit, pour la vulnérabilité CVE-2019-10955 affectant des PLC de Rockwell Automation, les entrées synthétisées en table 1.1 [10]. Sont également fournis par ce biais, lorsqu'ils existent, les correctifs logiciels (les « patchs » fournis par les éditeurs) et leur description, ou des exemples d'exploitation de la vulnérabilité, comme nous le montre la synthèse des ressources de la NVD relative à la vulnérabilité CVE-2020-11451 [11] opérée par la table 1.2.

De ce fait, l'identification des contremesures préexistantes – qu'il s'agisse de

10. https://nvd.nist.gov/vuln/detail/CVE-2019-10955
11. https://nvd.nist.gov/vuln/detail/CVE-2020-11451

Hyperlink	Ressource
https ://ics-cert.us-cert.gov/advisories/ICSA-19-113-01	Third Party Advisory ; US Government Resource
https ://www.securityfocus.com/bid/108049	Third Party Advisory ; VDB Entry

Table 1.1 – Contremesures proposées par la NVD pour la vulnérabilité CVE-2019-10955

Hyperlink	Ressource
http ://packetstormsecurity.com/files/157068/MicroStrategy-Intelligence-Server-And-Web-10.4-XSS-Disclosure-SSRF-Code-Execution.html	Exploit ; Third Party Advisory ; VDB Entry
http ://seclists.org/fulldisclosure/2020/Apr/1	
https ://community.microstrategy.com/s/article/Web-Services-Security-Vulnerability	Patch ; Vendor Advisory
https ://www.redtimmy.com/web-application-hacking/another-ssrf-another-rce-the-microstrategy-case/	Exploit ; Third Party Advisory

Table 1.2 – Contremesures proposées par la NVD pour la vulnérabilité CVE-2019-10955

« patchs logiciels », de mesures organisationnelles et techniques de contournement ou des conseils opérationnels – est par ce biais facilitée.

ii. Conception de contremesures ad-hoc

Il arrive toutefois qu'aucun correctif ne soit proposé (ce qui est naturellement quasiment systématique dans le cadre des vulnérabilités propres au système révélées par les méthodes d'audit) ou que les contremesures existantes paraissent insuffisantes dès leur identification : il peut s'avérer nécessaire, dans ces deux cas, de concevoir des mesures spécifiques permettant d'atténuer les vulnérabilités découvertes. Ces mesures peuvent être temporaires – et déployées dans l'attente de la publication d'un correctif issu de l'éditeur, dans le cadre d'une faille affectant un COTS – ou définitives, techniques ou organisationnelles. Cette conception pouvant être temporellement coûteuse, il est possible de la différer en évaluant sa nécessité au regard des résultats du calcul d'impact de la vulnérabilité, opéré en tâche 2-2.

1.4.1.3 Sortie

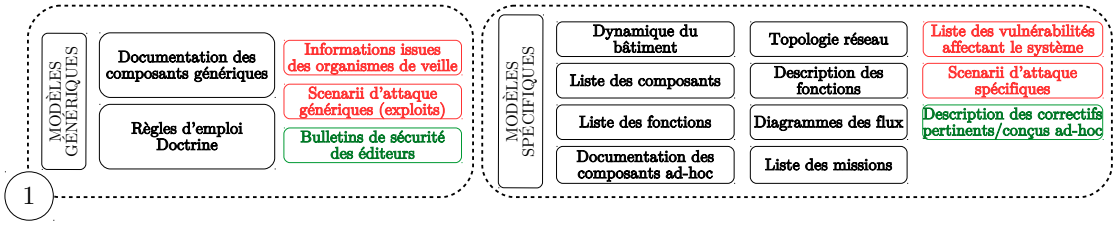


FIGURE 1.16 – Les différentes classes de modèles générées (en rouge et vert sur la figure) durant la tâche 2-1.

Pour chaque vulnérabilité découverte, les opérations exécutées dans le cadre de la tâche 2-1 permettent d'obtenir les informations suivantes :

- la description de la vulnérabilité ;
- les scenarii d'attaque génériques ou spécifiques l'exploitant ;
- le(s) composant(s) impacté(s) ;
- la(les) chaîne(s) fonctionnelle(s) impactée(s) ;
- l'identification de la(des) contremesure(s) préexistante(s) (dans le cas des contremesures proposées par les éditeurs des COTS déployés, cette identification comportera le lien vers les ressources relatives) et/ou proposée(s) ;
- dans le cas des contremesures conçues ad-hoc, leur description et éventuellement les éléments techniques pertinents (code source, etc.).

Une première décision peut éventuellement être prise sur la base de ces éléments en fonction de la criticité des chaînes fonctionnelles impactées par la vulnérabilité, dans l'hypothèse où un grand nombre de vulnérabilités serait successivement remonté par les opérations de veille en un temps réduit et où la réponse devrait être apportée dans un temps tout aussi contenu : par exemple, afin d'accélérer la prise de décision sur les failles probablement critiques, les vulnérabilités découvertes peuvent être hiérarchisées en fonction des chaînes fonctionnelles impactées, et les opérations de calcul d'impact de la tâche 2-2 peuvent ainsi se dérouler selon cet ordre.

1.4.2 Analyse d'impact

La finalité de la tâche 2-2 est, pour une vulnérabilité donnée, de fournir un ensemble de mesures comparables entre elles, modélisant l'impact sur la capacité du système à accomplir ses missions :

— de la vulnérabilité elle-même ;
— d'une série d'attaques l'exploitant ;
— de contremesures la corrigeant ;
— pour chaque contremesure, de la série d'attaques mentionnée précédemment sur le système où elle est déployée.

La comparaison de ces différentes mesures fournit pour chaque correctif deux indications essentielles : d'une part, par différence de l'impact de la vulnérabilité et de celui de la contremesure, on obtient une **mesure de non-régression** du système suite à son déploiement ; d'autre part, la **mesure d'efficacité** de la contremesure est fournie par la différence entre les impacts de la série d'attaque sur le système « vulnérable » et ceux de cette même série sur le système « corrigé » par le déploiement de cette contremesure. Afin de mener ces calculs d'impact, il est préalablement nécessaire d'enrichir le modèle comportemental exécutable issu du corpus ③ pour prendre successivement en compte dans notre abstraction la vulnérabilité, chacune de ses contremesures, et les attaques afférentes.

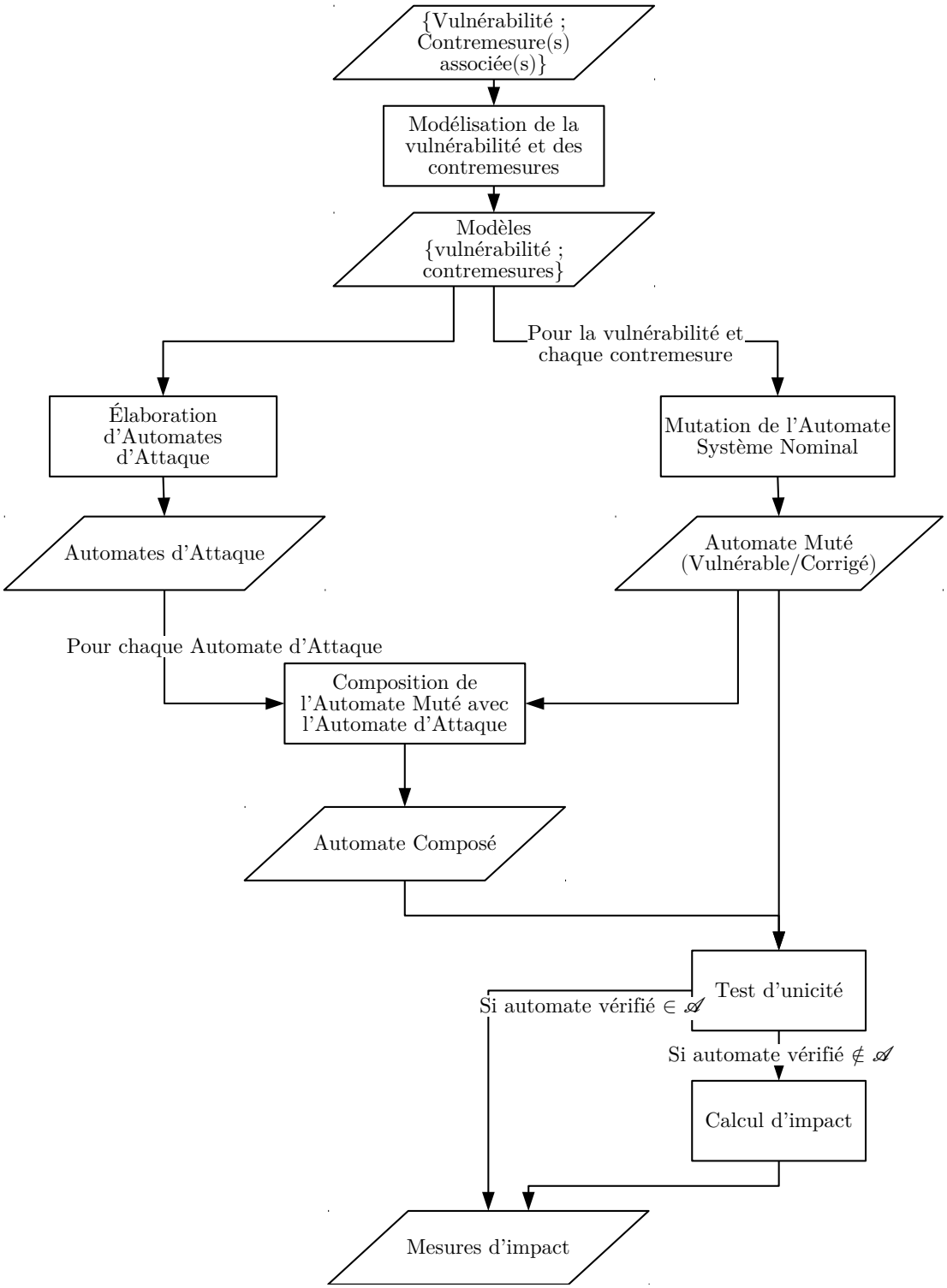


FIGURE 1.17 – Séquence des opérations de la tâche 2-2. $\mathscr{A}$ représente l'ensemble des automates système, qu'ils soient nominaux ou mutés, archivés au fil de leurs compilations (tâche 1-3) et modifications (tâches 2-2 et 2-4) successives tout au long du cycle de vie du système.

1.4.2.1 Pré-requis

— Données d'entrée : l'ensemble de modèles du corpus ③, la liste des vulnérabilité(s) et des contremesures établie à la tâche précédente, l'ensemble des informations collectées lors de cette tâche permettant la modélisation des vulnérabilité(s) et contremesures ;
— Outils : un environnement permettant le *model-checking* des modèles du corpus ③ dans le formalisme choisi, par exemple UPPAAL et ses scripts de vérification associés, un ensemble d'outils de compilation permettant la modification automatisée des modèles du corpus ③, une base de données permettant de conserver l'historique des modèles vérifiés et des mesures résultantes ;
— Savoirs-faire : *model-checking*, ingénierie des systèmes cyber-physiques, connaissance du contexte opérationnel du système, modélisation des systèmes sous forme d'automates finis temporisés.

1.4.2.2 Opérations

Les opérations de la tâche 2-2, dont la séquence est synthétisée en figure 1.17, s'exécutent en trois temps :

— Le temps de la modélisation des informations nouvelles, issues du corpus ① modifié – il s'agit ici de modéliser la ou les vulnérabilités révélées et leurs correctifs afférents, mais également de modéliser les scenarii d'attaque relatifs ;
— Le temps de la modification des modèles exécutables, consistant en la préparation du calcul d'impact par la mutation des modèles du corpus ③ à partir des modèles établis lors du temps précédent ;
— Le temps du calcul d'impact.

i. Modélisation des vulnérabilité(s) et contremesures

La modélisation d'une vulnérabilité et d'une contremesure sont deux procédés similaires. En effet, le but de cette modélisation est de rendre compte des modifications comportementales, effectives ou potentielles, induites sur le système par la présence de la vulnérabilité ou le déploiement de la contremesure. Par conséquent, modéliser l'une ou l'autre revient à décrire de quelle manière l'Automate Système

Nominal du corpus ③ sera affecté ; en d'autres termes à décrire la *mutation*[12] à appliquer à cet automate fini temporisé afin de rendre compte de la présence de la vulnérabilité ou du déploiement de la contremesure. Le formalisme de la description de ces mutations, construites à partir des informations collectées lors de la tâche 2-1, sera exposé au chapitre 2.

ii. Modélisation des séquences d'attaque

La modélisation des séquences d'attaque, exécutée en parallèle de la modélisation des vulnérabilité(s) et contremesures, suit une logique différente : en effet, il s'agit d'établir pour chaque scénario d'attaque un automate fini (exprimé dans le même formalisme que celui du corpus ③) qui simule le comportement d'un attaquant suivant la séquence d'attaque modélisée. Cet *automate d'attaque* sera ensuite composé avec l'automate système muté afin de modéliser les effets comportementaux de l'attaque du système.

Il est construit à partir des informations du corpus ① modifié ; son formalisme sera détaillé au chapitre 2.

iii. Mutation de l'Automate Système Nominal et composition avec les automates d'attaque

Les opérations de mutation et de composition de l'Automate Système Nominal visent à construire, pour chaque vulnérabilité, les automates finis temporisés suivants :

- l'Automate Vulnérable ;
- pour chaque attaque l'exploitant, les Automates Système Vulnérables Sous Attaque ;
- pour chaque contremesure la tempérant, les Automates Système Corrigés ;
- pour chaque attaque l'exploitant et pour chaque contremesure la tempérant, les Automates Système Corrigés Sous Attaque.

Cette série d'automates mutés est construite à partir du corpus ③ et des modèles établis lors du premier temps de la présente tâche. L'Automate Système Vulnérable et les Automates Système Corrigés sont calculés en appliquant les mutations afférentes précédemment décrites ; leurs équivalents sous attaque, ou *automates composés*, en sont dérivés par leur composition avec chacun des automates d'at-

12. On appelle *mutation* d'un automate fini temporisé $\mathcal{A}$ tel que défini au chapitre 2 toute modification de $\mathcal{A}$ résultant en un autre automate fini temporisé dans le même formalisme (se reporter à la définition 6 (chapitre 2, section 2.3.2) pour de plus amples détails).

taque précédemment générés. En plus d'être utilisé immédiatement après dans le cadre des opérations de calcul d'impact, chacun de ces automates est archivé dans une base de données qui s'enrichira par conséquent tout au long du cycle de vie du système.

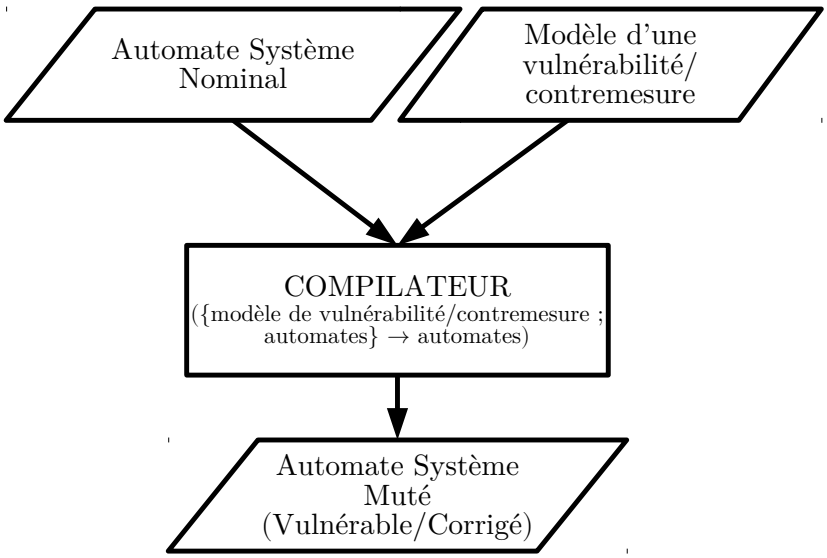


FIGURE 1.18 – Architecture de l'outil de mutation du modèle exécutable

Semblablement aux générations automatisées assistant les opérations de la tâche 1-3, les mutations de l'automate nominal peuvent être outillées par un compilateur dont le principe est exposé en figure 1.18. Il est à noter qu'en l'état, cet outil assure uniquement la « seconde moitié » de la chaîne de compilation, à savoir la génération des modèles mutés à partir d'instructions atomiques (ajouter une transition, modifier une garde, supprimer un état, etc.). L'analyse syntaxique de la description des mutations et sa transcription en un jeu d'instructions atomiques demeure donc à implémenter. Par ailleurs, ce compilateur génère uniquement des automates dans le formalisme UPPAAL [Beh+04] mais peut être aisément adapté à d'autres formalismes de modélisation.

iv. Calcul d'impact

Une fois ces opérations préalables exécutées, advient le dernier temps de la tâche 2-2 dont la visée est d'exprimer l'impact, tel que défini dans l'introduction :

— de la vulnérabilité elle-même ;
— de chaque attaque l'exploitant ;

— de chaque contremesure la tempérant ;
— de chaque attaque l'exploitant sur le système « corrigé » par chacune des contremesures testées.

À ces fins, une série de vérifications de modèles est conduite afin de mesurer l'adhérence des automates mutés et des automates composés, établis lors des opérations décrites au paragraphe iii, aux propriétés du corpus ③. Ces vérifications pouvant être temporellement coûteuses, un test d'appartenance, ou d'unicité, est systématiquement appliqué en amont de chacune d'entre elle afin de vérifier que l'automate vérifié n'est pas égal à un automate précédemment vérifié (gardé en mémoire dans la base de données mentionnée au paragraphe précédent) ; de la même manière, les automates mutés et les automates composés sont tous confrontés entre eux afin d'éliminer les doublons potentiels (voir séquence « Test d'unicité » → {« Calcul d'impact » ; « Mesures d'impact »}, figure 1.17). Si ce test d'unicité est positif, l'automate est vérifié et la mesure d'impact obtenue est renvoyée et enregistrée dans la base de données, associée à l'automate vérifié ; si le test est négatif, la mesure d'impact correspondant à l'automate identique existant dans la base de données est directement renvoyée.

À l'issue de ces vérifications est obtenu l'ensemble de mesures d'impact suivant :

— la mesure d'impact de la vulnérabilité $\mathcal{I}^{\mathsf{v}}$;
— pour chaque contremesure c_i, la mesure d'impact de son déploiement $\mathcal{I}_i^{\mathsf{p}}$;
— pour chaque attaque att_j, sa mesure d'impact sur le système vulnérable $\mathcal{I}_j^{\mathsf{v/att}}$;
— pour chaque attaque att_j et chaque contremesure c_i, la mesure d'impact de l'attaque sur le système corrigé $\mathcal{I}_{i,j}^{\mathsf{p/att}}$.

La mesure de non-régression $\mathcal{R}_i^{\mathsf{p}}$ de la contremesure c_i est obtenue par la comparaison entre $\mathcal{I}^{\mathsf{v}}$ et $\mathcal{I}_i^{\mathsf{p}}$; sa mesure d'efficacité $\mathcal{E}_{i,j}^{\mathsf{p}}$ pour chaque attaque a_j est obtenue par la comparaison entre $\mathcal{I}_{i,j}^{\mathsf{p/att}}$ et $\mathcal{I}_j^{\mathsf{v/att}}$. À des fins de comparaison des contremesures entre elles, une mesure d'efficacité étendue $\mathcal{E}_i^{\mathsf{p}}$ de la contremesure c_i peut être ainsi définie : si on considère $k, k \in \mathbb{N}$ attaques exploitant la vulnérabilité étudiée, $\mathcal{E}_i^{\mathsf{p}} = card(\{\mathcal{I}_{i,j}^{\mathsf{p/att}} | \mathcal{I}_{i,j}^{\mathsf{p/att}} >_{lex} \mathcal{I}_j^{\mathsf{v/att}}\}_{j \in [\![0;k-1]\!]})$ où $>_{lex}$ désigne l'ordre défini sur l'ensemble des mesures d'impact. Cet ordre permettant leur comparaison ainsi que leur formalisme seront définis au chapitre 3 qui s'attachera à approfondir les opérations menées dans le cadre de la tâche 2-2.

1.4.2.3 Sortie

La tâche 2-2 permet de fournir un ensemble d'informations décivises pour la prise de décision. Ces informations sont exprimées sous la forme de mesures d'impact qui expriment, pour chaque vulnérabilité :

- son impact ;
- l'impact de chaque attaque l'exploitant sur le système qu'elle affecte ;
- l'impact du déploiement de chaque contremesure candidate à sa correction ;
- l'impact de chaque attaque l'exploitant sur chaque système « corrigé » par le biais des contremesures mentionnées au point précédent.

Par la comparaison de ces mesures d'impact sont également obtenues la **mesure de non-régression** du système lors du déploiement de chaque contremesure, les **mesures d'efficacité** de chaque contremesure pour chaque attaque, ainsi que sa **mesure d'efficacité étendue**. Ces mesures, rendues immédiatement disponibles pour les besoins de la tâche suivante, sont archivées dans la base de données, associées au modèle dont elles sont issues.

1.4.3 Décision

La tâche de décision a pour but d'aider les décideurs à statuer sur les actions à mener afin de corriger une vulnérabilité affectant le système. Elle s'appuie sur la synthèse des connaissances accumulées lors des tâches précédentes et dont l'expression la plus concise réside dans l'ensemble des mesures d'impact issues de la tâche 2-2, et résulte en une prise de décision qui aura pour conséquence à son issue :

- d'arrêter immédiatement l'itération en cours de la seconde phase du processus de gestion de correctifs (non correction de la vulnérabilité) ;
- de poursuivre ses opérations constituant la tâche 2-4 (déploiement d'une ou plusieurs contremesures) ;
- ou de réexécuter la tâche 2-2 (proposition de nouvelles contremesures).

1.4.3.1 Pré-requis

- Données d'entrée : l'ensemble des mesures d'impact calculées à la tâche précédente ;
- Outils : aucun outil particulier n'est recommandé pour l'exécution de cette

tâche ;
— Savoirs-faire : connaissance du contexte opérationnel du système, ingénierie des systèmes cyber-physiques, MCO/MCS.

1.4.3.2 Opérations

La phase de décision implique différents acteurs qui sont fonction de la structure opérationnelle et de maintenance de l'organisme (compagnie d'armateurs, marine d'État, compagnie de croisière, etc.) armant le bâtiment, mais qui incluent idéalement son équipage, les acteurs de la chaîne opérationnelle ainsi que ceux chargés de ses MCO et MCS. Selon l'urgence à corriger la vulnérabilité découverte (et donc de l'impact des attaques potentielles l'exploitant), la décision pourra essentiellement reposer sur l'équipage du navire, le mieux placé pour évaluer d'après ses objectifs et son contexte opérationnel l'importance des chaînes fonctionnelles impactées par la vulnérabilité, les potentielles attaques l'exploitant et les contremesures candidates à sa correction complète ou à son atténuation.

Cette décision pourra revêtir trois formes :
— le déploiement d'une ou plusieurs contremesures candidates ;
— la non correction de la vulnérabilité avec demande de conception de contremesures ad-hoc et de « rejeu » de la tâche 2-2 sur cette série de contremesures ;
— la non-correction de la vulnérabilité s'il apparaît que son impact, ainsi que ceux des attaques l'exploitant, sont négligeables.

Un exemple d'arbre de décision pouvant servir d'ébauche à la réflexion des acteurs impliqués dans cette tâche est présenté en figure 1.19 ; il est relatif à chaque correctif et s'appuie sur les mesures d'impact afin d'orienter la décision à travers quatre tests :

1. un test d'efficacité de la contremesure $\mathcal{E}^{\mathsf{p}} \overset{?}{>} 0$: si elle n'est pas efficace, elle n'est pas déployée et le test 3 est mené ; si elle est efficace, le test 2 est conduit ;

2. un test de non-régression de la contremesure $\mathcal{I}^{\mathsf{p}} \overset{?}{\leq} \mathcal{I}^{\mathsf{v}}$: si elle n'implique pas de régression, elle est déployée ; si elle implique une régression, elle n'est pas déployée et le test 3 est mené ;

3. un test de criticité de la vulnérabilité basé sur l'étude de sa mesure d'impact $\mathcal{I}^{\mathsf{v}}$: s'il ressort que les chaînes fonctionnelles, les missions ou les

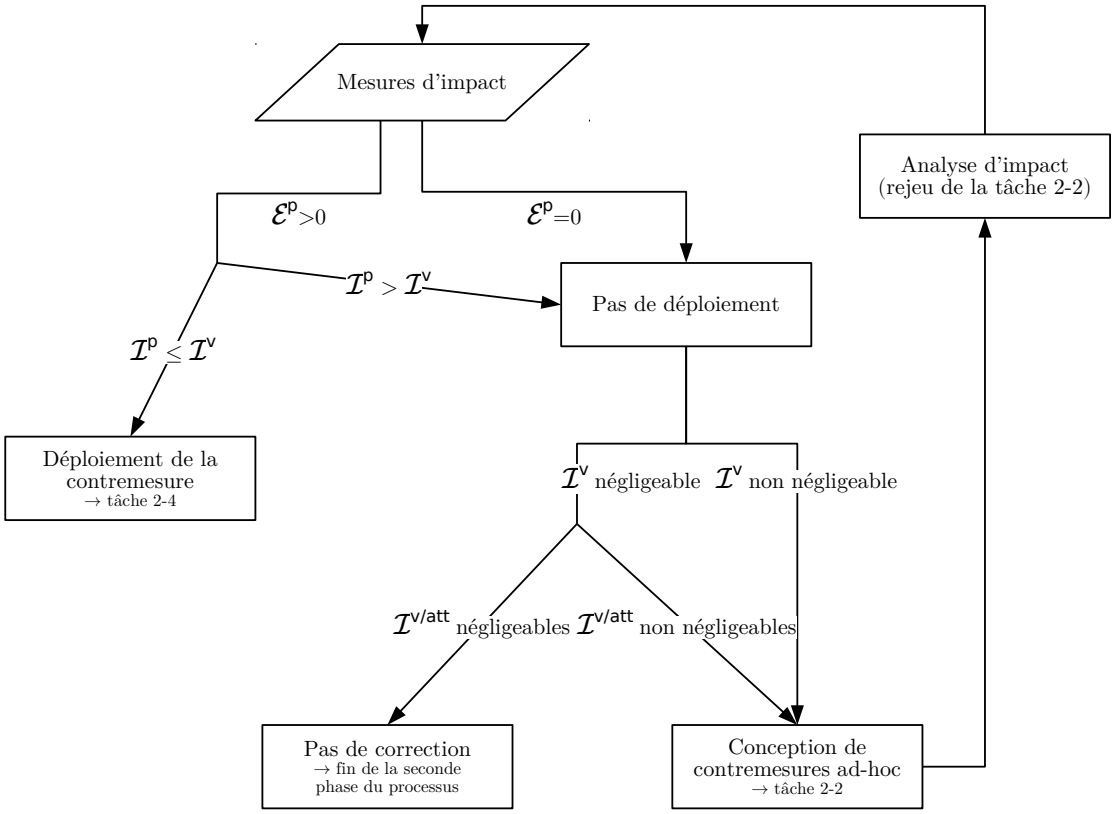


FIGURE 1.19 – Arbre de décision relatif à une contremesure. $\mathcal{E}^{\mathsf{p}}$ représente la mesure d'efficacité étendue de la contremesure, $\mathcal{I}^{\mathsf{p}}$ sa mesure d'impact, $\mathcal{I}^{\mathsf{v}}$ la mesure d'impact de la vulnérabilité corrigée, $\mathcal{I}^{\mathsf{v/att}}$ les mesures d'impact des attaques exploitant cette vulnérabilité.

composants que la vulnérabilité affecte sont critiques aux yeux des acteurs impliqués dans ce processus de choix, décision est prise de concevoir de nouvelles contremesures spécifiques et d'en évaluer les impacts ; sinon, le test 4 est réalisé ;

4. un test de criticité des k attaques exploitant la vulnérabilité, basé sur l'étude de l'ensemble de ses mesures d'impact $\{\mathcal{I}_j^{\mathsf{v/att}}\}_{j\in[\![0;k-1]\!]}\}$ que nous désignons pour des raisons de lisibilité $\mathcal{I}^{\mathsf{v/att}}$ sur l'arbre de décision : s'il ressort de cette étude que les chaînes fonctionnelles, les missions ou les composants qu'elles affectent sont critiques aux yeux des acteurs impliqués dans la présente phase, décision est prise de concevoir de nouvelles contremesures spécifiques et d'en évaluer les impacts ; sinon, décision est

prise de ne pas corriger la vulnérabilité.

Les criticités évaluées lors des tests 3 et 4 sont naturellement pondérées par l'urgence opérationnelle, laissée à la discrétion de l'état-major du bâtiment, qui peut décider de perdre des chaînes fonctionnelles et d'opérer le système en mode dégradé pour l'impératif de réalisation de ses missions.

1.4.3.3 Sortie

À l'issue de la tâche 2-3 sont rendues disponibles :
— la décision de déploiement d'une ou plusieurs contremesure(s) ou la décision de n'opérer aucun déploiement ;
si décision est prise de déployer une ou plusieurs contremesure(s) :
— dans le cas d'un correctif logiciel, son identification et les éventuelles consignes de déploiement afférentes ;
— dans le cas d'une contremesure autre, sa description et les éventuelles consignes de déploiement afférentes.
et, si décision est prise de n'opérer aucun déploiement :
— dans le cas d'un non déploiement avec conception de contremesures adhoc, la description de ces contremesures, nécessaire au rejeu des opérations de la tâche 2-2 afin d'évaluer leur impact ;
— dans le cas d'une non correction de la vulnérabilité, l'Automate Vulnérable. En effet, si la décision retenue est de ne pas corriger la vulnérabilité, un enrichissement des modèles du système est directement opéré à l'issue de cette tâche et le corpus ③ est modifié afin de rendre compte de la présence de cette vulnérabilité.

1.4.4 Déploiement, validation et enrichissement des modèles

Dernière étape du processus de gestion de correctifs proposé, la tâche 2-4 est conçue pour répondre à deux besoins : d'une part au besoin de déploiement des contremesures retenues lors de la tâche de décision, et d'autre part au besoin tout aussi critique d'enrichissment des modèles, permettant de maintenir une connaissance aussi à jour que possible de la configuration du bâtiment.

1.4.4.1 Pré-requis

— Données d'entrée : contremesures à appliquer (leur contenu et leur modélisation) en vertu de la décision de déploiement rendue à la tâche précédente, modèles du corpus ①, modèles exécutables du corpus ③ et modèles mutés (Automates Corrigés) issus de la tâche 2-2 ;
— Outils : aucun outil particulier n'est recommandé pour l'exécution de cette tâche ;
— Savoirs-faire : connaissance du contexte opérationnel du système, ingénierie des systèmes cyber-physiques, ingénierie logicielle, MCO/MCS.

1.4.4.2 Opérations

Les opérations menées dans le cadre de cette tâche peuvent être réparties en deux catégories, comme illustré sur la figure 1.20 : les opérations de déploiement des contremesures choisies, puis de maintien des connaissances (ou de gestion de configuration).

Les opérations de déploiement étant propres à chaque contremesure, il est difficile d'en établir une description générique ; à tout le moins, elles suivront systématiquement les étapes suivantes, menées sur le système et préalablement, si possible, sur un jumeau numérique :

1. le déploiement de la contremesure, où le système est modifié afin d'intégrer la contremesure ;
2. la validation de ce déploiement, où des tests aussi exhaustifs que possible seront menés afin de valider la bonne intégration de la contremesure ; s'il s'avère lors des opérations de validation que surviennent des effets indésirables non décelés par l'analyse d'impact conduite à la tâche 2-2 il peut être nécessaire, en fonction de leur gravité, de mener l'étape 2 bis ;

2 bis. l'amendement éventuel de la contremesure, afin de l'ajuster au mieux au contexte du système et d'effacer ainsi les effets indésirables décelés lors de son intégration.

Il est à noter que dans le cas de contremesures logicielles fournies par des sources extérieures (éditeurs des logiciels vulnérables, etc.) il est essentiel de réaliser des tests d'authentification et d'intégrité du correctif obtenu, afin d'écarter le risque de toute altération accidentelle ou frauduleuse.

Les opérations de maintien des connaissances menées subséquemment af-

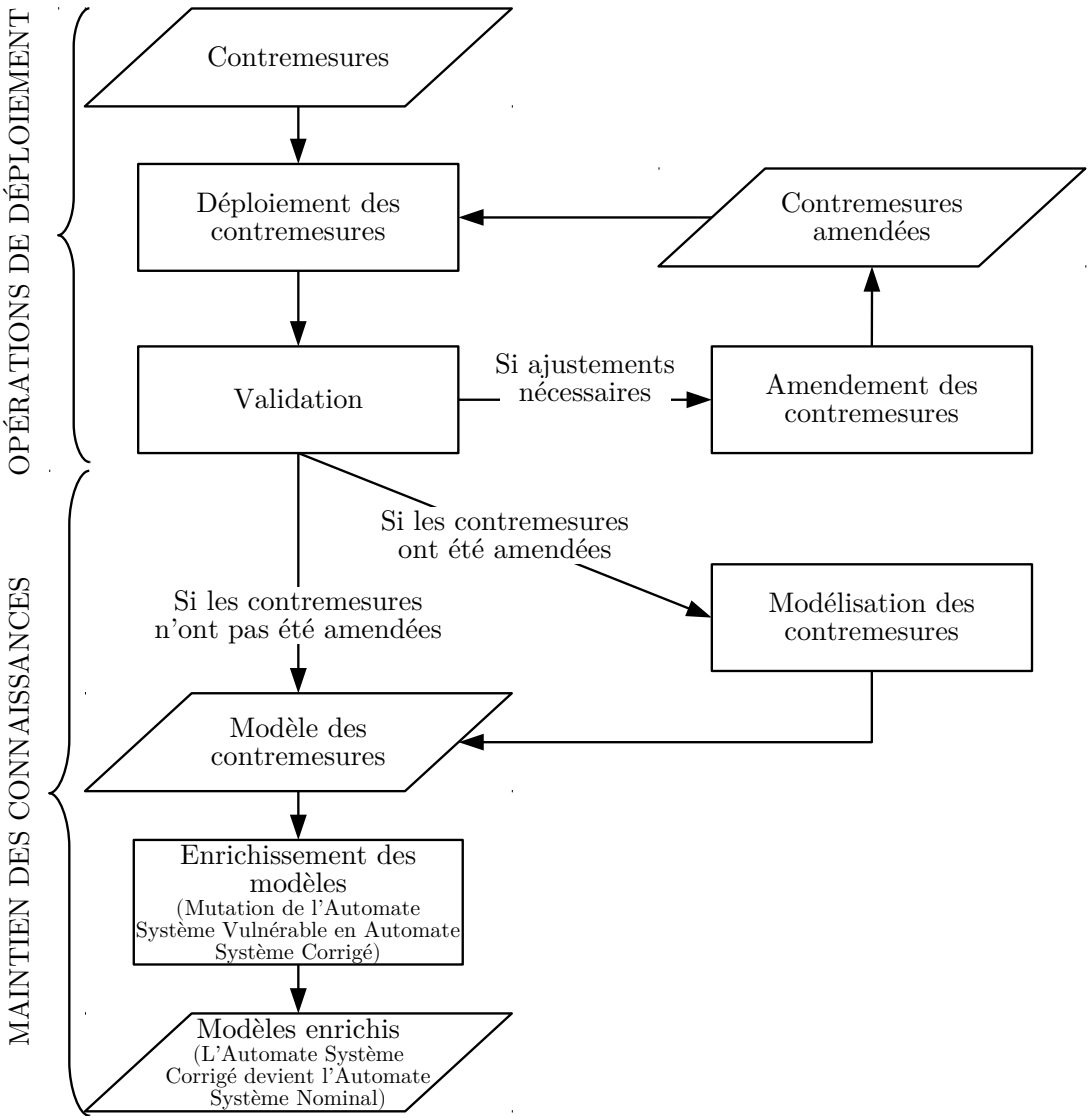


FIGURE 1.20 – Séquence des opérations menées dans le cadre de la tâche 2-4.

fectent deux catégories de modèles : d'une part les documents du corpus ① (et tout autre document à disposition des acteurs du maintien en condition opérationnelle du bâtiment) classiquement modifiés dans le cadre d'actions de gestion de configuration (versions des logiciels déployés, description fonctionnelle, description des composants, etc.), d'autre part les modèles du corpus ③ en ce que l'Automate Système Corrigé établi pour rendre compte du déploiement des contremesures devient l'Automate Système Nominal qui servira de base aux prochaines itérations du processus. Si les contremesures sont déployées sans aucune modification, l'enrichis-

sement de ces modèles est aisé puisqu'il suffit d'appliquer les mutations, obtenues lors de la tâche 2-2, à l'Automate Système Vulnérable établi lors de la même tâche. Si en revanche une contremesure a été amendée, il conviendra de modifier son modèle pour figurer les modifications effectuées avant de muter l'Automate Système Vulnérable.

1.4.4.3 Sortie

À l'issue de cette dernière tâche, sont obtenus :
— le système corrigé ;
— le corpus ① enrichi et tout autre document annexe utile à la gestion de configuration mis à jour ;
— le corpus ③ enrichi.

Ces modification étant apportées, le processus de gestion de correctifs peut reprendre son cours et les modèles qui l'étaient seront prêts à servir de base à une nouvelle itération de sa seconde phase dès qu'une nouvelle vulnérabilité sera découverte.

1.5 Conclusion

Le chapitre qui se clôt nous a permis d'apporter une proposition de réponse méthodologique appelée par la première question de recherche orientant ces travaux de thèse. Structuré par deux phases s'inscrivant dans des durées distinctes – la première fondée sur le temps long du cycle de vie d'un système naval, la seconde sur le temps contenu de la réaction –, le processus dont l'architecture a été proposée est orienté par la nécessité d'évaluer l'efficacité et l'innocuité, ou la non-régression, des contremesures conçues ou collectées dans le cadre d'une réponse à la découverte d'une vulnérabilité. Ces deux paramètres sont évalués grâce à un ensemble de mesures exprimant l'impact des contremesures, de la vulnérabilité découverte et des attaques l'exploitant ; du fait de la spécificité des systèmes pour lesquels ce processus a été conçu, cet impact traduit l'ensemble des retombées d'une contremesure, d'une attaque ou d'une vulnérabilité sur la capacité du bâtiment à accomplir ses missions.

Ces mesures d'impact, permettant d'aider la prise de décision orientant la réponse à la découverte d'une vulnérabilité, sont rendues possibles par une modélisation comportementale poussée du système, une veille minutieuse de son état de

sécurité, et la définition de méthodes de calcul d'impact et de métriques conçues pour les exprimer. Si ce premier chapitre a permis d'exposer les outils répondant au besoin de veille de l'état de sécurité du bâtiment étudié, les suivants s'attacheront à exposer et motiver le formalisme de modélisation retenu, à rappeler ses présupposés, et à approfondir les opérations de la tâche 2-2 dans le but de proposer une démarche de calcul d'impact et un formalisme de mesure répondant au besoin de comparaison des impacts qui, nous avons pu le constater, est fondamental pour la prise de décision constituant la finalité d'un processus de gestion de correctifs.

Chapitre 2

La modélisation d'un système complexe

Table des matières

Le processus de gestion des correctifs exposé au premier chapitre, réponse méthodologique globale à la problématique des travaux de thèse exposés dans le présent mémoire est, rappelons-le, divisé en deux phases – la première d'entre elles aboutissant à la construction d'un modèle comportemental du système naval analysé ainsi qu'à l'élaboration de propriétés de sûreté afférentes. Si ce modèle et ces propriétés sont le principal substrat de la seconde phase du processus de gestion de correctifs dont l'analyse d'impact est la tâche cruciale, celle-ci nécessite d'autres modèles afin d'être menée à son terme : une représentation des vulnérabilités, des attaques relatives et des correctifs associés.

Comme nous avons pu l'évoquer précédemment, la modélisation d'un système complexe est une problématique non triviale [LL96 ; Mad19]. L'objectif du présent chapitre est donc, dans un premier temps, d'exposer et de justifier la méthodologie de modélisation choisie dans le cadre de nos travaux, ses pré-requis en matière de corpus documentaire ainsi que le formalisme retenu pour l'expression du modèle comportemental. Ensuite, nous nous attacherons à présenter la démarche de modélisation des vulnérabilités, attaques et correctifs, en expliquant sa cohérence au regard du modèle issu de la phase de modélisation du système ainsi que de l'objectif d'analyse d'impact au cœur de ces travaux.

2.1 Revue de littérature

2.1.1 Modélisation des grands systèmes cyber-physiques

Si la littérature scientifique est relativement abondante, surtout depuis une dizaine d'années, sur la représentation des grands systèmes cyber-physiques, nous évoquerons dans cet état de l'art les travaux ayant trait à une modélisation exhaustive de ces systèmes (c'est-à-dire prenant en compte sa diversité à tous les niveaux d'abstraction, de ses missions au comportement de ses composants).

Catherine M. Burns *et al.* proposent [Bur+05] une modélisation des frégates de la classe Halifax selon la hiérarchie d'abstraction [Ras85] de Jens Rasmussen. Le métamodèle de Rasmussen est en effet particulièrement approprié à la représentation des grands systèmes cyber-physiques, étant donné qu'il permet la représentation de niveaux d'abstraction hétérogènes et qu'il permet la modélisation des liens fonctionnels entre ces différents niveaux (*cf.* figure 2.1) : in fine, un tel modèle

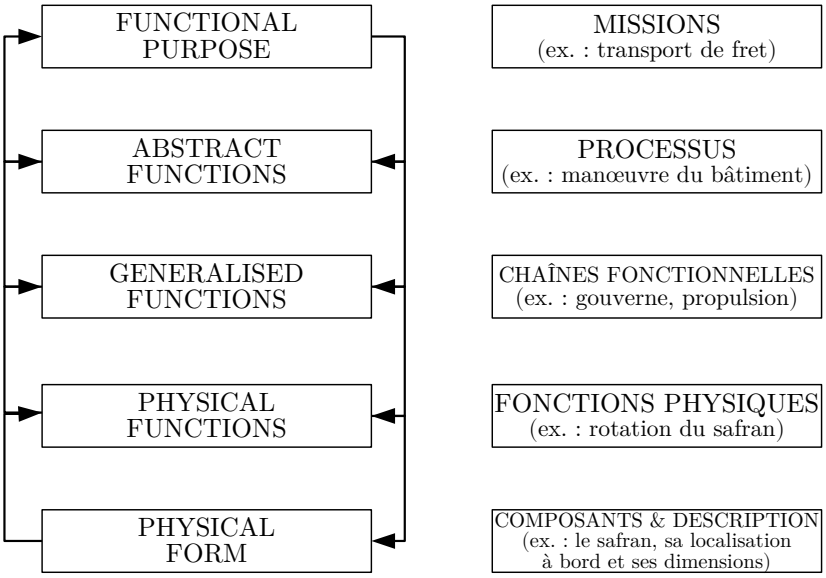


FIGURE 2.1 – Hiérarchie d'abstraction de Rasmussen [Ras85] ; la correspondance entre ses niveaux d'abstraction et ceux d'un système naval complexe est indiquée par la nomenclature et les exemples de la colonne de droite.

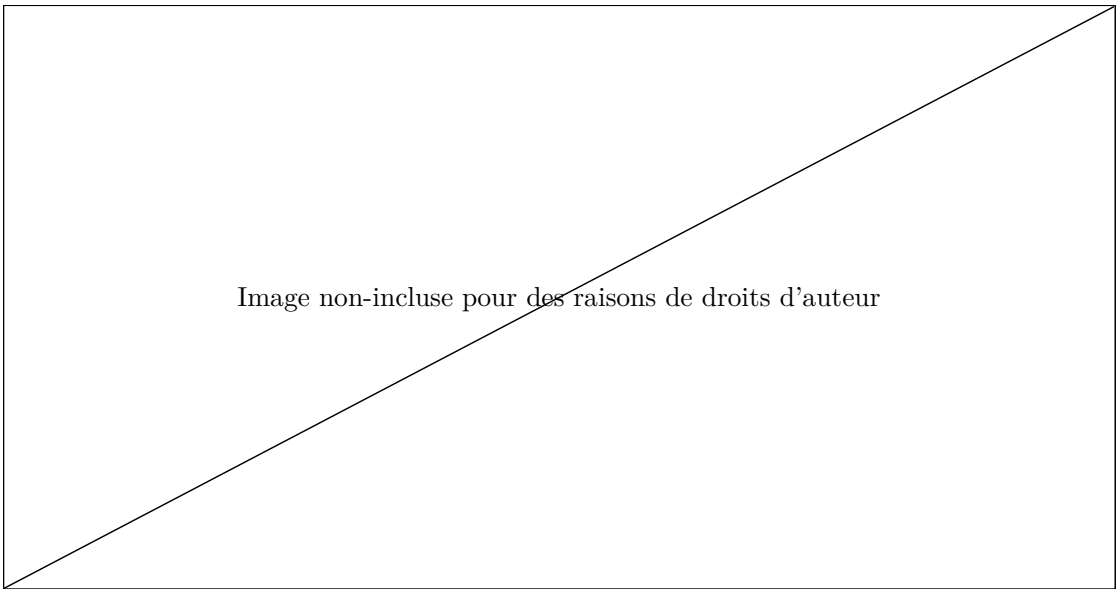


FIGURE 2.2 – Application du modèle de Rasmussen à la classe Halifax, telle que présentée dans [Bur+05].

lie les missions aux composants à travers des fonctions d'abstraction décroissante,

faisant ainsi émerger le comportement du système représenté. [Bur+05] valide donc l'utilisation de la hiérarchie d'asbtraction à des fins de modélisation d'un système naval complexe (*cf.* figure 2.2) ; similairement, Ann Bisantz *et al.* [Bis+01] l'avaient montré en appliquant le modèle de Rasmussen à un bâtiment de surface de l'US Navy. Les deux approches sont toutefois différentes puisque [Bur+05] vise à analyser la classe Halifax et son environnement opérationnel alors que les bâtiments existent et opèrent, tandis que [Bis+01] établit une modélisation lors de la conception du système, afin de faciliter celle-ci. Ces approches sont tout-à-fait pertinentes et efficaces pour répondre à l'objectif de modélisation de niveaux d'abstraction hétérogènes – problème inévitablement posé par l'étude de systèmes de l'ampleur des systèmes navals modernes –, d'expression de la dépendance missions/composants et de structuration des informations collectées à des fins de modélisation. Toutefois, elles ne permettent pas la modélisation dynamique nécessaire à notre processus (en effet, une contremesure peut être déployée à une étape donnée d'une mission, une vulnérabilité ne peut devenir exploitable que lors de l'exécution d'une fonction donnée, et une attaque est en elle-même nécessairement dynamique).

Gabriel Jakobson [Jak11b ; Jak11a ; Jak13] propose, dans une optique d'analyse d'impact des cyberattaques sur un système cyber-physique, une modélisation décrivant les systèmes sous la forme de *cyber-terrains*, extension du modèle proposé dans [Hol+08] et regroupant trois modèles, appelés *sub-terrains* et respectivement relatifs aux composants logiciels, aux composants matériels et aux services. Ces *cyber & sub-terrains* sont des graphes composés de trois types de sommets (*software*, *hardware* et *services*, selon le *sub-terrain* concerné) ; chaque arête représente la dépendance (lien topologique, contenance, etc.) entre ces sommets, et ces graphes sont des structures dynamiques en ce que « les composants et leurs interdépendances sont fonction du temps. » Une modélisation des missions du système sous la forme d'un enchaînement d'étapes élémentaires est également proposée, ainsi qu'un graphe de dépendances permettant d'illustrer les dépendances fonctionnelles unissant les missions aux services et aux composants logiciels et matériels. Ces travaux proposent également une modélisation des attaquants et des cyberattaques ; toutefois, du fait de la nature de cette modélisation qui traduit essentiellement la capacité d'une attaque à déteriorer les capacités opérationnelles du composant qu'elle cible et du niveau de granularité offert par les *cyber-terrains*, il paraît difficile dans ce formalisme de rendre compte de modifications apportées par une attaque sur le système – telles que l'émergence de nouvelles fonctionnalités par l'installation d'un programme malveillant, une modification de la topologie réseau, etc. Par ailleurs, la modélisation des services rend probablement difficile

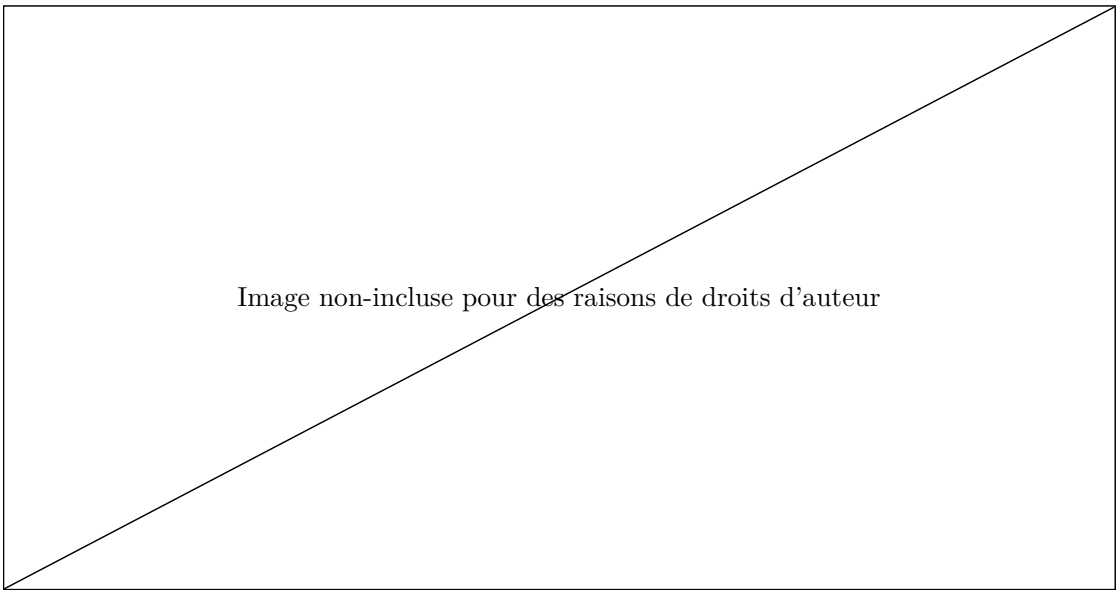


FIGURE 2.3 – Exemple de graphe de dépendances proposé dans [Jak11b] et [Jak11a]. ©IEEE 2011

la modélisation de l'évolution des grandeurs physiques liées au système, pourtant nécessaire au calcul d'impact de certaines attaques (modification malicieuse des consignes de la chaîne de gouverne menant à un changement de cap, telle que mentionnée dans [Pau+17] pouvant mener le bâtiment à dévier de sa route ou à s'échouer, comme l'a montré l'incident du 13 juillet 2017 où le roulier Siem Cicero a perdu sa capacité de macœuvre du fait d'une défaillance logicielle [New17]). Ces modèles ne permettent pas non plus la modélisation d'une vulnérabilité ni une abstraction suffisamment détaillée d'une contremesure pour servir de base à notre processus ; nous pouvons toutefois souligner leur dynamisme et leur pertinence pour représenter une mission et ses dépendances fonctionnelles vis-à-vis des composants, des logiciels et des services proposés par le système modélisé.

Shankara Narayanan Krishna [KT15], Tuo Mingfu *et al.* [Tuo+16] et Wang Qiang *et al.* [Wan+13] proposent une modélisation permettant de saisir plus finement le comportement des systèmes par l'utilisation d'automates hybrides dans une visée de vérification formelle des propriétés de sûreté des systèmes représentés. Ces modèles permettent à la fois une modélisation dynamique par l'utilisation d'automates finis et une représentation des grandeurs physiques associées au système par le biais d'un ensemble d'équations différentielles ; ils répondent efficacement à cet égard à notre besoin de modélisation, mais à l'exception de [KT15] exposant

une démarche de composition de réseaux d'automates hybrides, ils ne mettent pas en œuvre une modélisation hiérarchique permettant la lisibilité du modèle en différentes strates (telles que celles proposées par Rasmussen, par exemple) – ce qui complexifie le processus de modélisation du système et rend moins aisée l'utilisation du modèle qui en est issu par les utilisateurs. Par ailleurs, aucun de ces travaux ne traite de la modélisation des missions associées au système représenté. Il n'en demeure pas moins que les automates hybrides sous-tendant ces trois méthodologies convergentes sont particulièrement adaptés à notre besoin comme nous l'exposerons ultérieurement.

Il ressort de l'étude de ces trois approches plusieurs éléments permettant d'esquisser les contours de notre modélisation qui s'attachera à les faire converger au sein d'un même formalisme. En effet, une réponse à notre seconde question de recherche, enrichie des besoins exposés au cours du chapitre précédent, peut être apportée par l'établissement d'un modèle permettant l'association d'une modélisation comportementale dynamique du système via l'utilisation d'automates finis, d'une stratification de ces automates selon une hiérarchie inspirée du modèle de Rasmussen, et d'une représentation séquentielle des missions et de leurs dépendances fonctionnelles telle qu'elle peut figurer dans [Jak11b ; Jak11a ; Jak13].

Du choix de cette modélisation découlent des contraintes permettant de limiter le champ de représentation des vulnérabilités, attaques et contremesures affectant les systèmes étudiés.

2.1.2 Modélisation des vulnérabilités, des attaques et des contremesures

La littérature en la matière est une fois de plus relativement abondante [GG+17 ; KC13 ; Che+03 ; Sam+13] ; toutefois, du fait de notre parti-pris de modélisation, nous avons choisi de transcrire ces trois types d'éléments dans nos modèles à travers leur altération. L'altération des modèles comportementaux des systèmes cyber-physiques a déjà été explorée par plusieurs travaux traitant essentiellement de sûreté de fonctionnement. À cet égard, nous pouvons notamment citer les recherches de Bernhard Aichernig *et al.* [Aic+14] qui proposent une démarche d'explication de défaillance lors de tests en boîte noire d'un système donné, basée sur la mutation des automates finis temporisés le représentant. Le but de cette méthode est de déterminer un ensemble de modèles « mutants » faisant état d'une implémentation incorrecte du système testé : étant donné un modèle nomi-

nal de ce système, une implémentation défaillante et un protocole de test auquel échoue cette implémentation, leur approche consiste à générer tous les « mutants » possibles à partir du modèle nominal, et ensuite à exécuter plusieurs étapes de filtrage dans le but d'éliminer les mutants qui ne montrent aucun comportement non-conforme en vertu du protocole de tests. Demeure au final un sous-ensemble de « mutants » correspondant aux modèles possibles du système testé défaillant, permettant donc de comprendre où l'erreur d'implémentation a pu survenir. Cet article définit également des opérateurs de mutation (changement d'état initial, changement d'état final, changement de garde, négation de garde, etc.) appliqués un-à-un pour créer leur ensemble exhaustif de « mutants ».

Dans un esprit proche, nous nous attacherons dans ce qui va suivre à proposer une démarche de modélisation des vulnérabilités, attaques et correctifs basés sur l'application de mutations aux modèles du système étudié. Notre approche diffère toutefois de cele de Bernhard Aichernig en ce que les opérateurs de mutation que nous définirons par la suite sont différents pour plusieurs d'entre eux et que nous en appliquons plusieurs en parallèle ; et surtout, la démarche que nous proposerons génère les « mutants » à partir de la description d'un événement donné plutôt que de générer tous les « mutants » possibles dans le but de vérifier ensuite ceux qui se conforment à cette description.

2.2 Modéliser le système et ses missions

Notre postulat est qu'une solution efficace pour associer dynamisme du modèle et granularité modulaire réside en l'utilisation d'une modélisation comportementale du système et d'une abstraction de ses missions basées sur un réseau d'automates finis temporisés. Comme nous l'avons évoqué dans le développement du chapitre 1, cette modélisation est établie sur la base d'informations issues d'opérations de collecte et d'analyse de modèles préexistants, éventuellement complétés par des raffinements apportés par des entretiens avec divers acteurs impliqués dans la « vie » du navire étudié.

Nous nous attacherons dans cette section à justifier le choix de ce formalisme, ainsi qu'à exposer ses adaptations, extensions et partis pris de modélisation que nous avons retenus dans le contexte de ces travaux [1].

1. En ce qui concerne les aspects méthodologiques régissant l'établissement de la modélisation d'un système selon le formalisme présenté dans ce chapitre, le lecteur pourra utilement se

2.2.1 Des automates finis temporisés pour la modélisation de missions, processus, composants et liens topologiques

Définition 5 (**Automate fini temporisé** (issue de [Beh+04])). *Un automate fini temporisé est un 6-uplet* $(\mathcal{L}, l_0, \mathcal{C}, \mathcal{A}, \mathcal{E}, \mathcal{I})$ *où* $\mathcal{L}$ *est un ensemble d'états,* $l_0 \in \mathcal{L}$ *est l'état initial,* $\mathcal{C}$ *est un ensemble d'horloges,* $\mathcal{A}$ *est un alphabet – ou ensemble d'actions, de co-actions et d'actions internes aux états –,* $\mathcal{E} \subseteq \mathcal{L} \times \mathcal{A} \times \mathcal{B}(\mathcal{C}) \times 2^{\mathcal{C}} \times \mathcal{L}$ *est un ensemble de transitions entre états possédant une action, une garde et un ensemble d'horloges à réinitialiser, et* $\mathcal{I} : \mathcal{L} \to \mathcal{B}(\mathcal{C})$ *associe des invariants aux états,* $\mathcal{B}(\mathcal{C})$ *étant l'ensemble des expressions de la forme* $x \bowtie y$ *ou* $x - y \bowtie c$ *où* $\bowtie \in \{<, \leq, =, \geq, >\}$, $(x, y) \in \mathcal{C}^2$ *et* $c \in \mathbb{N}$.

Les automates finis temporisés sont particulièrement adaptés à la modélisation comportementale des composants des systèmes critiques [And15]. En effet, leurs états décrivent les différentes phases de fonctionnement du composant au cours de son opération, son alphabet représente les actions effectuées par le composant, et ses horloges figurent les contraintes de temps réel. Par exemple, un PLC contrôlant le gouvernail d'un navire pourra être ainsi représenté :

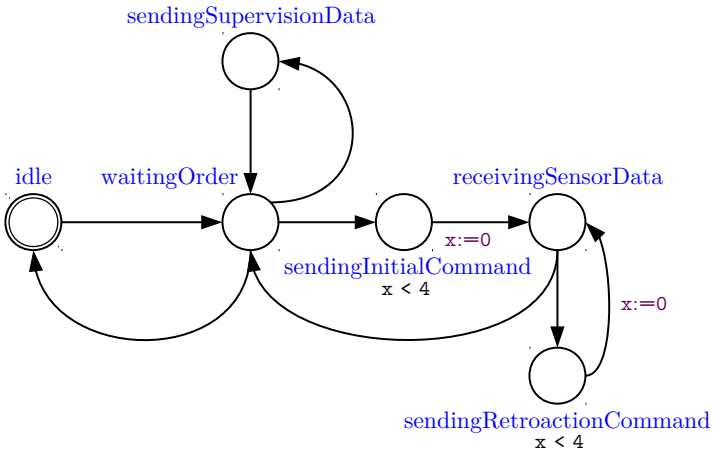


FIGURE 2.4 – Automate modélisant un PLC contrôlant le gouvernail d'un bâtiment.

Une représentation simplifiée de son fonctionnement peut en effet être [2] de considérer six états dont un état initial (*idle*) où le composant est éteint (ou dans

référer au chapitre 1, notamment à la section 1.3.

2. *cf.* annexe A, section A.2

une configuration ne lui permettant pas d'effectuer les tâches de commande de l'actionneur), un état d'attente des ordres de l'opérateur (*waitingOrder*), un état d'envoi à l'actionneur de la commande initiale issue de la consigne reçue de l'opérateur (*sendingInitialCommand*), deux états modélisant la boucle de rétroaction (*receivingSensorData* et *sendingRetroactionCommand*), et un état où le PLC rend compte à l'opérateur des données du composant supervisé (*sendingSupervisionData*).

Par ailleurs, les missions et les processus sont également modélisables par le biais d'automates finis temporisés du fait de leur nature séquentielle ; leurs étapes, nœuds des graphes proposés par Gabriel Jakobson [Jak11b ; Jak11a ; Jak13], sont figurées par des états, et les actions permettant de passer de l'une à l'autre sont représentées par les actions des transitions liant ces états entre eux (*cf.* figure 2.5).

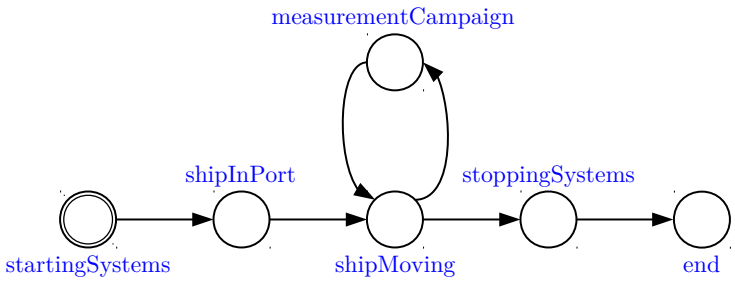


FIGURE 2.5 – Automate modélisant une mission d'un bâtiment océanographique articulée autour d'une campagne de mesures [Sul+18].

Ce concept d'automate fini temporisé peut être étendu afin de correspondre plus finement à notre besoin. UPPAAL opère une telle extension en y ajoutant notamment [Beh+04] :

— des variables et des constantes entières, pouvant être manipulées par des mises à jour au sein des transitions – ces mises à jour pouvant entre autres revêtir la forme d'appels de fonctions programmées et déclarées au sein du modèle UPPAAL – et être évaluées par les gardes des transitions ;
— des synchronisations directionnelles, permettant de synchroniser deux transitions de deux automates distincts entre elles : une transition associée à une action « c ! » se synchronisera avec une transition associée à une co-action « c ? » ;
— des synchronisations en *broadcast*, permettant de synchroniser n transi-

tions de n automates distincts avec une transition d'un $n+1$-*ième* automate.

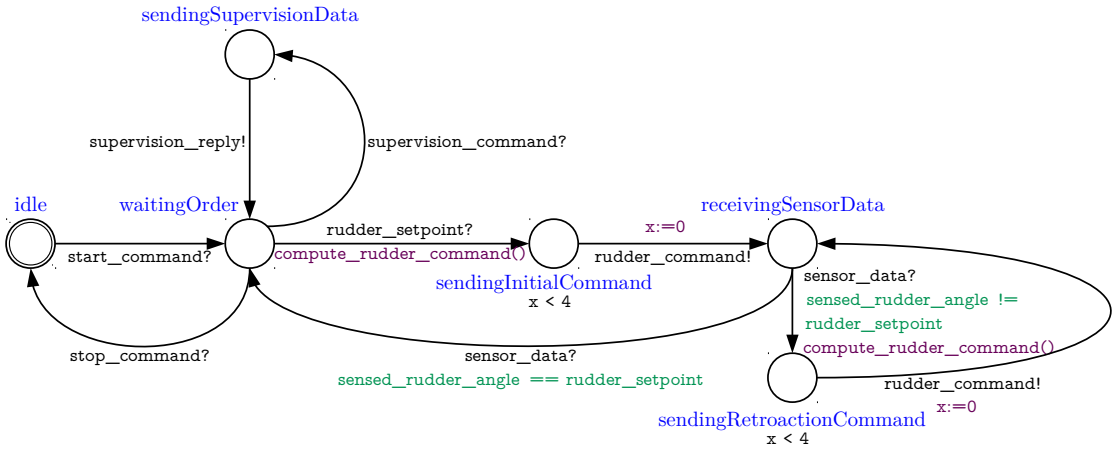


FIGURE 2.6 – Automate modélisant un PLC contrôlant le gouvernail d'un bâtiment. Les synchronisations et les invariants apparaissent en noir, les gardes en vert et les actions de mise à jour des variables en violet.

Grâce à cette extension, nous pouvons enrichir la modélisation du PLC représenté ci-dessus (*cf.* figure 2.4) en ajoutant des gardes et des actions ayant trait aux grandeurs physiques manipulées lors de l'opération d'une chaîne de gouverne (*cf.* figure 2.6). Par ailleurs, les synchronisations apportent de la lisibilité au réseau auquel cet automate s'intègre – en permettant notamment d'expliciter les messages échangés entre les différents composants modélisés, les actions opérées par les utilisateurs du système dans le cadre de ses fonctions, etc. Ces synchronisations sont une pierre angulaire de la modélisation d'un système complexe, étant donné qu'elles permettent la « communication » de différents automates entre eux dans le cadre de réseaux d'automates. Nous pouvons à cet égard aisément modéliser l'architecture réseau du système à travers les liens topologiques existant entre les différents composants, que nous traduisons par autant d'*automates topologiques* servant d'intermédiaires aux synchronisations figurant la communication réseau entre deux *automates composant*, comme l'illustre la figure 2.7.

Notons finalement que le formalisme des des automates finis temporisés présente l'avantage d'être explicite, largement enseigné et éprouvé, et compréhensible par des ingénieurs de différentes cultures techniques pouvant être impliqués dans la conception des systèmes et la réalisation du processus proposé au chapitre 1 (automaticiens, informaticiens, ...).

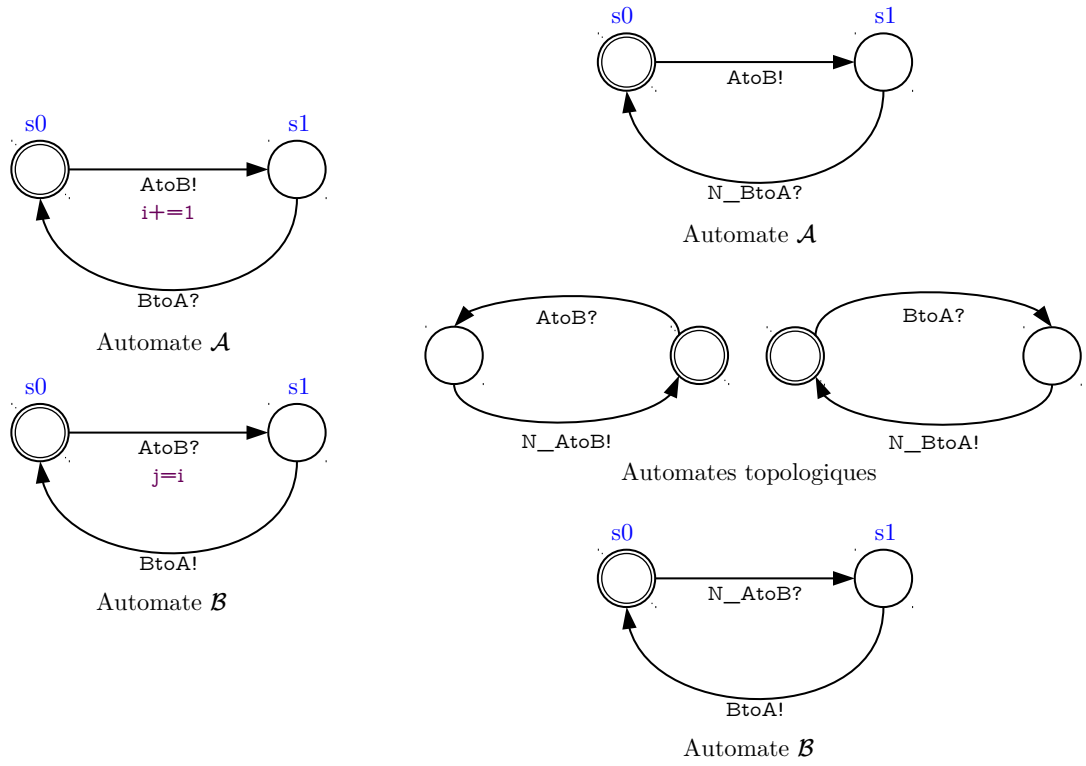


FIGURE 2.7 – Modélisation d'un lien réseau entre deux composants (modélisés par les automates $\mathcal{A}$ et $\mathcal{B}$ utilisant ce lien réseau pour échanger une variable i – transmission de $\mathcal{A}$ à $\mathcal{B}$ – et accuser réception de chaque message – transmission de $\mathcal{B}$ à $\mathcal{A}$ –) par l'ajout d'automates topologiques.

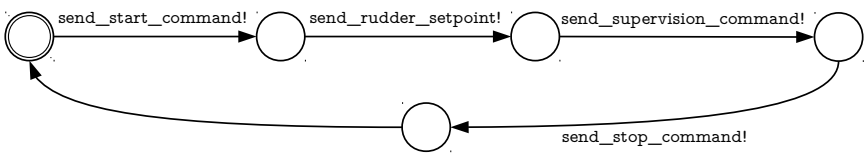
2.2.2 Un réseau d'automates pour la modélisation d'un système complexe

Plusieurs automates finis temporisés peuvent être composés en un *réseau d'automates finis temporisés* [Beh+04 ; KT15] ; ils partagent alors leurs ensembles d'actions et d'horloges, ainsi que de variables, constantes et fonctions en vertu de l'extension proposée par le formalisme UPPAAL. Cette composition d'automates est notablement utile dans une optique de modélisation d'un grand système cyber-physique, puisqu'elle permet de manière intuitive de conserver la structure « réelle » du système représenté : à un composant correspond un automate, et la composition de ces automates figure le système. Un avantage connexe est la lisibilité conférée au modèle ; en effet, sa manipulation est bien plus aisée du fait de cet homologie structurelle avec le système puisque chaque composant est représenté

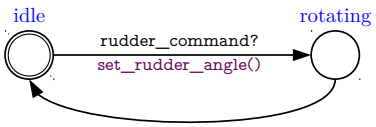
par un modèle unique et distinct – chaque composant, mais également chaque processus, chaque mission ou chaque lien topologique : un avantage supplémentaire d'un tel réseau d'automates réside en la hiérarchisation des automates entre eux.

En effet, comme l'illustre l'exemple de composition fourni par la figure 2.8, les transitions existant entre les différents automates permettent aisément la restitution de niveaux d'abstraction croissants, des composants portant les fonctions physiques aux missions : les automates figurant les composants « physiques » (par exemple un actionneur) se synchronisent avec ceux modélisant les composants « logiques » (par exemple un PLC), se synchronisant quant à eux avec les modèles des processus ou d'autres composants « logiques » plus abstraits (par exemple une interface utilisateur) ; et par la synchronisation des modèles des processus avec les automates abstrayant les missions, l'ensemble des niveaux d'abstraction du système sont liés entre eux (*cf.* figure 2.10, synthétisant ces différents niveaux retenus dans le cadre de notre modélisation). Nous noterons que les chaînes fonctionnelles ne sont pas représentées par un automate particulier : en effet, leur comportement émerge de la composition entre les automates « processus » et les automates « composant » impliqués dans chaque fonction représentée ; la trace d'exécution de ces automates composés permet de représenter la chaîne fonctionnelle, sous une forme bien adaptée une fois encore à sa compréhension par des spécialistes de cultures techniques différentes (voir à titre d'exemple le diagramme de séquence de la fonction de gouverne – figure 2.9 – permettant de retrouver rigoureusement les flux de cette fonction et leur enchaînement temporel tels que décrits dans l'exemple de modèle d'architecture système proposé à l'annexe A, section A.1).

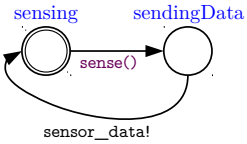
La composition des automates « processus », « composant » et « topologiques » résulte en un réseau d'automates que nous appelons l'Automate Système ; sa composition avec chaque Automate Mission résulte en un réseau permettant d'étudier le comportement du système pendant chacune des missions représentées. Ainsi, si les automates sont adaptés à l'abstraction du comportement des composants, les réseaux d'automates en sont le pendant pour la modélisation du comportement des systèmes.



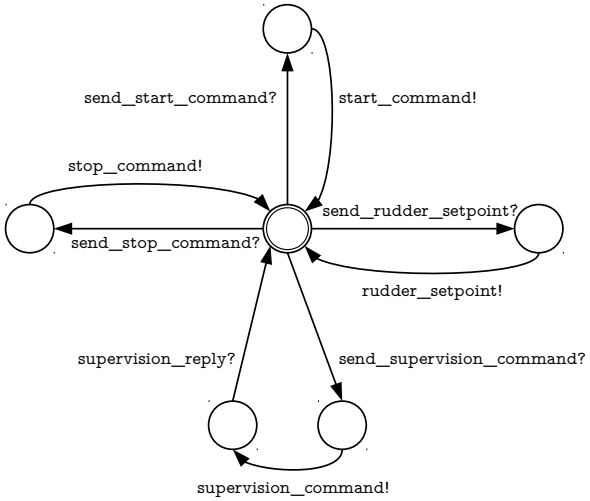
(a) Processus de gouverne



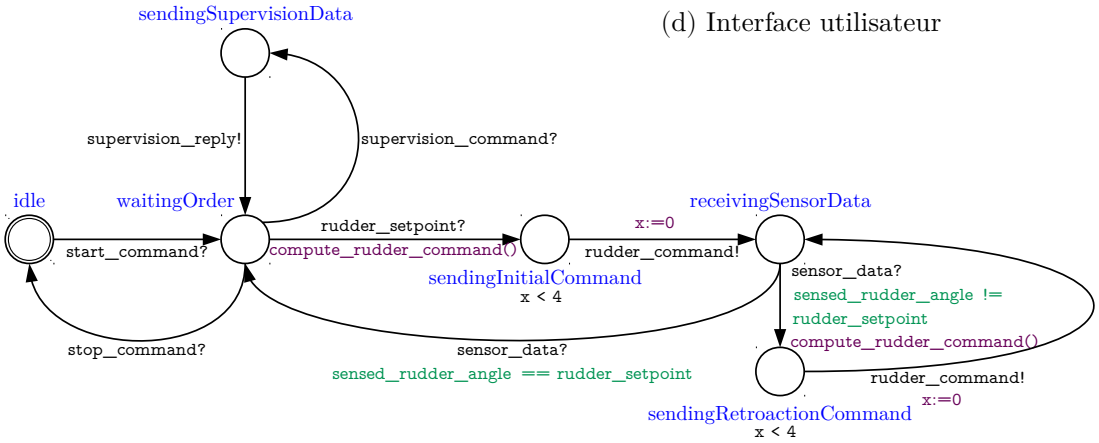
(b) Actionneur du safran



(c) Capteur d'angle de barre



(d) Interface utilisateur



(e) PLC

FIGURE 2.8 – Réseau d'automates modélisant un sous-système de gouverne – selon l'exemple proposé à l'annexe A – et un processus élémentaire de gouverne associé.

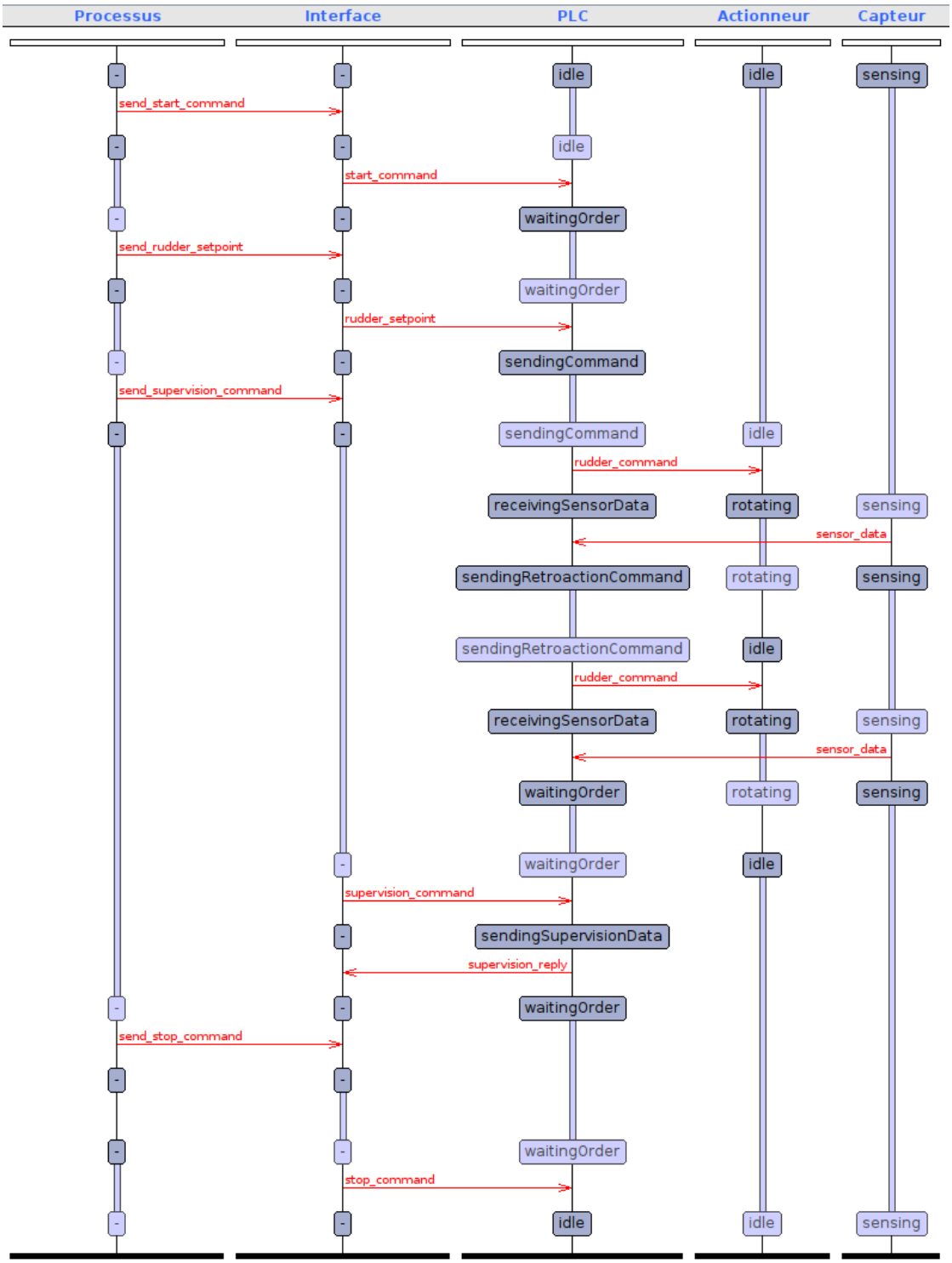


FIGURE 2.9 – Diagramme de séquence généré par UPPAAL 4.1.24 d'une itération du processus de gouverne sur le réseau d'automates représenté en figure 2.8.

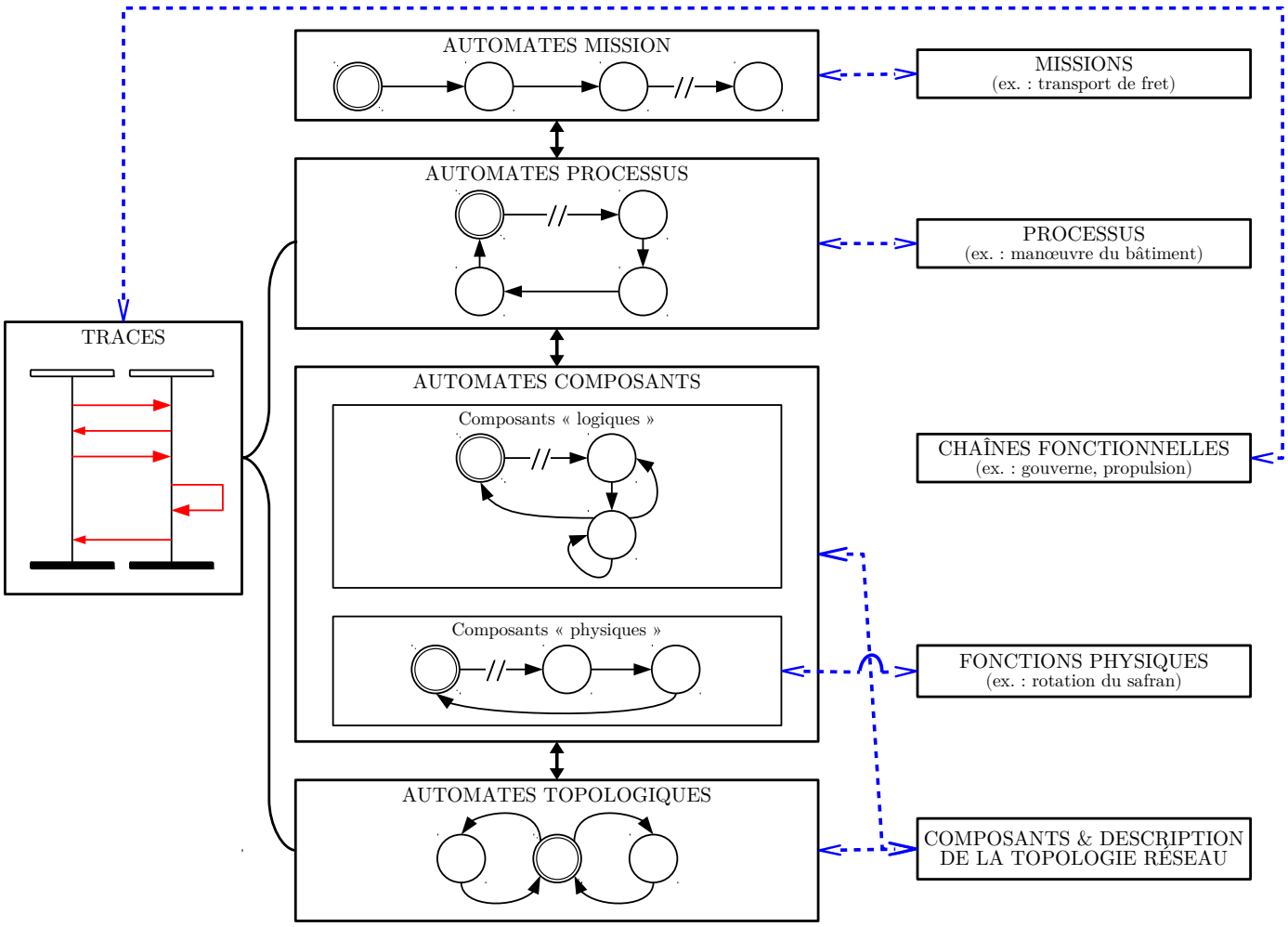


FIGURE 2.10 – L'Automate Système et les Automates Missions : un réseau d'automates hiérarchiques.

Ainsi, un atout supplémentaire d'une modélisation basée sur les automates finis est cette granularité modulaire permettant de modéliser **via le même formalisme** le comportement d'un composant atomique – du simple actionneur mécanique au PLC, plus complexe, disposant de fonctions lui permettant de « faire des choix » en autonomie – jusqu'au déroulement d'une mission en passant par les processus du système, ses chaînes fonctionnelles et sa topologie réseau. Nous disposons de ce fait d'une modélisation complète, qui permet de restituer totalement ou partiellement [3] les informations collectées lors de la première phase du

3. La restitution est par exemple totale en ce qui concerne la topologie réseau – aucune

processus proposé au chapitre précédent.

2.2.3 Intégration des grandeurs physiques et automates hybrides pour la modélisation d'un système dynamique

Parmi ces informations figurent les lois de commande liées aux actionneurs et PLC, et les grandeurs physiques mesurées par les capteurs et nécessaires à leur exécution. Étant donné que les valeurs de ces mesures et de ces consignes peuvent être la finalité – ou un indicateur – d'une cyberattaque, il nous semble essentiel de les intégrer au sein de notre formalisme de modélisation comportementale. Nous nous intéressons par conséquent aux grandeurs descriptives de la dynamique du système et sur lesquelles les opérateurs du système peuvent œvrer pour amender sa marche. Parmi elles, nous pouvons notamment citer ces grandeurs, liées au système dans son ensemble :

- les coordonnées géographiques du bâtiment ;
- sa vitesse ;
- son cap ;
- sa vitesse angulaire autour de son axe de lacet ;
- ses angles de roulis et de tangage (l'angle de lacet étant l'angle de cap) ;

Et, par ailleurs, les grandeurs liées aux actionneurs :

- son angle de barre ;
- la vitesse de rotation de ses moteurs ;
- le rapport de réduction, ou *rapport d'engrènement* de son boîtier inverseur/réducteur.

Nous nous attacherons dans ce qui va suivre à exposer deux approches d'intégration de ces grandeurs physiques, à travers une démarche simple et une démarche plus complexe mais plus proche de la réalité du système.

2.2.4 Une approche de modélisation simple

Si l'abstraction des grandeurs en elles-mêmes revêt ici la même forme (celle d'une variable numérique) qu'elles soient *modifiées* ou *mesurées* par le système, nous pouvons distinguer deux méthodes de représentation de leur « élaboration » en fonction de ces cas de figure.

---

perte d'information ne survient – mais partielle relativement aux informations collectées sur les composants – seul leur comportement et leurs interactions sont représentés.

2.2.4.1 Modélisation des mesures

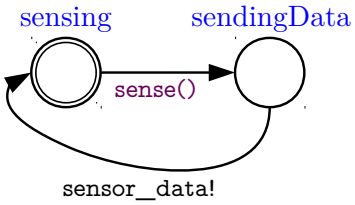


FIGURE 2.11 – Automate représentant le comportement d'un capteur.

La modélisation de la mesure d'une grandeur s'effectue par le biais de déclarations de variables la modélisant (une pour la valeur « vraie » du mesurande, une pour la valeur de la mesure effectuée), d'un automate abstrayant le comportement du capteur afférent (*cf.* 2.11), et d'une fonction de mesure représentant les caractéristiques métrologiques du capteur.

Par exemple, ayant déclaré en préambule du modèle UPPAAL les variables suivantes :

```
1 int rudder_angle ; //valeur vraie du mesurande
2 int sensed_rudder_angle ; //valeur de la mesure
3 int n ; //nombre de cycles d'operation du capteur
```

Nous pouvons ainsi définir la fonction *sense*, modélisant une mesure avec erreur systématique de 2 :

```
1 void sense(){
2         sensed_rudder_angle = rudder_angle + 2 ;
3 }
```

Ou encore ainsi, dans le but de modéliser en plus de l'erreur systématique une erreur liée au vieillissement du composant, ou dérive :

```
1 void sense(){
2         sensed_rudder_angle = rudder_angle + 2 + n/100 ;
3         n += 1 ;
4 }
```

2.2.4.2 Modélisation de la commande

De manière similaire, la commande se modélise à travers l'utilisation de variables dédiées (une pour la valeur de la grandeur physique, une pour la valeur de la consigne) et d'automates représentant le comportement des composants impliqués dans l'élaboration et l'exécution de cette commande (par exemple, les automates (b), (c) et (e) de la figure 2.8 correspondant à l'actionneur du gouvernail, au capteur d'angle de barre et au PLC).

La modélisation d'une boucle de commande d'une grandeur donnée sous la forme d'automates finis temporisés peut permettre de représenter le fonctionnement de régulateurs simples – par exemple des contrôleurs à hystérésis ou des contrôleurs purement proportionnels – n'intégrant pas de part temporelle (en particulier intégrale ou dérivée) à leur fonction de régulation.

Considérons à titre d'exemple la boucle de régulation de l'angle de barre. Ayant déclaré en préambule les variables suivantes :

```
1 int rudder_setpoint_value = 4500 ; //valeur de la consigne
    fixee par l'utilisateur
2 int rudder_command_value ; //valeur de la commande calculee
    par le PLC
```

Et considérant un actionneur ayant une erreur proportionnelle à la commande reçue u, de la forme $e(u) = -\frac{u}{100}$, et représentée à travers la fonction suivante associée à la transition `idle` → `rotating` (*cf.* figure 2.8 (b)) :

```
1 void set_rudder_angle(){
2         rudder_angle = rudder_command_value -
              rudder_command_value/100 ;
3 }
```

Nous pouvons modéliser un régulateur proportionnel $u = x + 0.85e$ à travers la fonction suivante, associée aux transitions `waitingOrder` → `sendingInitialCommand` et `receivingSensorData` → `sendingRetroactionCommand` (*cf.* figure 2.8 (d)) :

```
1 void compute_rudder_command(){
2         int error = rudder_setpoint_value -
              sensed_rudder_angle ;
3         rudder_command_value = rudder_setpoint_value + 85*
              error/100 ;
```

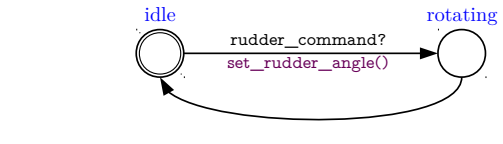
```
4 }
```

NB : du fait de l'impossibilité de manipuler des variables autre qu'entières avec UPPAAL, les valeurs angulaires utilisées par ces fonctions (exprimées en degrés dans le cadre de l'opération du système modélisé) sont multipliées par 100 afin d'exprimer des fractions d'angles.

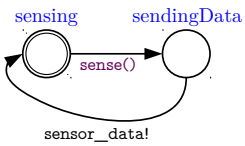
Cette approche de modélisation simple ne permet toutefois pas la représentation fidèle de fonctions de régulation plus élaborées, commes celles mises en œuvre par les régulateurs PID [4], largement utilisés dans les systèmes de contrôle industriels – en particulier dans l'industrie navale [Rob08 ; LJ13 ; Mat+07 ; Jag], étant donné qu'ils s'appuient sur l'évolution de phénomènes continus. Leur modélisation s'effectue donc de manière simpliste par la modification des fonctions associées aux transitions afférentes de l'actionneur et du PLC de telle sorte à ce qu'elles résultent en une simple affectation de la valeur de la consigne à la grandeur régulée. Si la réalité de l'algorithme de régulation ne peut de ce fait être entièrement préservée, il convient néanmoins de maintenir une modélisation du comportement « apparent » du PLC, de l'actionneur et du capteur impliqués par le maintien des échanges réseau entre les trois composants. Cela peut se faire, par exemple, par l'introduction d'un booléen forçant la réalisation d'une itération de la boucle de régulation :

```
1  void set_rudder_angle(){
2      rudder_angle = rudder_command_value + 2 ;
3  }
4
5  void compute_rudder_command(int n){
6  //la commande est appelee avec n = 0 lors de son premier
       appel, et n = 1 lors du suivant
7      if (n == 0){
8          rudder_command_value = rudder_setpoint_value ;
9      }
10     else {
11         rudder_command_value = rudder_setpoint_value - 2 ;
12     }
13 }
```

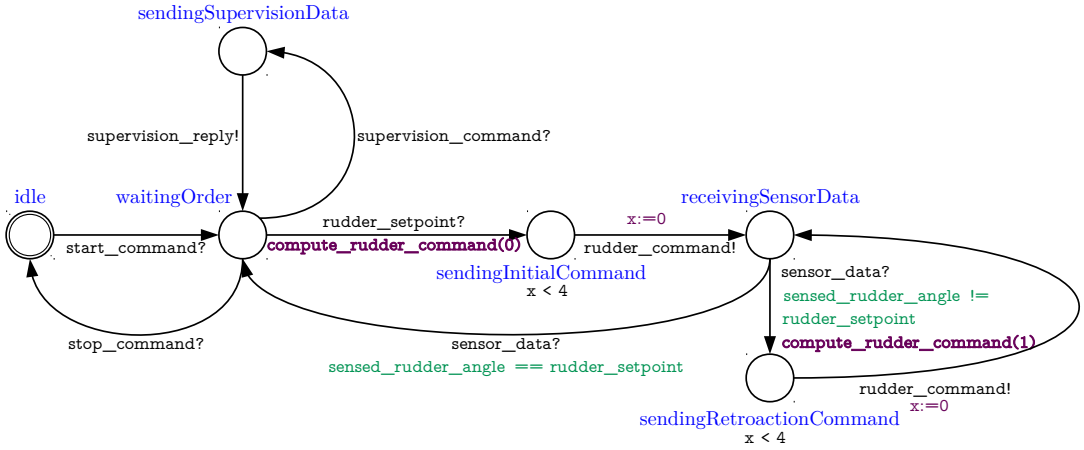
4. Proportionnel – Intégral – Dérivé. Régulateurs s'appuyant sur trois termes : un terme proportionnel à l'erreur, un second proportionnel à son intégrale et un troisième à sa dérivée.



(a) Actionneur du safran



(b) Capteur d'angle de barre



(c) PLC

FIGURE 2.12 – Automates impliqués dans la modélisation simpliste d'une régulation.

Même si une telle modélisation ne retranscrit pas dans son entièreté la réalité de l'algorithme de régulation, elle peut toutefois s'avérer suffisante pour traduire l'impact d'une attaque ou d'un correctif le modifiant. Il est en effet moins souvent crucial d'exprimer comment la valeur de la grandeur régulée va être affectée que d'exprimer le simple fait qu'elle va être affectée – comme nous le verrons ultérieurement, l'utilisation ce modèle simple suffit à le traduire [5].

2.2.4.3 Modélisation de l'évolution des grandeurs non commandées

L'étude de la dynamique du navire en évolution permet d'établir les relations liant entre elles les évolutions des différentes grandeurs du système (*cf.* Annexe B). Nous savons par exemple que dans le référentiel terrestre, les coordonnées x et y du centre de gravité du bâtiment évoluent selon les équations suivantes (*cf.* Annexe B,

5. Par exemple, par l'introduction d'un décalage arbitraire dans la sortie de l'algorithme de régulation considéré.

équation B.9) :

$$\begin{cases} \dot{x} = v \times cos(\theta) + dérive \\ \dot{y} = v \times sin(\theta) + dérive \end{cases} \tag{2.1}$$

θ étant le cap du navire et v sa vitesse tangentielle.

Une résolution numérique pas à pas de ces équations différentielles n'est pas possible dans le formalisme utilisé dans le cadre de cette approche de modélisation simple des grandeurs physiques. Toutefois, une option est de paramétrer au sein de la modélisation d'une mission la durée de chaque tâche impliquant une évolution de la grandeur non commandée que l'on cherche à faire évoluer, les autres grandeurs impliquées dans son évolution étant constantes, et de calculer à l'issue de la tâche la nouvelle valeur de la grandeur étudiée. En reprenant l'exemple des coordonnées du centre de gravité dans le référentiel terrestre, cela revient à exprimer la durée de chaque déplacement à vitesse et cap constants (*cf.* figure 2.13). Cependant, dans

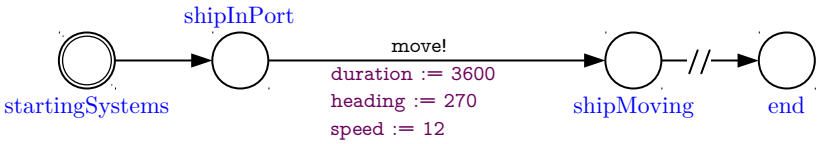


FIGURE 2.13 – Automate modélisant une mission en intégrant la durée de chaque phase.

l'hypothèse où cette démarche de modélisation soit trop lourde et où certaines grandeurs liées au système et non commandées s'avéreraient complexes à calculer du fait du lien unissant entre elles leurs évolutions, il convient de garder à l'esprit que la modélisation des grandeurs régulées des actionneurs peut être suffisante. En effet, comme nous pouvons le constater à la lecture des équations différentielles synthétisées par le système B.9 de l'Annexe B – qui confirment une approche intuitive de la question –, elles se déduisent des commandes des actionneurs. Dans cette optique, l'expression de l'impact d'une attaque ou d'un correctif résultant en la modification de ces grandeurs est toujours possible : il suffit en effet de baser le calcul d'impact sur la formulation de propriétés ayant trait aux grandeurs régulées des actionneurs plutôt qu'aux grandeurs du système en elles-mêmes. Par exemple, une propriété du type *« à la fin de la première phase de déplacement de la mission,*

le bâtiment se situe aux coordonnées (x_i, y_i) » peut dans cette visée se traduire par *« pendant la première phase de déplacement de la mission, l'angle de barre, le rapport de réduction et la vitesse de rotation des moteurs ont toujours été égaux aux consignes*[6] *»*.

2.2.4.4 L'approche de modélisation simple en bref

Cette approche demande un travail de modélisation supplémentaire, en ce qu'elle requiert une analyse et une synthèse des lois de régulation ou des caractéristiques métrologiques des capteurs – qui ne se fait pas sans perte de finesse dans la représentation –, et une multiplication des états et transition au sein des automates représentant les missions. Toutefois, elle présente l'avantage de conserver des modèles lisibles et compréhensibles ainsi que de ne pas augmenter excessivement les temps de calcul inhérents à leur exploration ; par ailleurs, si les « raccourcis » de modélisation (l'abstraction des lois de commande, des caractéristiques métrologiques ou de l'évolution des grandeurs non commandées) sont correctement établis, elle permet une représentation des grandeurs physiques du système suffisante aux besoins de calcul d'impact de notre processus.

2.2.5 Axe d'amélioration : l'utilisation d'automates hybrides

La seconde approche d'intégration des grandeurs physiques à notre modèle réside en l'utilisation d'automates hybrides. Un automate hybride est un automate tel que défini en amont (*cf.* définition 5), étendu par un ensemble de *variables* continues dont l'évolution est décrite par un jeu d'*équations différentielles*[7]. Ce formalisme paraît tout-à-fait approprié à notre besoin de modélisation et s'avère compatible avec notre choix d'outils : via son extension UPPAAL-SMC [Dav+15] où les équations différentielles sont résolues numériquement via l'application de la méthode d'Euler, UPPAAL permet la représentation et la simulation d'automates hybrides – il a notamment été utilisé avec succès pour l'abstraction de la dynamique de véhicules autonomes [Gu+19] –; par ailleurs, les automates hybrides peuvent

6. La valeur du cap et des coordonnées du bâtiment dépendant de ces trois paramètres régulés, comme nous l'établissons dans l'Annexe B.

7. Formellement, un « simple » automate temporisé est déjà un automate hybride : ses horloges modélisent l'évolution (uniforme) d'une grandeur continue (le temps).

être utilisés pour modéliser des régulateurs PID [Sek+14].

Les variables et équations différentielles des automates hybrides sont extraites lors de la collecte et synthétisées lors de la fédération des modèles, tâches menées lors de la première phase du processus que nous proposons au chapitre 1 : aux variables correspondent les grandeurs physiques liées au système et à ses actionneurs, aux équations différentielles les informations déduites des lois de commande (implémentant pour leurs besoins d'asservissement les équations différentielles décrivant l'évolution des grandeurs régulées) et des modèles dynamiques établis lors de la conception du bâtiment – en leur absence (les modèles dynamiques issus des phases d'étude, notamment relatifs aux bâtiments anciens, peuvent ne plus être disponibles), l'analyse de la dynamique du système étudié permet de les obtenir. L'évolution simplifiée d'un navire peut par exemple être décrite par ce jeu d'équations différentielles (*cf.* Annexe B, équation B.9) :

$$\begin{cases} \dot{x} = vcos(\theta) + dérive \\ \dot{y} = vsin(\theta) + dérive \\ \dot{\theta} = \omega \\ \dot{v} = \frac{K_t \rho n^2 D^4 - \alpha_g v \sin^2(\delta_g) - \alpha_f v}{m} \\ \dot{\omega} = \frac{r_g \alpha_g v \cos(\delta_g) \sin(\delta_g) - \alpha_\theta \omega}{J} \end{cases} \tag{2.2}$$

où le couple (x, y) représente les coordonnées du bâtiment dans le référentiel terrestre, θ son angle de cap, v sa vitesse tangentielle, ω sa vitesse angulaire (représentant ses mouvements de giration autour d'un axe vertical passant par son centre de gravité), δ_g son angle de barre et n la vitesse de rotation de son arbre moteur. Les autres grandeurs impliquées dans ces équations sont des constantes liées au système et dont la signification est donnée en annexe B.

L'avantage d'une modélisation basée sur les automates hybrides est triple :

— d'une part, son établissement nécessite moins d'interprétation étant donné qu'il n'y a pas de nécessité d'analyser et de représenter de manière simpliste les algorithmes des régulateurs impliqués dans la commande des actionneurs : leur « vrai » comportement est directement transcrit dans le modèle et donc un risque de distorsion est supprimé ;

— d'autre part, elle permet de suivre l'évolution des grandeurs descriptives de la dynamique du système (e.g. coordonnées, cap, vitesse, etc.) avec une granularité « temporelle » plus fine – il n'est par exemple pas nécessaire d'attendre la fin d'une phase de déplacement à vitesse et cap constants pour calculer les coordonnées du bâtiment – et elle permet de toutes les représenter. De ce fait, la formulation des propriétés à vérifier dans le cadre

du calcul d'impact ne nécessite pas la « traduction » évoquée précédemment et peut donc se focaliser sur l'expression d'objectifs opérationnels, plus naturels à transcrire ;
— enfin, une plus grande fidélité dans la modélisation de l'évolution temporelle des caractéristiques métrologiques des capteurs et donc, de l'erreur de mesure liée à leur vieillissement, peut être permise par une modélisation basée sur les automates hybrides.

Un désavantage de cette méthode réside toutefois dans les performances de la vérification du réseau d'automates ainsi étendu, la vérification de modèles hybrides étant plus coûteuse que celle de modèles non-hybrides [Iva+18].

2.3 Modéliser les vulnérabilités, attaques & contremesures

De la finalité du processus proposé au chapitre 1 découle un second besoin de modélisation, permettant de rendre compte de la présence de vulnérabilités affectant le système étudié, du déploiement des contremesures les tempérant, et de la conduite potentielle des attaques les exploitant.

Nous proposons en réponse à cette problématique annexe un formalisme de modélisation s'intégrant à celui – relatif au système – que nous venons d'exposer à la section précédente et qui s'appuie sur la notion de *mutation d'automates* ou de *réseaux d'automates*. La suite de ce chapitre s'attachera à expliquer ce concept et à mettre en évidence sa pertinence relative à notre besoin de modélisation ; nous y proposerons enfin un langage permettant de décrire les vulnérabilités, contremesures et attaques à travers son prisme.

2.3.1 Différents types de modifications induites par une vulnérabilité, une attaque ou une contremesure : essai de catégorisation

L'objet des mutations présentées dans ce chapitre étant de décrire un ensemble de modifications apportées au système, il paraît important d'introduire liminairement un aperçu des différents types de modifications pouvant résulter d'attaques l'affectant ou de correctifs tempérant ses vulnérabilités. La seule pré-

sence de ces dernières n'apporte a priori pas de modification au système, mais peut toutefois se traduire par une variation de ses modèles : représenter une faille consiste en effet à rendre compte de la potentialité de comportements nouveaux résultant de son exploitation. Ces comportements étant couverts par l'étude des modifications découlant potentiellement des attaques l'exploitant, nous nous bornerons donc ici à lister les modifications *du système* liées aux attaques et contremesures.

Scott Musman *et al.* [Mus+10 ; Mus+11] proposent, à des fins de simulation de son impact sur un système cyber-physique affecté, une catégorisation des différentes conséquences possibles d'une attaque. Ces conséquences, que nous cherchons à traduire par les mutations des modèles du système, sont synthétisées au tableau suivant issu de [Mus+11] :

« ***Catégorie***	***Effets sur les processus***	***Effets sur les données***
Interruption	*Le processus est indisponible pour une durée donnée et le demeurera jusqu'à résolution de l'incident*	*Les données sont indisponibles pour une durée donnée*
Dégradation	*L'exécution du processus est ralentie*	*La fréquence de réception des données est diminuée ; La qualité ou la précision de l'information produite par l'activité impactée est affectée*
Modification	*Le processus* [incluant son séquencement, les logiciels qui le sous-tendent, etc.] *a été altéré d'une manière pouvant altérer ses sorties, résultats,* [son comportement ou provoquer son arrêt]	*Les données ont été modifiées – par conséquent les processus les utilisant peuvent échouer ou produire des résultats incorrects*
Insertion	*Une mission illégitime a été affectée au système, pouvant interférer avec ses missions légitimes ;* [un processus a été créé ou démarré]	*Des données ont été illégitimement introduites dans le système*

Interception	*Des éléments du processus (e.g. les logiciels ou composants matériels qui le sous-tendent) ont été interceptés*	*Des données ont été interceptées*
Utilisation indue	*Crée les conditions propices à la survenue des effets précédents sur le processus, ou d'autres conséquences imprévisibles*	*Crée les conditions propices à la survenue des effets précédents sur les données* »

TABLE 2.1 – Catégorisation des cyber-attaques selon leurs effets, telle que proposée dans [Mus+11].

De la même manière, nous pouvons ainsi définir les différentes conséquences d'une contremesure sur le système visé par son déploiement. À la différence de celles découlant d'une attaque, elles peuvent affecter positivement ou négativement le système :

Catégorie	**Effets sur les processus**	**Effets sur les données**
Interruption	*cf.* table 2.1	*cf.* table 2.1
Dégradation	*cf.* table 2.1	*cf.* table 2.1
Amélioration	L'exécution du processus est fluidifiée	« *La fréquence de réception des données est* [améliorée] *; La qualité ou la précision de l'information produite par l'activité impactée* [par le déploiement de la contremesure est améliorée] [Mus+11] »

Modification, incluant la modification de la topologie réseau et l'ajout ou la suppression de composants et de logiciels	« *Le processus* [incluant notamment son séquencement, les logiciels ou l'architecture des sous-systèmes qui le sous-tendent] *a été altéré d'une manière pouvant* [affecter positivement ou négativement/ *ses sorties ou résultats,* [et permettre ou prévenir sa communication avec les autres processus du système] »	« *Les données ont été modifiées* [ou sont inaccessibles] – *par conséquent les processus les utilisant peuvent échouer ou produire des résultats incorrects* [ou hors délai] » [Mus+11]

TABLE 2.2 – Catégorisation des effets possibles d'une contremesure.

2.3.2 Des mutations du réseau d'automates pour la modélisation des modifications du système

Notre postulat de modélisation est que ces modifications possibles peuvent être traduites par des mutations des automates modélisant le système étudié [Sul+18]. Nous pouvons définir ces mutations, s'appliquant soit à un automate donné, soit à un réseau d'automates, comme suit :

Définition 6 (**Mutation** (dérivée de [Sul+18]))**.**
La ***mutation d'un automate*** $\mathcal{A}$ *tel que défini à la section 2.2 correspond à la modification, l'ajout ou la suppression de l'un de ses états, horloges, variables, contantes, invariants, fonctions et transitions (incluant la modification, l'ajout ou la suppression de synchronisations, mises à jour et gardes) résultant en un automate* $\mathcal{A}'$ *valide.*
La ***mutation d'un réseau d'automates*** $\mathcal{R} = \{\mathcal{A}_0, \ldots, \mathcal{A}_n, \mathcal{X}, \mathcal{F}\}$ *($\mathcal{X}$ représentant l'ensemble des horloges, variables et constantes partagées entre les automates de R et $\mathcal{F}$ l'ensemble des fonctions partagées entre ces mêmes automates) correspond à la mutation ou la suppression d'un des automates $\mathcal{A}_i$ qui le composent, l'ajout d'un automate $\mathcal{A}_{n+1}$ à sa composition, ou la modification, l'ajout ou la suppression d'éléments de $\mathcal{X}$ ou de $\mathcal{F}$.*

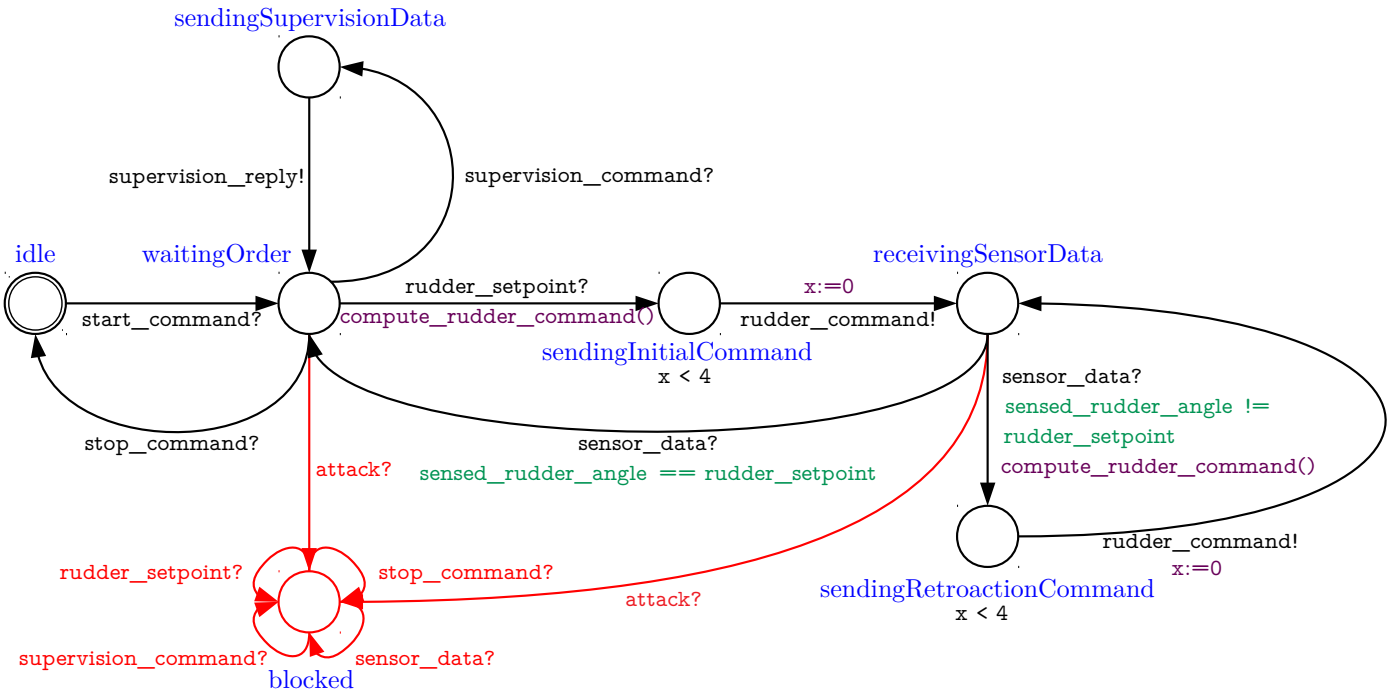


FIGURE 2.14 – Automate modélisant un PLC contrôlant le gouvernail d'un bâtiment, après prise en compte d'une vulnérabilité pouvant entraîner une interruption subséquente à une attaque par déni de service.

À titre d'exemple, l'automate représenté en figure 2.14 est issu de sept mutations appliquées à l'automate représenté en figure 2.6 :

1. l'ajout d'un état « puits » `blocked` ;
2. l'ajout d'une transition `waitingOrder` → `blocked`, synchronisée avec une transition ayant pour action la synchronisation `attack!` ;
3. l'ajout d'une transition `receivingSensorData` → `blocked`, synchronisée avec une transition ayant pour action la synchronisation `attack!` ;

4 à 7. l'ajout de quatre transitions `blocked` → `blocked`, « consommant » les synchronisations déclenchant les transitions préexistantes issues des états `waitingOrder` et `receivingSensorData` et maintenant l'automate dans l'état puits ;

De telles mutations permettent de figurer la présence d'une vulnérabilité inhérente au système d'exploitation du PLC, pouvant être exploitée dans le but d'entraîner une interruption du fonctionnement du composant par déni de service. Nous pouvons, de manière plus générale, proposer des exemples de mutation gé-

nériques pouvant servir de base à la modification des automates impliquées par la découverte d'une vulnérabilité, la conduite d'une attaque ou le déploiement d'un correctif et correspondant aux catégories précédemment définies.

2.3.2.1 Des mutations pour décrire les différentes modifications catégorisées à la section 2.3.1

i. Interruption

a. Interruption définitive

Mutations possibles

Un état « puits » est inséré dans l'automate modélisant le composant concerné ; cet état est atteint lors de l'attaque (dans le cadre de la modélisation des conséquences possibles d'une attaque découlant d'une vulnérabilité découverte), ou devient l'état initial de l'automate (dans le cadre de la modélisation de l'effet de bord d'une contremesure) et aucun autre état ne peut être atteint depuis cet état « puits ».

En pratique

Nous pouvons considérer l'application des mutations suivantes à l'automate représentant le composant affecté, selon que l'on cherche à modéliser une interruption découlant d'une attaque ou du déploiement d'une contremesure :

— création d'un nouvel état ;
— création d'un canal de synchronisation modélisant le début de l'attaque ;
— création de transitions vers l'état nouvellement inséré, se synchronisant via le canal précédemment créé ;
— création d'un ensemble de transitions bouclant sur le nouvel état, permettant de « consommer » les messages de synchronisation reçus par l'automate.

Ou bien :

— création d'un nouvel état initial ;
— création d'un ensemble de transitions bouclant sur l'état nouvellement inséré, permettant de « consommer » les messages de synchronisation reçus par l'automate.

L'automate représenté en figure 2.14 permet ainsi de modéliser une potentielle interruption définitive du fonctionnement d'un PLC découlant d'une attaque –

en d'autres termes, de modéliser la présence d'une vulnérabilité la permettant. Notons que les états liés à l'état puits par les transitions créées en vertu de ce qui précède ont été choisis à dessein : étant donné que nous avons cherché à modéliser une vulnérabilité menant à des attaques par déni de service, il s'agit des états représentant une écoute réseau du PLC.

b. Interruption temporaire

Mutations possibles

En plus des mutations que nous venons d'exposer, nous pouvons utiliser les temporisations de l'automate pour modéliser les interruptions temporaires. La durée t de l'interruption (qui peut par exemple représenter la durée d'une attaque ou du déploiement d'une contremesure) est modélisée par une horloge, qui est ajoutée à l'automate modélisant le composant concerné s'il n'en possédait pas préalablement. Un nouvel état est ensuite créé : cet état est atteint lors de l'attaque, (dans le cadre de la modélisation des conséquences possibles d'une attaque découlant d'une vulnérabilité découverte), ou devient l'état initial de l'automate (dans le cadre de la modélisation de l'effet de bord d'une contremesure) et aucun autre état ne peut être atteint tant que la valeur de l'horloge, réinitialisée lors de l'atteinte de l'état, est inférieure à t.

En pratique

Nous pouvons considérer l'application des mutations suivantes à l'automate représentant le composant affecté, selon que l'on cherche à modéliser une interruption découlant d'une attaque ou du déploiement d'une contremesure :

— création d'une horloge x si nécessaire ;
— création d'un nouvel état ;
— création d'un canal de synchronisation modélisant le début de l'attaque ;
— création de transitions vers l'état nouvellement inséré, se synchronisant via le canal précédemment créé ;
— création de transitions de l'état nouvellement inséré vers les états choisis, ayant pour garde $x \geq t$ et réinitialisant x ;
— création d'un ensemble de transitions bouclant sur l'état nouvellement inséré, permettant de « consommer » les messages de synchronisation reçus par l'automate.

Ou bien :

— création d'une horloge x si nécessaire ;

— création d'un nouvel état initial ;
— création de transitions de l'état nouvellement inséré vers l'état initial précédent, ayant pour garde $x \geq t$ et réinitialisant x ;
— création d'un ensemble de transitions bouclant sur l'état nouvellement inséré, permettant de « consommer » les messages de synchronisation reçus par l'automate.

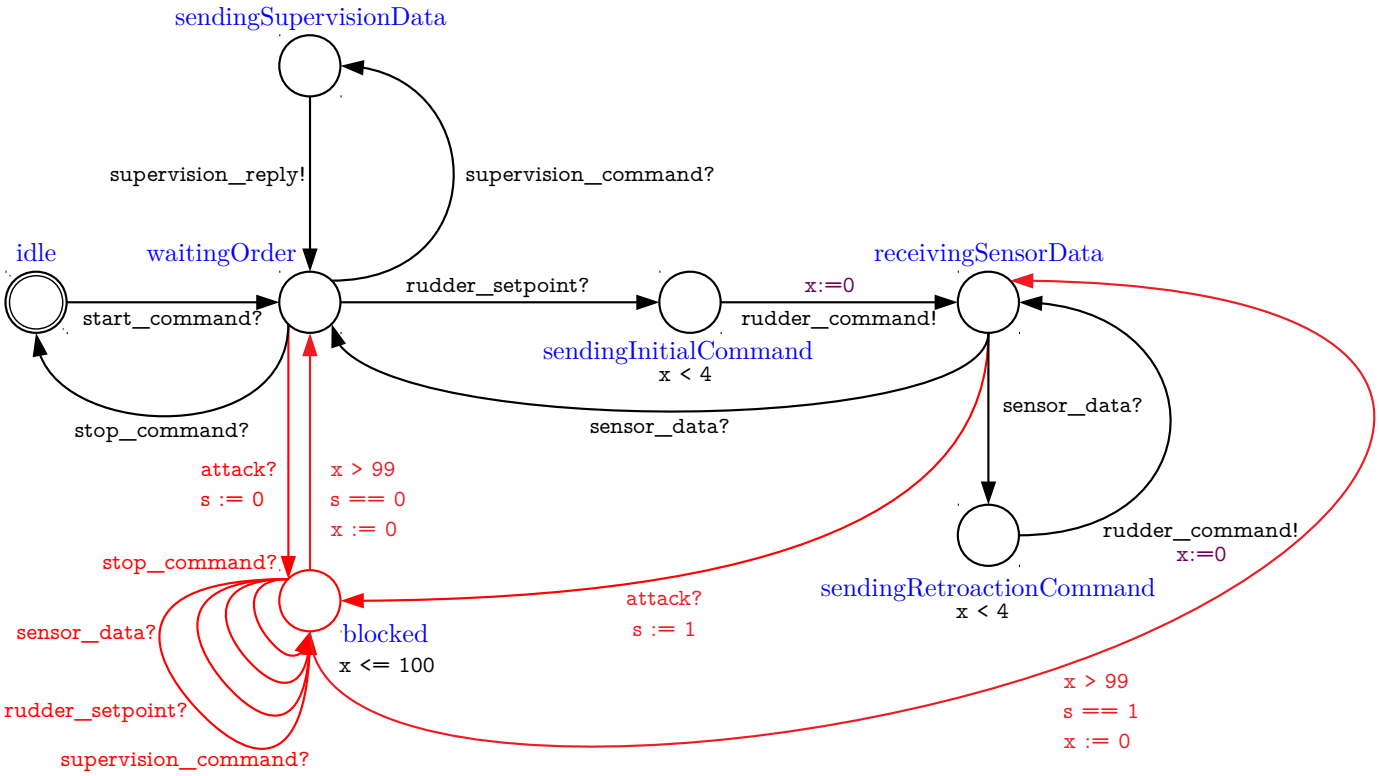


FIGURE 2.15 – Automate modélisant un PLC contrôlant le gouvernail d'un bâtiment, après prise en compte d'une vulnérabilité pouvant entraîner une interruption temporaire subséquente à une attaque par déni de service. Pour des raisons de lisibilité, les gardes et mises à jour de variables hors celles insérées ou modifiées par la mutation ne sont pas mentionnées.

En poursuivant à titre d'exemple nos mutations sur l'automate représenté en figure 2.6, nous obtenons un nouvel automate (*cf.* figure 2.16) illustrant l'application de l'ensemble de mutations précédemment décrit afin de modéliser une vulnérabilité permettant des attaques pouvant mener à une interruption temporaire du fonctionnement de notre PLC.

ii. Dégradation

a. Augmentation des temps de fonctionnement

Mutations possibles

La modélisation d'une variation des délais de fonctionnement se fait simplement par l'utilisation de la temporisation des automates concernés. Ainsi, les invariants ou gardes figurant les temps de fonctionnement des composants affectés sont incrémentés.

En pratique

— création d'une horloge x (si nécessaire) ;
— modification des transitions issues de l'état (ou des états) représentant l'action dont le temps de fonctionnement est augmenté, en définissant pour garde $x > t$, t étant le nouveau délai de fonctionnement ;
— si nécessaire, augmenter le second terme des invariants d'horloge associés aux états concernés.

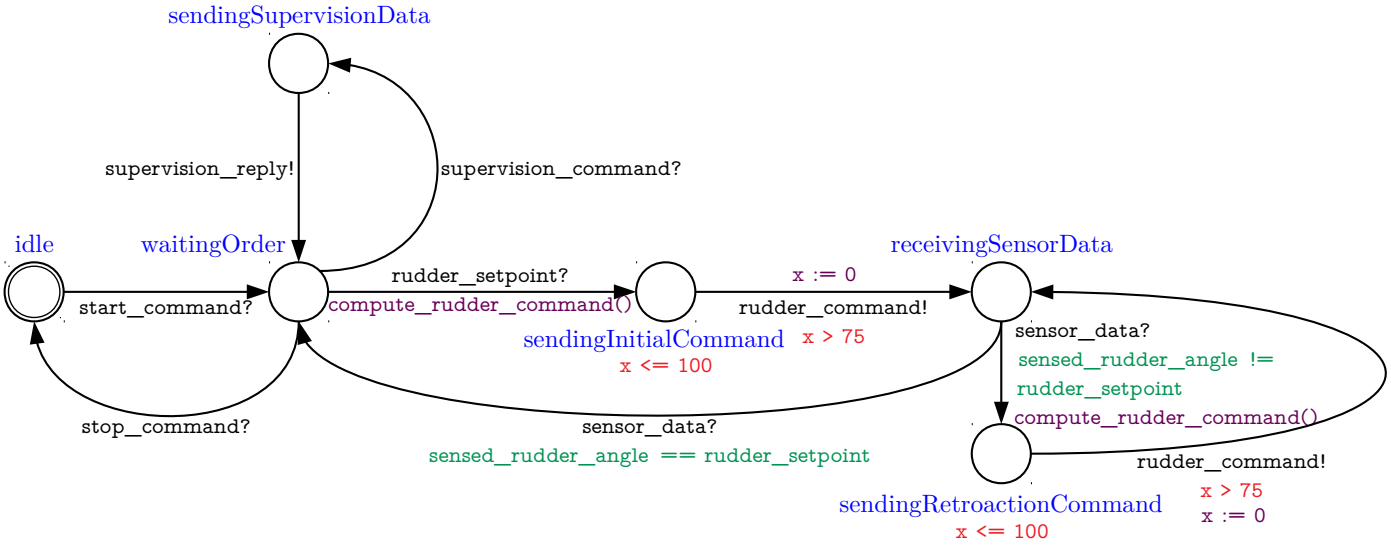


FIGURE 2.16 – Automate modélisant un PLC contrôlant le gouvernail d'un bâtiment, après prise en compte d'une vulnérabilité pouvant entraîner une augmentation de ses temps de fonctionnement subséquente à une attaque par déni de service.

À titre d'illustration, ces mutations peuvent aboutir à l'automate représenté en figure 2.16.

b. Dégradation physique ou logique

Ce type de dégradation est couvert par la notion de *modification*, que nous aborderons au paragraphe suivant.

c. Perte de messages

Mutations possibles

Pouvant résulter d'une dégradation de la topologie réseau ou de la dégradation physique ou logique des composants générant les messages échangés par le système au cours de son fonctionnement, la perte de messages peut se traduire par des mutations des *automates topologiques* figurant la topologie réseau. En effet, nous pouvons les faire muter de telle sorte à ce qu'ils transmettent les messages entre les différents automates de manière amoindrie (aléatoire ou non). Notons qu'une mutation des automates dont les transitions se synchronisent avec l'automate topologique muté peut être nécessaire afin d'éviter des interblocages du modèle, comme nous l'évoquerons dans l'exemple suivant.

En pratique

- systématiquement, dans l'automate topologique concerné :
  - modification de la transition liant le « second » état à l'état initial et effectuant l'action de synchronisation sortante par l'ajout d'une garde subordonnant son exécution à la condition exprimée par la garde (par exemple un test de congruence modulo n sur un compteur incrémenté à chaque envoi de message par l'automate composant situé en amont, permettant d'exécuter la transition une fois sur n) ;
  - création d'une transition parallèle, conditionnée par la garde « complémentaire », permettant à l'automate topologique de revenir à son état initial ;
- si besoin, afin d'éviter tout interblocage du réseau d'automates :
  - création d'un canal de synchronisation `c` en diffusion multiple, et ajout de l'action de synchronisation sortante `c!` à la transition parallèle ajoutée à l'automate topologique ;
  - dans les automates composant situés en amont et en aval de l'automate topologique concerné, création de transitions se synchronisant avec la transition mentionnée ci-dessus par une co-action `c?`.

À titre d'exemple, de telles mutations appliquées aux automates présentés en

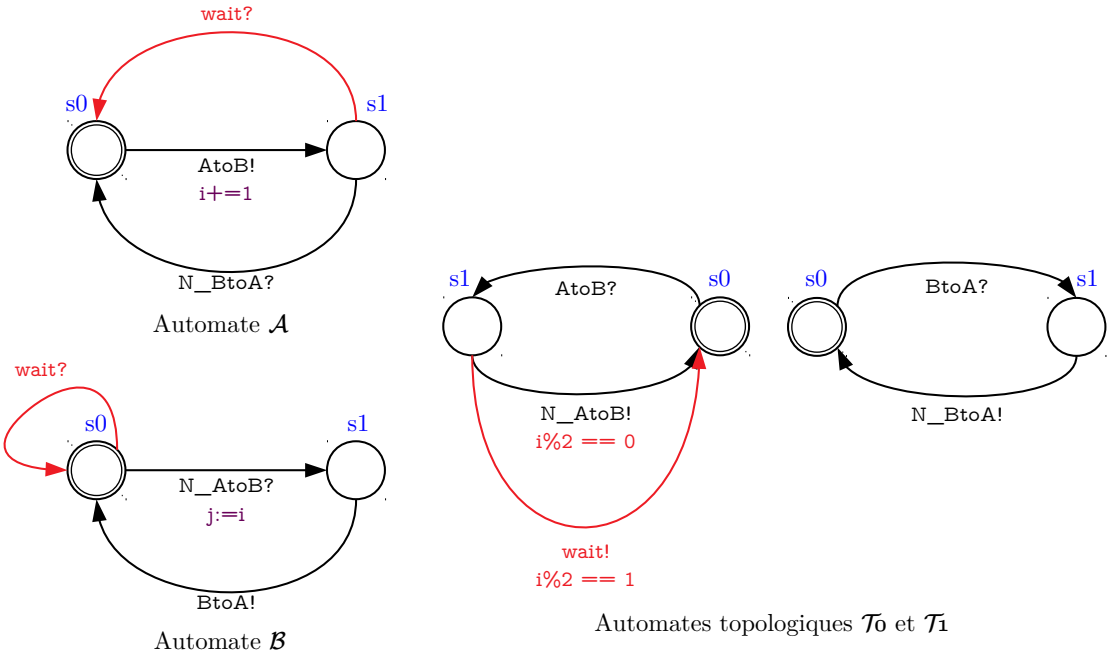


FIGURE 2.17 – Modélisation possible d'une perte de messages par mutation d'automates topologiques.

figure 2.7 peuvent résulter en les automates de la figure 2.17, où le lien entre le composant modélisé par l'automate $\mathcal{A}$ et l'automate $\mathcal{B}$ est perturbé de telle sorte qu'il ne permette la transmission que d'un message sur deux. La vérification de la propriété *« i est systématiquement égal à j lorsque l'automate T0 atteint son état s0 »*, traduisant la bonne réception du contenu du message (i) envoyé par $\mathcal{A}$ à $\mathcal{B}$ suite à sa transmission via $\mathcal{T}_0$ (`A[] T0.s0 imply i == j` dans la syntaxe des requêtes d'UPPAAL) permet aisément de vérifier la (non) transmission du message.

ii bis. Amélioration

Une amélioration subséquente au déploiement d'une contremesure est ici entendue dans le sens d'une diminution des temps de fonctionnement des processus et composants affectés ; elle se traduit simplement par des mutations symétriques à celles relatives à l'augmentation des temps de fonctionnement et présentées au paragraphe précédent, point *a.* – à ceci près que les seconds membres de l'expression des invariants et des gardes éventuelles modélisant ces temps de fonctionnement sont décrémentés au lieu d'être incrémentés.

iii. Modification

a. Modification du logiciel

Une modification d'un logiciel déployé sur un composant donné – revenant sur le plan comportemental à une modification des actions pouvant être effectuées par le composant – peut se traduire d'une part par l'ajout ou la suppression d'états, et d'autre part par l'ajout, la suppression ou la modification des transitions (y compris leurs gardes, synchronisations et mises à jour, en particulier du code source des fonctions appelées par ces mises à jour).

NB : dans le cas de la modélisation d'une attaque affectant le comportement d'un composant impliqué dans la mise en œuvre de la régulation d'une grandeur physique, un travail supplémentaire d'analyse est à mener si l'approche retenue pour la modélisation des grandeurs physiques est l'approche « simple » et non pas l'approche « hybride » et que le régulateur affecté intègre une part temporelle dans les lois de commande qu'il applique (typiquement, s'il s'agit d'un régulateur PID).

b. Modification de la topologie

Du fait de notre démarche de modélisation, une modification de la topologie du réseau du système se traduit simplement par l'ajout ou la suppression d'automates topologiques et des canaux de synchronisation associés, ainsi que la modification des transitions des automates composant associées à ces canaux – pour les prendre en compte ou les supprimer.

iv. Insertion

L'insertion recouvre la création d'une mission ou d'une étape d'une mission existante, ou encore la création de données illégitimes. De ce fait, deux cas se distinguent :

1. création d'un nouvel automate mission et éventuellement de canaux de synchronisation servant à le lier aux automates processus, et mutation des automates processus afin que leurs transitions se synchronisent avec cette nouvelle mission ;
2. création de nouveaux états au sein d'un automate mission existant et éventuellement de canaux de synchronisation servant à le lier aux automates

processus, et mutation des automates processus afin que leurs transitions se synchronisent avec ces nouveaux états ;

3. création de variables ou modification de la valeur des variables utilisées par le modèle (par exemple la valeur des consignes, des données issues des capteurs, ...).

v. Interception

La confidentialité des données contenues par le système est probablement l'élément le moins aisé et naturel à modéliser étant donné notre formalisme axé par essence sur le comportement du système. Nous pouvons toutefois, de manière très simpliste, associer une variable booléenne à chaque donnée ou type de données utilisé par le système (par exemple, une variable par mot de passe d'administration de chaque composant, une variable par grandeur physique mesurée, une variable associée aux données produites par le système – les jeux de mesures bathymétriques dans le cas d'un navire effectuant la mission décrite en figure 2.5 –, ...). Cette variable booléenne représente la *confidentialité* de la donnée ou du type de données, et a pour valeurs `true` si la confidentialité est assurée, `false` sinon. Par conséquent, chacune de ces variables est initialisée à `true` dans l'automate système nominal ; les mutations possibles consistent par conséquent à effectuer, à chaque interception d'une donnée lors d'une attaque, une mise à jour de sa variable de confidentialité. Cette mise à jour sera donc effectuée en effet de bord d'une transition synchronisée avec celle d'un automate d'attaque, comme nous l'évoquerons à la section suivante.

À titre d'exemple, en considérant que le PLC représenté en figure 2.6 présente une vulnérabilité permettant à un attaquant de prendre connaissance de son mot de passe d'administration, les mutations possibles pourraient consister en l'ajout d'une variable `password_c` représentant la confidentialité dudit mot de passe, et d'une transition synchronisée avec un automate d'attaque et mettant à jour (à `false`) cette variable. Cette nouvelle transition a été arbitrairement choisie comme bouclant sur l'état `sendingInitialCommand` (dans l'idée de modéliser une attaque suffisamment élaborée pour « exfiltrer » des données au moment où le composant est dans une phase d'émission de messages réseau, *cf.* figure 2.18) mais aurait pu boucler sur les états `sendingRetroactionCommand` ou `sendingSupervisionData`.

vi. Utilisation indue

L'utilisation indue d'un ou plusieurs composants du système peut entraîner les

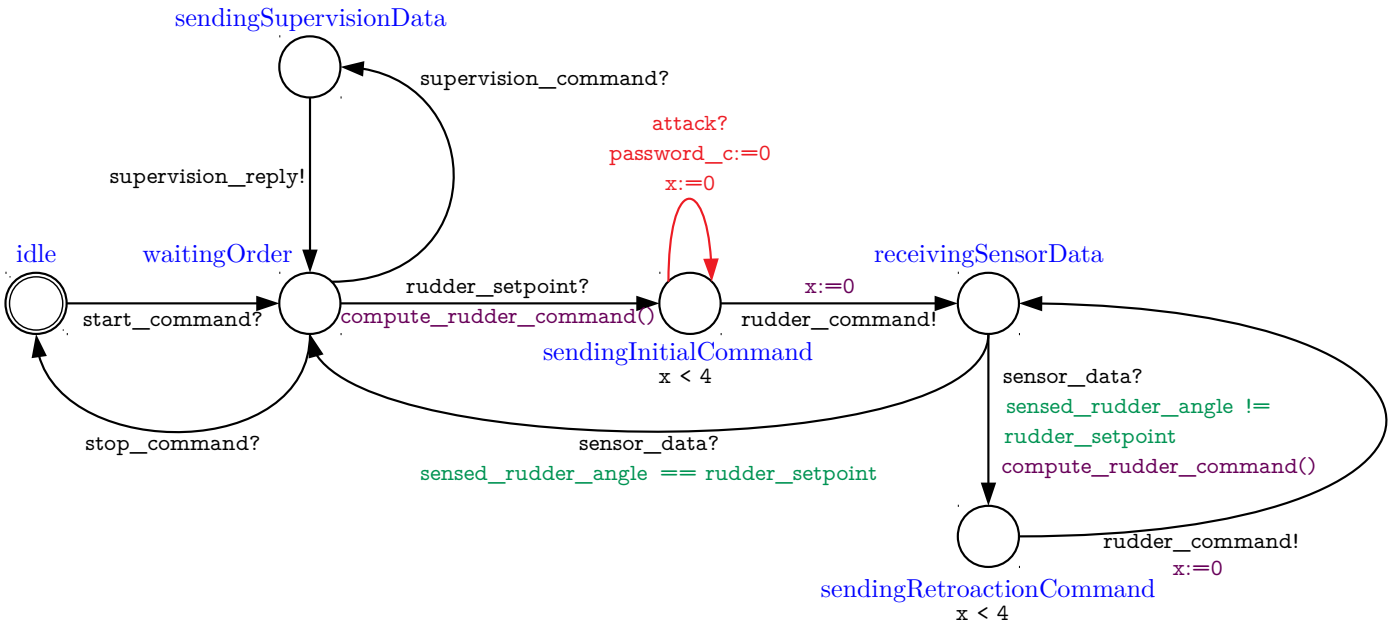


FIGURE 2.18 – Automate modélisant un PLC contrôlant le gouvernail d'un bâtiment, après prise en compte d'une vulnérabilité pouvant permettre à un attaquant de prendre connaissance de son mot de passe d'administration.

précédents types de conséquences.

2.3.2.2 Des automates pour modéliser le déroulement des attaques

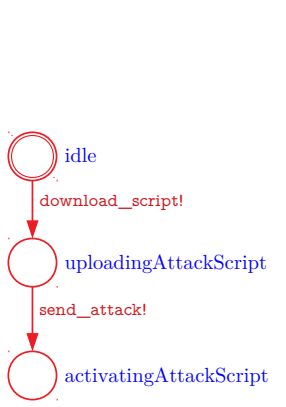
Si les mutations que nous venons de décrire sont nécessaires afin de modéliser la présence d'une vulnérabilité ou le déploiement d'une contremesure, la composition de l'automate système ainsi muté avec des automates modélisant le comportement des attaquants potentiels et établis ad-hoc est nécessaire afin de calculer l'impact des attaques exploitant les vulnérabilités considérées, comme nous l'évoquerons au chapitre 3. Notons que la génération de ces *automates d'attaque* et leur composition avec l'automate système muté est un ensemble de mutations du réseau d'automates au sens décrit par la définition 6.

Ces automates d'attaque sont construits de manière à transcrire le plus fidèlement possible les informations contenues dans les scenarii d'attaque établis lors de la tâche 1-3 du processus exposé au chapitre 1. Un scénario d'attaque étant

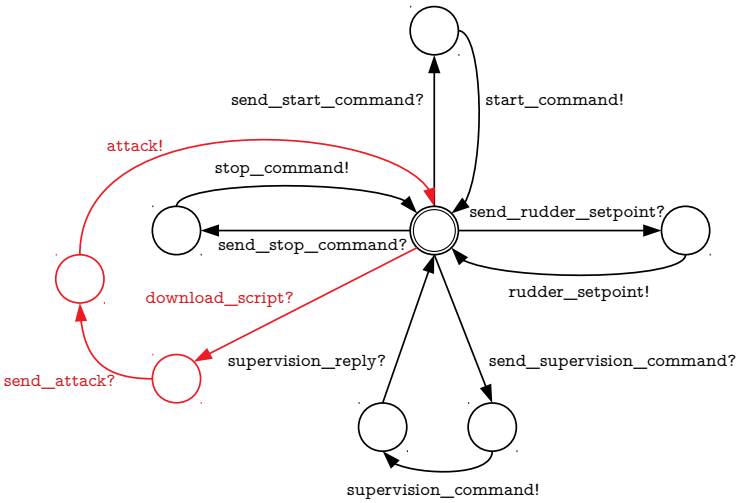
essentiellement une séquence d'actions (qui peut notamment être synthétisée sous la forme d'*arbres d'attaque* [Sch99 ; Sch15]), il est aisé de déduire les automates correspondants, de manière homologue à la modélisation des missions. Imaginons, par exemple, une attaque par déni de service visant le PLC utilisé en exemple dans ce chapitre, et exploitant dans cet ordre deux vulnérabilités :

— une vulnérabilité (1) autorisant une exécution de code arbitraire sur le composant portant l'interface utilisateur ;
— une vulnérabilité (2) permettant une attaque par déni de service sur le PLC et prise en compte en figure 2.14.

Nous supposons que l'attaquant utilise la vulnérabilité (1) pour installer puis exécuter un logiciel malveillant servant à l'envoi des messages réseau menant au déni de service par exploitation de la vulnérabilité (2) : la figure 2.19 représente les mutations nécessaires pour prendre en compte la vulnérabilité (1) sur l'automate modélisant l'IHM (*cf.* figure 2.8) et l'automate d'attaque exploitant ces deux vulnérabilités – les mutations traduisant la présence de la vulnérabilité (2) sur le PLC ayant pour mémoire déjà été décrites en figure 2.14.



(a) Automate d'attaque



(b) Interface utilisateur

FIGURE 2.19 – Automate d'attaque et automate modélisant le comportement du composant portant l'IHM, après mutation.

Notons que nous pouvons utiliser les variables booléennes modélisant la confidentialité des données, introduites par les mutations représentant une interception, afin de conditionner par des gardes les transitions liant deux états successifs d'un

automate d'attaque. Cela peut, à titre d'exemple, être utile à la modélisation d'une séquence d'attaque impliquant une prise de connaissance d'un mot de passe donné comme préalable à ses étapes suivantes.

*
* *

Les sections qui précèdent nous ont permis de le montrer : en appliquant des mutations à un automate ou à un réseau d'automates, nous sommes en mesure de modéliser la présence d'une vulnérabilité, le déploiement d'une contremesure ou la conduite d'une attaque sur le système modélisé selon les principes établis au titre 2.2. Nous nous attacherons dans ce qui va suivre à présenter un langage de description de ces mutations permettant une génération automatisée des modèles mutés.

2.3.2.3 Un langage pour décrire les mutations

Du fait du formalisme rigoureux des automates finis temporisés, il est aisé de décrire les mutations qui leur sont relatives. Nous pouvons, en vertu de la définition 6, distinguer deux cas de figure :

1. les mutations du *réseau d'automates*, revenant à ajouter ou supprimer un automate, une horloge, une constante ou une variable partagées ;
2. les mutations d'un *automate*, revenant à ajouter, supprimer ou modifier un état, une transition, une horloge, une variable ou une constante.

De ce fait, nous pouvons proposer le formalisme suivant afin de décrire chaque mutation :

- `add automaton(`*automaton_name*`, clocks="`*clock*$_0$`, ..., `*clock*$_n$`",`
  `variables="`*variable*$_0$*_type variable*$_0$*_name, ..., variable*$_p$*_type*
  variable$_p$*_name*`")`
- `remove automaton` *automaton_name*
- `add variable` *variable_type variable_name*
- `add clock` *clock_name*
- `in automaton` *automaton_name*
  - `add`
    - `clock` *clock_name*
    - `variable` *variable_type variable_name*

- `state(`*state_name*`, invariant="`*invariant*`")`
- `transition(`*transition_description*`)`

- `remove`
  - `clock` *clock_name*
  - `variable` *variable_type variable_name*
  - `state` *state_name*
  - `transition(`*transition_description*`)`
- `in transition(`*transition_description*`)`
  - `guard becomes` *new_guard*
  - `update becomes` *new_update_list*
  - `synchronisation becomes` *new_synchronisation*
  - `remove`
    - `guard`
    - `update`
    - `synchronisation`
- `in state` *state_name*
  - `invariant becomes` *new_invariant*
  - `remove invariant`

Notons que :

1. si un automate nouvellement créé ne possède pas d'horloge ou de variable, ou un état nouvellement créé ne possède pas d'invariant, les valeurs entre guillemets des champs afférents seront remplacées par le symbole `none` ;
2. le champ *transition_description* est constitué des paramètres suivants : *initial_state_name*, *final_state_name*, `guard="`*guard*`"`, `update="`$update_0$, ..., $update_n$`"`, `synchronisation="`*channel_name*$\{!, ?\}$`"`.
   Si la transition ne possède pas de garde, de mises à jour ou de synchronisation, les valeurs entre guillemets des champs afférents seront remplacées par le mot réservé `none` ;
3. les variables et les constantes sont manipulées à travers le même identifiant (`variable`) : une constante se différenciera par son type ayant `const` pour premier terme ;
4. les valeurs des champs doivent être conformes à la syntaxe utilisée par UPPAAL ;
5. si les appels de fonctions peuvent être modifiés par ce biais (via les champs manipulant les mises à jour), la mutation du code-source des fonctions

sous-jacentes ne peut être décrite à travers ce langage et devra par conséquent être réalisée « manuellement ».

À titre d'exemple, les mutations permettant la transformation de l'automate représenté en figure 2.4 en celui représenté en figure 2.14 sont ainsi décrites dans ce langage :

```
1 in automata PLC add state blocked
2 in automata PLC add transition(waitingOrder, blocked, guard=
    none, update=none, synchronisation="attack?")
3 in automata PLC add transition(receivingSensorData, blocked,
    guard=none, update=none, synchronisation="attack?")
4 in automata PLC add transition(blocked, blocked, guard=none,
    update=none, synchronisation="rudder_setpoint?")
5 in automata PLC add transition(blocked, blocked, guard=none,
    update=none, synchronisation="stop_command?")
6 in automata PLC add transition(blocked, blocked, guard=none,
    update=none, synchronisation="supervision_command?")
7 in automata PLC add transition(blocked, blocked, guard=none,
    update=none, synchronisation="sensor_data?")
```

2.4 Conclusion

Si choisir une approche de modélisation pertinente pour des systèmes complexes et hétérogènes n'est pas chose aisée, nous avions toutefois la chance de disposer d'un ensemble de contraintes orientant notre réflexion. Ces contraintes, issues du processus décrit au chapitre 1 et reposant sur les modèles des systèmes qu'il a pour visée d'étudier, et liées et associées avec une littérature scientifique variée (tant en termes de granularité de modélisation que d'approche ou de formalisme) ont permis d'élaborer une démarche et un formalisme de modélisation répondant à nos objectifs.

Basé sur l'utilisation de réseaux d'automates finis temporisés étendus grâce aux apports d'UPPAAL [Lar+97], ce formalisme permet une modélisation intuitive, compréhensible par de nombreux acteurs de culture technique différente et suffisamment précise des objets étudiés (des missions aux liens topologiques, en passant par les processus, fonctions et composants). Il permet également la représentation des grandeurs physiques décrivant l'état du système en tant que système

dynamique : nous proposons à cet égard deux démarches de modélisation de ces grandeurs et de leur évolution. Enfin, par la mutation des modèles s'y conformant, il permet l'abstraction des vulnérabilités, attaques et contremesures affectant l'objet modélisé.

Le second objectif du chapitre qui se clôt était en effet la proposition d'une modélisation de ces trois types d'éléments. Nous y avons répondu par l'établissement et la catégorisation de *mutations* s'appliquant aux automates et réseaux d'automates, permettant de transcrire dans les modèles du système étudié la survenue de chacun de ces événements. Afin de faciliter leur prise en compte dans le cadre du processus proposé au chapitre 1, nous avons également établi un langage de description de ces mutations.

Ainsi, ce chapitre nous aura permis d'apporter un ensemble de réponses aux questions de recherche **QR2** et **QR5**. Les modèles proposés demeurent naturellement perfectibles ; des travaux futurs pourront notamment s'intéresser à diverses pistes d'enrichissement des modèles produits, par exemple par l'utilisation des plans du système (pour la prise en compte des risques liés à l'émission de signaux parasites par les composants, ou pour la modélisation de contremesures organisationnelles par restriction d'accès à une zone donnée pour éviter l'interaction avec un composant vulnérable, ...) ou par une reflexion plus poussée sur la modélisation de la confidentialité des données utilisées par le système. Ces enrichissements permettraient une expression d'autant plus fine de l'impact des vulnérabilités, attaques et contremesures en vertu de la méthodologie que le chapitre suivant s'attachera à exposer.

Chapitre 3

Méthodologie d'analyse d'impact

Table des matières

Au cœur du processus de gestion des correctifs de sécurité, l'analyse d'impact (tâche 2-2) s'appuie sur la modélisation comportementale que nous avons exposée au chapitre précédent. Le but de cette tâche critique est, pour une vulnérabilité donnée, de fournir un ensemble de mesures comparables entre elles, modélisant l'impact sur la capacité du système à accomplir ses missions :

— de la vulnérabilité elle-même ;
— d'une série d'attaques l'exploitant ;
— de correctifs la corrigeant ;
— pour chaque correctif, de la série d'attaques mentionnée précédemment sur le système «corrigé».

La comparaison de ces différentes mesures fournit pour chaque correctif deux indi-

cations essentielles : d'une part, la **non-régression** du système suite au déploiement de la contre-mesure ; d'autre part, l'**efficacité du correctif**.

L'analyse d'impact s'appuie sur différents *calculs d'impact* qui consistent, via une série d'opérations de *model-checking*, à vérifier la conformité du modèle issu de la phase de modélisation comportementale du système (tâche 1-3 du processus de gestion de correctifs) – et de ses mutations successives – aux propriétés établies lors de cette même phase. Par la suite, lorsque nous parlerons de calcul d'impact, cela désignera cet ensemble de vérifications menées sur le modèle afin de vérifier sa conformité à chacune des propriétés.

Dans ce chapitre [1], nous présenterons notre méthodologie d'analyse d'impact ; nous exposerons et motiverons également la métrique que nous avons conçue ; enfin, nous passerons en revue les différentes stratégies de limitation de l'explosion combinatoire, problématique récurrente des méthodologies reposant sur des techniques de *model-checking*.

3.1 Revue de littérature

L'estimation de l'impact d'événements cyber sur la capacité d'un système cyber-physique à accomplir ses missions a été traitée à plusieurs reprises, notamment par Sun Xiaoyan *et al.* [Sun+15], proposant une estimation de la propagation de l'impact d'une attaque affectant une ressource du système sur ses missions, à travers les tâches composant la mission et s'appuyant sur les ressources. Missions, ressources et tâches sont représentées par des graphes statiques liés entre eux par des relations de dépendance, ce qui permet d'affirmer qu'une mission peut être impactée par une attaque si une des ressources sur lesquelles elle s'appuie au travers de ses tâches est affectée. Une limite que nous voyons dans ce modèle réside dans le manque de précision dans l'expression des impacts : on sait si une mission, une tâche ou une ressource est affectée ; mais on ne sait pas comment.

Wael Kanoun *et al.* [Kan+07] proposent une méthode d'analyse du risque lié aux attaques et aux contremesures associées – les contremesures étant ici entendues en tant que moyens engagés pour faire cesser une attaque donnée. Le risque associé à une attaque est issu de la synthèse de la *potentialité* de sa survenue et

1. L'essentiel de ce chapitre, y compris la revue de littérature et les figures qui y sont présentées, a fait l'objet d'une publication dans [Sul+18] et reprend le contenu – traduit – de cette publication.

de son impact total (synthétisant ses impacts en termes de disponibilité, intégrité et confidentialité) ; celui associé à une contremesure est établi de la même manière à ceci près que sa survenue étant garantie, la potentialité de cette dernière est considérée comme étant maximale au cours du calcul du risque. Ces risques et impacts ne sont toutefois pas exprimés en fonction des missions du système et ne répondent donc pas entièrement à notre besoin.

Gustavo Gonzalez-Granadillo *et al.* [GG+17] proposent « une méthode pour calculer l'impact de cyber-attaques et de contremesures de sécurité » en s'appuyant sur un modèle géométrique. « Les attaques et les contremesures [sont représentés] dans un système de coordonnées à n dimensions », chacune de ces dimensions modélisant un élément du système étudié (e.g. un compte utilisateur, une ressource, un canal – c'est-à-dire le mode d'interaction liant un compte utilisateur et une ressource : lecture, écriture, connection, ... –, etc.) nécessaire à la conduite d'une attaque, ou affectée par une contremesure. Cette méthode permet d'exprimer les impacts d'une attaque sur le système, ainsi que l'efficacité des contremesures ou leurs potentiels dommages collatéraux. Toutefois, ces impacts ne sont pas exprimés en fonction des missions du système ; la jonction entre cette méthode d'analyse et le procédé de calcul de propagation proposé par Sun Xiaodan *et al.* [Sun+15] permettrait d'enrichir respectivement les deux méthodologies des éléments qui leur manquent au regard de notre objectif d'expression d'impact.

La méthode d'évaluation de l'impact des cyber-attaques sur les missions d'un système cyber-physique proposée par Scott Musman *et al.* [Mus+10 ; Mus+11] s'approche davantage de cet objectif. L'approche exposée consiste à suivre les étapes suivantes :

1. modéliser la mission à travers le standard BPMN (pour chaque mission, un nouveau modèle des ressources du système et de leurs interactions est généré) ;
2. exécuter le modèle et estimer la mesure d'efficacité (*measure of effectiveness* (MOE)) du système au regard des objectifs de la mission. Cette mesure est notée MOE1 ;
3. transformer le modèle initial de la mission pour prendre en compte les effets de la cyber-attaque étudiée sur la mission ;
4. exécuter à nouveau le modèle et calculer la mesure d'efficacité MOE2 : l'impact de la cyber-attaque est donné par la différence entre MOE1 et MOE2.

Notons que cette méthodologie pourrait également convenir pour l'analyse des contremesures. Toutefois, elle ne permet pas la modélisation des composants ne concourant pas directement à la mission étudiée, qui peuvent pourtant être utilisés comme vecteur pour propager une attaque aux ressources sur lesquelles s'appuie la mission.

Gabriel Jakobson développe dans le cadre des travaux que nous avons mentionnés précédemment [Jak11b] « une méthode pour estimer l'impact que des cyber-attaques peuvent avoir sur les composants, les services et les missions [d'un système cyber-physique]. » Comme nous l'avons vu dans la revue de littérature du chapitre 2, il propose une modélisation des composants logiciels, matériels et des services sous la forme de graphes dont les sommets représentent une entité et les arêtes les dépendances les liant. Chacun de ces sommets possède une capacité opérationnelle dynamique $OC \in [0, 1]$. Les attaques, quant à elles, sont modélisées à travers un facteur d'impact $IF \in [0, 1]$ « indiquant à quel point l'attaque est capable de dégrader le composant qu'elle affecte » ; la capacité opérationnelle de ce dernier diminuant à mesure que le facteur d'impact augmente ($OC_{t+1} = OC_t - IF$). Étant donné qu'un graphe de dépendances permettant d'illustrer les dépendances fonctionnelles unissant les missions aux services et aux composants logiciels et matériels est également fourni par le modèle proposé, l'impact d'une cyber-attaque sur une mission donnée est estimé à travers les trois étapes suivantes :

1. l'OC des composants attaqués est calculé en tenant compte de l'IF de l'attaque ;
2. l'OC des composants pouvant être affectés par la propagation de l'attaque est calculé de la même manière ;
3. l'OC de la mission est déduit des OC précédemment calculés (en le consolidant de manière incrémentale par une moyenne des OC des composants concourant à la mission ou à une tâche à travers un « ou » logique, ou en retenant le minimum des OC des composants concourant à la mission ou à une tâche à travers un « et », en partant du terrain des composants pour remonter jusqu'à celui des missions (*cf.* figure 2.3)).

Notre approche s'appuie également, comme nous l'avons vu dans le chapitre précédent, sur des modèles distincts pour représenter le système étudié, ses missions et leurs interactions, mais notre démarche de modélisation est différente dans la mesure où nous nous concentrons sur le comportement des composants. Une attaque, le déploiement d'une contremesure ou la découverte d'une vulnérabilité est

modélisée par une altération de l'Automate Système, qui nous permet d'exprimer avec une granularité plus fine l'impact de l'un de ces trois événements sur le système étudié, grâce à la vérification de propriétés de sûreté ou de sécurité modulables à l'envi, en fonction de la finesse d'analyse souhaitée.

3.2 Présentation générale de la méthode

La tâche 2-2 du processus de gestion de correctifs, présentée en section 1.4.2 du chapitre 1 et visant à fournir un ensemble de mesures d'impact liées à une vulnérabilité, est une application d'une méthodologie que nous avons précédemment exposée dans [Sul+18] et dont nous nous attacherons à présenter l'esprit à travers les pages qui vont suivre.

La méthodologie d'analyse d'impact sur laquelle nous nous appuyons est divisée en quatre tâches successives, comme l'illustre la figure 3.1. Chacune de ces tâches implique de dériver un modèle, ou un ensemble de modèles, à partir d'un modèle d'entrée et de la description d'un ensemble de mutations, puis de calculer une mesure d'impact, ou un ensemble de mesures d'impact relatives aux modèles ainsi dérivés.

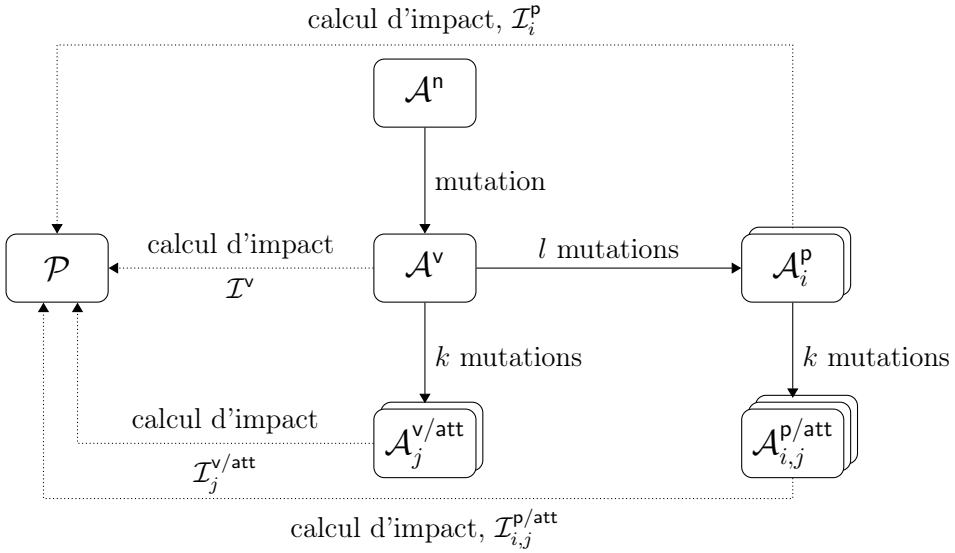
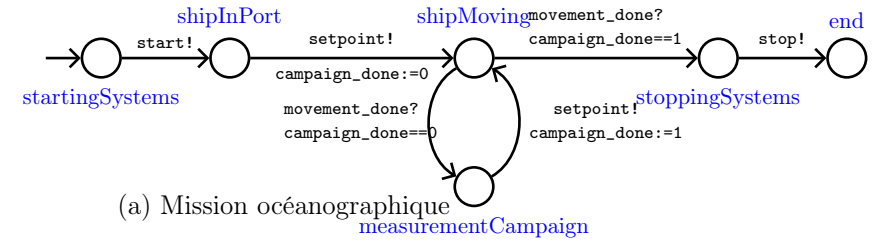
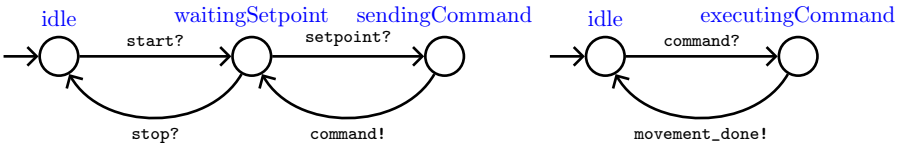


FIGURE 3.1 – Méthodologie d'analyse d'impact [Sul+18].

Nous nous attacherons à expliquer le principe de ces quatre tâches, en nous appuyant à chaque fois sur des exemples simples afin de les illustrer.



(a) Mission océanographique



(b) PLC contrôlant le gouvernail (c) Actionneur du gouvernail

FIGURE 3.2 – Trois automates simples [Sul+18].

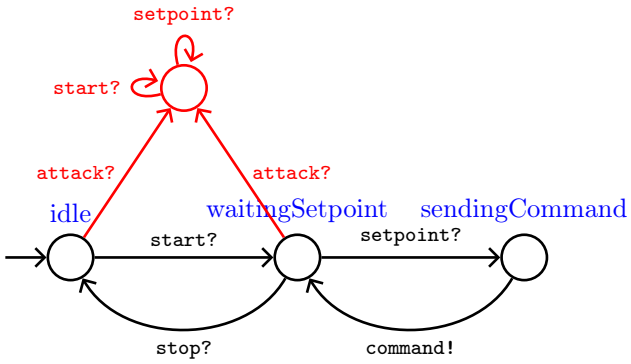


FIGURE 3.3 – PLC contrôlant le gouvernail, après application des mutations modélisant la vulnérabilité découverte [Sul+18].

Tâche 0. Avant la découverte d'une vulnérabilité, nous disposons d'une modélisation comportementale erronée de notre système. Cette modélisation, que nous nommons Automate Système Nominal, est notée $\mathcal{A}^{\mathsf{n}}$. $\mathcal{A}^{\mathsf{n}}$ vérifie – par définition – l'ensemble des propriétés $\mathcal{P}$.

Exemple : Considérons un navire océanographique. Nous nous concentrerons sur trois de ses composants : le safran, le PLC contrôlant le safran, et un pare-feu matériel filtrant les communications entre internet et le réseau du bord ; et sur deux missions : celle modélisée par l'automate représenté en figure 3.2(a) (mission 1), et une seconde mission, consistant à faire parvenir via internet à l'opérateur du bâtiment des rapports réguliers sur l'avancement de la campagne, et lui fournir en temps réel les données des

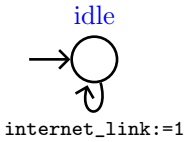


FIGURE 3.4 – Pare-feu.

différents capteurs (mission 2).

Le pare-feu matériel est modélisé par un automate simpliste qui initialise une variable *internet_ link* à 1 lorsqu'il autorise les échanges entre internet et le réseau du bord. Dans cet exemple, $\mathcal{A}^{\mathsf{n}}$ est donc la composition des automates représentés sur les figures 3.2(b) et (c), et de l'automate modélisant le pare-feu représenté en figure 3.4. $\mathcal{A}^{\mathsf{n}}$ vérifie l'ensemble de propriétés $\mathcal{P}$ suivant : {L'état `end` de l'automate mission 1 peut être atteint ; L'état `end` de l'automate mission 1 peut être atteint en x heures ; L'état `end` de l'automate mission 2 peut être atteint ; L'état `end` de l'automate mission 2 peut être atteint en x heures ; Le comportement du safran est conforme à sa spécification au cours de la mission 1 ; Le comportement du PLC est conforme à sa spécification au cours de la mission 1 ; Le comportement du pare-feu est conforme à sa spécification au cours de la mission 1 ; Le comportement du safran est conforme à sa spécification au cours de la mission 2 ; Le comportement du PLC est conforme à sa spécification au cours de la mission 2 ; Le comportement du pare-feu est conforme à sa spécification au cours de la mission 2 }.

Tâche 1. Suite à la découverte d'une vulnérabilité, le modèle $\mathcal{A}^{\mathsf{n}}$ devient erroné et il est nécessaire de le modifier afin de rendre compte des comportements induits par la vulnérabilité. Nous appliquons une série de mutations afin d'obtenir $\mathcal{A}^{\mathsf{v}}$, l'Automate Système Vulnérable modélisant le système vulnérable. Si $\mathcal{A}^{\mathsf{v}}$satisfait le test d'unicité présenté en section 1.4.2, un premier calcul d'impact est mené afin de vérifier si la simple présence de la vulnérabilité affecte la capacité du système à se comporter de manière conforme. Nous notons $\mathcal{I}^{\mathsf{v}}$ l'impact de la vulnérabilité.

Exemple : Imaginons que le PLC contrôlant le safran soit affecté d'une vulnérabilité le rendant vulnérable aux attaques de type « déni de service ». Nous appliquons à l'automate représenté en figure 3.2(b) les mutations nécessaires afin d'intégrer cette vulnérabilité, et obtenons l'automate représenté en figure 3.3. $\mathcal{A}^{\mathsf{v}}$est ici la composition de l'automate modélisant

le pare-feu matériel, ainsi que des automates représentés par les figures 3.3, 3.2(c). Dans le cas d'espèce, $\mathcal{I}^{\mathsf{v}}$ rendra compte que la seule présence de la vulnérabilité n'entrave pas la bonne marche du système.

Tâche 2. Pour chacune des l contre-mesures p_i ($i \in [\![1..l]\!]$) dont le déploiement est envisagé afin d'atténuer ou de supprimer la vulnérabilité, nous modélisons son effet sur le système vulnérable en mutant $\mathcal{A}^{\mathsf{v}}$ afin d'obtenir l'Automate Système Corrigé, $\mathcal{A}_i^{\mathsf{p}}$. Si $\mathcal{A}_i^{\mathsf{p}}$ satisfait le test d'unicité, un calcul d'impact est à nouveau mené : nous notons $\mathcal{I}_i^{\mathsf{p}}$ l'impact de la contre-mesure p_i. La comparaison entre $\mathcal{I}^{\mathsf{v}}$ et $\mathcal{I}_i^{\mathsf{p}}$ permet d'estimer la régression du système suite au déploiement de p_i.

Exemple : Soient deux contre-mesures disponibles : p_1, consistant en un correctif logiciel qui supprime la vulnérabilité sans effet collatéral sur le fonctionnement du PLC, et p_2, contre-mesure organisationnelle consistant à intervenir au niveau du pare-feu afin de couper les communications entre internet et le réseau du bord – et donc de supprimer le chemin d'attaque permettant à un attaquant d'exploiter la vulnérabilité. Nous notons $\mathcal{I}_1^{\mathsf{p}}$ et $\mathcal{I}_2^{\mathsf{p}}$ leurs impacts respectifs. $\mathcal{I}_1^{\mathsf{p}}$ indique qu'il n'y a aucune régression du système ; $\mathcal{I}_2^{\mathsf{p}}$ rend compte d'une perte de fonctionnalités sur traduisant par l'incapacité de réaliser la mission 2.

Tâches 3–4. L'étape suivante consiste à composer l'Automate Système Vulnérable et les l Automates Système Corrigés avec k automates d'attaque modélisant les k scénarios d'attaque redoutés exploitant la vulnérabilité que nous souhaitons corriger. Pour chaque attaque att_j ($j \in [\![1..k]\!]$), nous obtenons donc de la composition de l'automate $\mathcal{A}^{\mathsf{v}}$ avec l'automate modélisant la séquence d'attaque att_j l'Automate Système Vulnérable sous attaque $\mathcal{A}_j^{\mathsf{v}/\mathsf{att}}$; et de la composition de l'automate modélisant att_j avec les automates $\{\mathcal{A}_i^{\mathsf{p}}\}_{i\in[\![1..l]\!]}$, les Automates Système Corrigés sous attaque $\left\{\mathcal{A}_{i,j}^{\mathsf{p}/\mathsf{att}}\right\}_{i,j\in[\![1..l]\!]\times[\![1..k]\!]}$.

Deux calculs d'impact successifs sont menés pour chaque couple $(\mathsf{p}_i, \mathsf{att}_j)$ si $\mathcal{A}_j^{\mathsf{v}/\mathsf{att}}$ et $\mathcal{A}_{i,j}^{\mathsf{p}/\mathsf{att}}$ satisfont le test d'unicité : l'impact de l'attaque att_j sur le système vulnérable est noté $\mathcal{I}_j^{\mathsf{v}/\mathsf{att}}$, et son impact sur le système où la contre-mesure p_i est déployée est noté $\mathcal{I}_{i,j}^{\mathsf{p}/\mathsf{att}}$. La comparaison entre $\left\{\mathcal{I}_j^{\mathsf{v}/\mathsf{att}}\right\}_{j\in[\![1..k]\!]}$ et $\left\{\mathcal{I}_{i,j}^{\mathsf{p}/\mathsf{att}}\right\}_{j\in[\![1..l]\!]\times[\![1..k]\!]}$ permet d'estimer l'efficacité de la contre-mesure p_i.

Exemple :

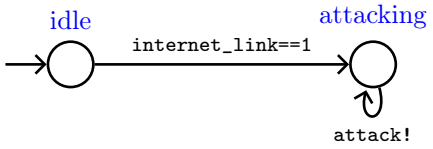


FIGURE 3.5 – Un automate d'attaque [Sul+18].

Considérons une attaque très simple, consistant à exploiter la vulnérabilité du PLC afin de le rendre inopérant par un déni de service mené par un attaquant situé en-dehors du navire, et accédant donc au réseau du bord via son lien avec internet. Cette attaque att_1 est modélisée par l'automate représenté en figure 3.5.

Dans un premier temps, nous composons l'automate d'attaque avec $\mathcal{A}^{\mathsf{v}}$ afin d'obtenir $\mathcal{A}_1^{\mathsf{v/att}}$. Nous menons un calcul d'impact, dont le résultat est noté $\mathcal{I}_1^{\mathsf{v/att}}$. $\mathcal{I}_1^{\mathsf{v/att}}$ rend compte du fait que l'état `end` de l'automate mission 1 ne peut être atteint – par conséquent qu'il ne peut être atteint dans les délais – et que le comportement du PLC n'est pas conforme à sa spécification dans le contexte de la mission 1. Ensuite, nous composons l'automate d'attaque avec $\mathcal{A}_1^{\mathsf{p}}$ et $\mathcal{A}_2^{\mathsf{p}}$. De cette composition découlent $\mathcal{A}_{1,1}^{\mathsf{p/att}}$ et $\mathcal{A}_{2,1}^{\mathsf{p/att}}$. Nous menons deux calculs d'impact afin d'obtenir $\mathcal{I}_{1,1}^{\mathsf{p/att}}$ et $\mathcal{I}_{2,1}^{\mathsf{p/att}}$: ces deux mesures rendent compte, respectivement, que le modèle $\mathcal{A}_{1,1}^{\mathsf{p/att}}$ réagit conformément aux propriétés $\mathcal{P}$; et que $\mathcal{A}_{2,1}^{\mathsf{p/att}}$ est insensible à l'attaque mais ne vérifie pas la propriété « l'état `end` de l'automate mission 2 peut être atteint », et par conséquent qu'il ne vérifie pas non plus que « l'état `end` de l'automate mission 2 peut être atteint en x heures ».

L'application de cette méthodologie d'analyse d'impact permet ainsi de disposer de mesures autorisant l'estimation de la non-régression et de l'efficacité de chaque contremesure pour une vulnérabilité donnée. Nous allons exposer, dans la section suivante, comment formellement exprimer ces mesures.

3.3 Une métrique pour quantifier et comparer les impacts

3.3.1 Le calcul d'impact en bref

Les opérations de calcul d'impact auxquelles nous faisons référence dans la section précédente consistent à vérifier, pour l'ensemble de ses missions, qu'un modèle donné (Automate Système Nominal $\mathcal{A}^{\mathsf{n}}$, Automate Système Vulnérable $\mathcal{A}^{\mathsf{v}}$, Automate Système Corrigé $\mathcal{A}_i^{\mathsf{p}}$, Automate Système Vulnérable sous attaque $\mathcal{A}_j^{\mathsf{v/att}}$ ou Automate Système Corrigé sous attaque $\mathcal{A}_{i,j}^{\mathsf{p/att}}$) se conforme aux propriétés $\mathcal{P}$; cette vérification étant faite à travers une série d'opérations de *model checking*.

Rappel (composition de $\mathcal{P}$)

$\mathcal{P}$ est constitué de propriétés permettant d'exprimer différentes exigences de sûreté de fonctionnement ou de sécurité. Dans le cadre du processus exposé au chapitre 1, nous avons choisi de traduire les exigences suivantes (*cf.* chapitre 1, section 1.3.2) :

- l'atteignabilité de l'état final de chaque mission ;
- l'atteignabilité de l'état final de chaque mission, dans des délais précis ;
- la conformité du comportement de chaque composant à leur spécification dans le cadre de chaque mission ;
- l'intégrité de chaque composant ;
- la confidentialité des données traitées par le système.

L'expression de ces propriétés est soit dépendante des automates mission (pour les trois premiers groupes), soit indépendante (pour les deux derniers). Pour n missions, $\mathcal{P}$ est donc composé des $n+1$ sous-ensembles $P_1, \ldots, P_n, P_{ind}$ où $P_{i,i\in[\![1,n]\!]}$ est l'ensemble des propriétés dont l'expression dépend de la mission i, et P_{ind} est l'ensemble des propriétés dont l'expression est indépendante des missions.

Chaque calcul d'impact se déroule de la manière suivante : pour chaque mission i, nous composons l'Automate Système vérifié avec l'Automate Mission afférent ; puis nous vérifions une-à-une les propriétés spécifiquement liées à l'automate mission P_i ainsi que les propriétés d'expression constante P_{ind}. Si nous considérons que un système constitué de m composants, réalisant n missions et traitant

p données dont nous souhaitons garantir la confidentialité, nous obtenons donc à l'issue du calcul d'impact un ensemble d'au plus $n \times (p + 2m + 2)$ valeurs binaires, en fonction des propriétés retenues lors de la phase de modélisation du système.

3.3.2 Présentation de la métrique

L'enjeu est dès lors de construire une métrique [2] permettant de rendre compte de l'information apportée par la somme de ces valeurs, dans trois visées :

1. disposer d'une expression de l'information lisible ;

2. disposer d'une expression de l'information interprétable de manière fine : nous excluons donc une expression d'impact sous forme d'un unique nombre réel ;

3. disposer de mesures comparables, *i.e.* sur l'ensemble desquelles nous pouvons définir une relation d'ordre.

La meilleure solution afin de répondre aux exigences 1 et 2 paraît être de présenter les valeurs issues du calcul d'impact sous forme de matrice, appelée *matrice d'impact* $\mathcal{I}$; chaque colonne de cette matrice correspond à une propriété, et chacune de ses lignes correspond à une mission [3]. Afin de répondre à l'exigence de comparabilité, les lignes sont triées entre elles en fonction de la criticité de la

2. Rappelons que cette métrique fonde la prise de décision de la tâche 2.3 du processus proposé au chapitre 1, conformément au déroulé de cette tâche présenté en section 1.4.3.

3. NB : la métrique ici proposée est différente de la métrique proposée dans [Sul+18]. En effet, la matrice d'impact est transposée : ce sont ici les lignes, et non plus les colonnes, qui correspondent aux missions. Par ailleurs, elle intègre désormais l'évaluation des propriétés ayant trait (1) à la conformité du comportement de chaque composant à leur spécification dans le cadre de chaque mission, et (2) à la confidentialité des données traitées par le système.

mission, de la plus critique à la moins critique :

$$\mathcal{I}^\top = \begin{pmatrix} A_0 & A_1 & \cdots & A_{n-2} & A_{n-1} \\ T_0 & T_1 & \cdots & T_{n-2} & T_{n-1} \\ C_{0,0} & C_{0,1} & \cdots & C_{0,n-2} & C_{0,n-1} \\ C_{1,0} & C_{1,1} & \cdots & C_{1,n-2} & C_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ C_{m-1,0} & C_{m-1,1} & \cdots & C_{m-1,n-2} & C_{m-1,n-1} \\ I_{0,0} & I_{0,1} & \cdots & I_{0,n-2} & I_{0,n-1} \\ I_{1,0} & I_{1,1} & \cdots & I_{1,n-2} & I_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{m-1,0} & I_{m-1,1} & \cdots & I_{m-1,n-2} & I_{m-1,n-1} \\ Pr_{0,0} & Pr_{0,1} & \cdots & Pr_{0,n-2} & Pr_{0,n-1} \\ Pr_{1,0} & Pr_{1,1} & \cdots & Pr_{1,n-2} & Pr_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Pr_{p-1,0} & Pr_{p-1,1} & \cdots & Pr_{p-1,n-2} & Pr_{p-1,n-1} \end{pmatrix}$$

NB : pour des raisons de lisibilité, la matrice précédente, $\mathcal{I}$, a été transposée.

Les coefficients de cette matrice sont binaires, valant 1 si la propriété à laquelle ils correspondent est vérifiée, et 0 sinon. Chaque A_k correspond à la propriété d'atteignabilité de l'état final de l'Automate Mission k, T_k à la propriété d'atteignabilité de cet état dans les délais fixés, $C_{j,k}$ à la propriété faisant état de la conformité du comportement du composant j dans le cadre de la mission k, $I_{j,k}$ à la propriété faisant état de la préservation de l'intégrité du composant j au cours de l'exécution de la mission k, et $Pr_{l,k}$ à la propriété faisant état de la préservation de la confidentialité de la donnée l au cours de l'exécution de la mission k par le système.

Toujours à des fins de comparabilité, il est nécessaire de définir un ordre de criticité sur l'ensemble des propriétés. La situation est un peu plus complexe dans la mesure où l'importance d'une propriété donnée dépend du contexte de chaque mission : par exemple, l'intégrité d'un sonar bathymétrique peut être plus importante que l'intégrité d'un thermosalinographe dans le contexte d'une campagne de mesures bathymétriques, alors que l'intégrité du thermosalinographe primera dans le contexte d'une campagne de mesures de la salinité de l'océan. Cela revient donc à ordonner les valeurs entre elles, ligne par ligne. Nous définissons donc une permutation σ sur les éléments de chaque ligne. Cette permutation ordonne ces

éléments en fonction de l'importance accordée à leur propriété afférente, de la plus importante à la moins importante. Nous obtenons à l'issue de l'application de cette permutation la *matrice d'impact filtrée* $\mathcal{I}_f$:

$$\mathcal{I}_f \begin{pmatrix} \sigma(\mathcal{L}_0) \\ \sigma(\mathcal{L}_1) \\ \vdots \\ \sigma(\mathcal{L}_{n-1}) \end{pmatrix}$$

Où $\mathcal{L}_{0\cdots n}$ sont les colonnes de la matrice d'impact. Les mesures d'impact fournies à l'issue des opérations d'analyse d'impact sont des matrices d'impact filtrées.

Exemple : considérons la matrice d'impact suivante, afférente à un système dont trois composants et quatre missions sont modélisées, et pour lequel nous cherchons à nous assurer de l'atteignabilité (dans l'absolu et dans les délais) de l'état final de ses missions ainsi que de l'intégrité de ses composants :

$$\begin{pmatrix} A_0 & T_0 & I_{0,0} & I_{1,0} & I_{2,0} \\ A_1 & T_1 & I_{0,1} & I_{1,1} & I_{2,1} \\ A_2 & T_2 & I_{0,2} & I_{1,2} & I_{2,2} \\ A_3 & T_3 & I_{0,3} & I_{1,3} & I_{2,3} \end{pmatrix}$$

Alors, la matrice d'impact filtrée peut prendre cette forme :

$$\begin{pmatrix} A_0 & T_0 & I_{2,0} & I_{1,0} & I_{0,0} \\ A_1 & I_{0,1} & I_{2,1} & T_1 & I_{1,1} \\ A_2 & T_2 & I_{0,2} & I_{1,2} & I_{2,2} \\ A_3 & I_{0,3} & I_{1,3} & I_{2,3} & T_3 \end{pmatrix}$$

Les lignes et les colonnes de la matrice d'impact filtrée ainsi définie étant triées par ordre décroissant, nous pouvons aisément ordonner différentes matrices d'impact filtrées selon un ordre lexicographique, adapté au contexte, que nous définissons ci-dessous.

Définition 7 (**Ordre lexicographique**). Soient $(a_{i,j}), (b_{i,j}) \in \{0,1\}^{m\times n}$.
$(a_{i,j}) <_{lex} (b_{i,j}) \iff \exists (k,l) \in [\![0,m]\!]\times[\![0,n]\!] \mid (\forall p,q \in [\![0,m]\!]\times[\![0,l-1]\!], a_{p,q} = b_{p,q})$
$\wedge (a_{0,l} = b_{0,l} \wedge \cdots \wedge a_{k-1,l} = b_{k-1,l}) \wedge (a_{k,l} < b_{k,l})$.

3.3.3 Exemple

Considérons à nouveau le navire océanographique, ses deux missions, sa vulnérabilité, l'attaque l'exploitant ainsi que les deux contre-mesres pris en exemple en section 3.2 du présent chapitre. Considérons également que les matrices d'impact filtrées feront état, dans cet ordre, des résultats des vérifications des propriétés suivantes :

Mission 1	L'état `end` de l'automate mission 1 peut être atteint	L'état `end` de l'automate mission 1 peut être atteint en x heures	Le comportement du safran est conforme à sa spécification au cours de la mission 1	Le comportement du PLC est conforme à sa spécification au cours de la mission 1	Le comportement du pare-feu est conforme à sa spécification au cours de la mission 1
Mission 2	L'état `end` de l'automate mission 2 peut être atteint	L'état `end` de l'automate mission 2 peut être atteint en x heures	Le comportement du pare-feu est conforme à sa spécification au cours de la mission 2	Le comportement du PLC est conforme à sa spécification au cours de la mission 2	Le comportement du safran est conforme à sa spécification au cours de la mission 2

Alors, les impacts $\mathcal{I}^{\mathsf{v}}$, $\mathcal{I}^{\mathsf{p}}_1$, $\mathcal{I}^{\mathsf{p}}_2$, $\mathcal{I}^{\mathsf{v/att}}_1$, $\mathcal{I}^{\mathsf{p/att}}_{1,1}$ et $\mathcal{I}^{\mathsf{p/att}}_{2,1}$ seront représentés par les matrices suivantes :

$$\mathcal{I}^{\mathsf{v}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \mathcal{I}^{\mathsf{p}}_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \mathcal{I}^{\mathsf{p}}_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathcal{I}^{\mathsf{v/att}}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \mathcal{I}^{\mathsf{p/att}}_{1,1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \mathcal{I}^{\mathsf{p/att}}_{2,1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

En comparant ces matrices selon notre ordre lexicographique, nous pouvons constater que $\mathcal{I}^{\mathsf{p}}_1 =_{lex} \mathcal{I}^{\mathsf{v}}$ et $\mathcal{I}^{\mathsf{p}}_2 <_{lex} \mathcal{I}^{\mathsf{v}}$. Cela signifie que la contre-mesure p_1

n'induira vraisemblablement aucune régression sur le système, contrairement à la contremesure p_2 qui l'ampute de sa capacité à accomplir sa seconde mission. Par ailleurs, nous remarquons que $\mathcal{I}_{1,1}^{\mathsf{p}/\mathsf{att}} >_{lex} \mathcal{I}_1^{\mathsf{v}/\mathsf{att}}$ et que $\mathcal{I}_{2,1}^{\mathsf{p}/\mathsf{att}} >_{lex} \mathcal{I}_1^{\mathsf{v}/\mathsf{att}}$: p_1 comme p_2 permettent au système d'avoir une meilleure résilience face à l'attaque att_1. Toutefois, $\mathcal{I}_{1,1}^{\mathsf{p}/\mathsf{att}} >_{lex} \mathcal{I}_{2,1}^{\mathsf{p}/\mathsf{att}}$: nous en concluons que p_1 est plus efficace que p_2.

3.4 Stratégies de limitation de l'explosion combinatoire

Une des limitations notoires de la vérification de modèles est l'explosion combinatoire pouvant d'autant plus aisément survenir dès lors que croît la complexité des systèmes modélisés [Dha+11 ; Cla09 ; Cla+01] : étant donné le nombre et la diversité des sous-systèmes, composants et processus d'un navire moderne, il est nécessaire de prendre ce problème en considération.

Nous pouvons considérer que le coût calculatoire de notre méthode d'analyse d'impact dépend de deux paramètres : le nombre d'opérations de *model-checking* à mener (i.e. le nombre de propriétés à vérifier), et la taille des automates vérifiés. Nous pouvons donner une majoration du premier de ces deux paramètres : considérant comme précédemment un système constitué de m composants, accomplissant n missions, et traitant p données dont nous souhaitons vérifier la confidentialité. Comme nous l'avons vu précédemment, l'établissement de chaque matrice d'impact mène au plus à $n \times (p + 2m + 2)$ vérifications du modèle. Or, d'après la méthode exposée en section 3.2 (*cf.* figure 3.1), chaque vulnérabilité implique jusqu'à $(1+l+k+k\times l)$ calculs d'impact, l étant le nombre de contremesures tempérant la vulnérabilité et k le nombre d'attaques potentielles l'exploitant. Par conséquent, la découverte d'une vulnérabilité entraîne au plus $n\times(p+2m+2)\times(1+l+k+k\times l)$ vérifications de modèles. Si nous considérons qu'un système naval type est constitué de 2000 composants traitant chacun deux types de données dont la confidentialité doit être assurée (ses mots de passe et ses données « métier »), effectue 20 missions différentes et que pour chaque vulnérabilité 5 attaques et 5 contremesures relatives existent, alors la découverte d'une vulnérabilité entraînera jusqu'à 5 761 440 vérifications de modèles.

La taille du modèle vérifié (issu de la composition de l'Automate Système vérifié avec un Automate Mission) dépend du système modélisé et de ses missions. Dans le cadre des travaux présentés dans ces pages, plusieurs systèmes ont été mo-

délisés afin d'éprouver les limites de la méthode ; nous en donnerons un exemple significatif dans le chapitre suivant, dérivant d'un premier cas d'étude fictif pouvant servir de base à l'estimation des dimensions des modèles vérifiés. Nous avons en effet modélisé pour ces besoins deux sous-systèmes d'un bâtiment océanographique fictif : un système de propulion/gouverne (constitué d'une console fournissant une interface homme-machine, de trois PLC, d'un boîtier inverseur/réducteur, d'un moteur Diesel et d'un gouvernail) et un système de relevés bathymétriques (composé d'un ordinateur fournissant une interface homme-machine, d'un système d'acquisition de données et d'un sondeur multifaisceaux).

Ce cas d'étude a été modélisé à l'aide du langage FIACRE [Ber+08] et compilé et exécuté avec OBP [Dha+11] (permettant, contrairement à UPPAAL, d'obtenir des informations sur l'automate composé). Nous avons par ailleurs modélisé deux Automates Mission : un premier proche de celui présenté en figure 3.2(a), utilisant les deux sous-systèmes du bâtiment, et un second utilisant uniquement le sous-système de propulsion/gouverne. À la composition de l'Automate Système avec chacun de ces Automates Mission, nous pouvons remarquer que la complexité de l'automate résultant est variable : à la composition avec l'Automate Mission 1, OBP calcule un automate composé possédant 39 069 états de 144 756 transitions, tandis que le modèle résultant de la composition avec l'Automate Mission 2 possède seulement 4 095 états et 11 380 transitions [Sul+18].

Si la vérification des propriétés sur un modèle de cette dimension, et sur les autres modèles que nous avons étudiés dans le cadre de nos travaux, ne nous a pas confrontés à des problèmes d'explosion combinatoire, nous avons toutefois pu constater des temps de calcul élevés (nous y reviendrons dans le chapitre suivant) et nous pouvons dessiner les pistes suivantes afin de réduire ces temps de calcul ou de prévenir l'apparition de phénomènes d'explosion de l'espace d'états :

1. Calcul haute-performance. Mener les opérations de vérifications de modèle à l'aide de supercalculateurs et d'infrastructures de calcul fortement parallélisées peut permettre une amélioration des temps de calcul. Cette solution a le désavantage d'un coût élevé en ingénierie, contrairement aux quatre suivantes ;

2. Analyse d'impact par évaluation paresseuse. Étant donné que l'analyse d'impact, dans le contexte du processus proposé au chapitre 1 est de comparer les impacts de différentes contremesures, attaques, vulnérabilités entre eux, les matrices d'impact peuvent être calculées en parallèle (propriété par propriété, mission par mission). Aussitôt que l'une des ma-

trices « devient » inférieure aux autres au sens de l'ordre lexicographique formalisé par la définition 7, elle est éliminée du processus d'analyse ;

3. Calcul partiel. En complément de l'évaluation paresseuse, nous pouvons imaginer une restriction de l'ensemble des missions étudiées aux plus critiques, réduisant ainsi le nombre de lignes des matrices d'impact (et donc de vérifications de modèle à mener) ;
4. Réduction de l'espace d'état par analyse des contextes. Un degré d'optimisation supplémentaire peut être atteint par l'élimination de l'Automate Système des automates modélisant les composants ne concourant pas à l'exécution de la mission étudiée, lorsque le cloisonnement des systèmes le permet et que la certitude est acquise que ces composants ne peuvent pas être utilisés afin d'atteindre ceux sur lesquels la mission étudiée s'appuie ;
5. Enfin, concernant les propriétés de confidentialité des données, il peut être plus efficient de vérifier uniquement la « confidentialité » du composant, sans différencier les données exfiltrées de celles demeurant protégées ; cela permet ainsi de borner p à m et par conséquent de faire chuter le nombre de propriétés à vérifier.

Notons par ailleurs que les tests d'unicité menés avant chaque calcul d'impact permettent, à mesure que le processus s'applique et s'enrichit des calculs passés, une réduction croissante du nombre de propriétés vérifiées.

3.5 Conclusion

Nous posions, dans l'introduction générale, les deux questions de recherche suivantes : *comment exprimer l'impact d'une vulnérabilité, de l'exploitation d'une vulnérabilité, d'un correctif, sous forme de mesure comparable contenant suffisamment d'information ?* et *comment mener les calculs aboutissant à ces mesures ?* Étant donné que le résultat de ces calculs fonde la prise de décision au sein du processus que nous proposons au chapitre 1 et que leur conduite en est l'opération décisive, au cœur de la tâche 2-2, les réponses apportées à ces questions ont une implication forte sur la méthodologie qui les appelle.

Afin de répondre à la seconde question, nous avons proposé dans ce chapitre une méthode d'analyse d'impact en quatre temps, construite pour fournir des mesures comparables visant à permettre l'estimation de deux paramètres clé – la non-régression d'une contremesure et son efficacité. Ces paramètres, pour être

estimés, nécessitent plusieurs calculs d'impact rendant compte des effets de la présence d'une vulnérabilité, de la conduite d'une attaque ou du déploiement d'une contremesure.

Ces calculs, pour être exploitables, nécessitent d'être synthétisés par une mesure illustrative et comparable servant d'appui efficace à la décision. Par conséquent, nous avons proposé dans ces pages une métrique matricielle permettant d'ordonner entre elles les propriétés de sûreté fondant le calcul d'impact, en fonction de leur criticité. Subsidiairement, nous avons également défini un ordre facilitant la comparaison automatique de ces différentes mesures entre elles, d'autant plus nécessaire à mesure que croît le nombre de propriétés vérifiées.

Ainsi, le chapitre qui se clôt aura permis d'apporter un ensemble de réponses aux questions de recherche **QR3** et **QR4**, que nous citions en début de conclusion. Il nous aura également permis de poser les bases d'une réflexion sur les limites, notamment en termes de temps de calcul, de la méthode d'analyse d'impact proposée. Cette réflexion ainsi qu'une interrogation plus large sur la pertinence de la métrique proposée seront développées au chapitre suivant, qui synthétisera un retour expérimental sur les modèles conçus dans le cadre de nos travaux.

Chapitre 4

Synthèse des expérimentations

Table des matières

Les chapitres précédents nous ont permis d'introduire et de présenter plusieurs notions et outils qui ont été élaborés au cours de nos travaux de recherche. Ainsi, nous avons pu exposer le principe de notre processus de gestion de correctifs et, en détaillant les méthodologies retenues ou conçues pour chacune de ses phases, introduire une méthodologie de modélisation et un outil de génération de code permettant de l'automatiser partiellement, présenter un outil de veille permettant d'identifier en temps réel les vulnérabilités affectant chaque équipement et chaque fonction d'un système complexe, exposer une méthodologie et une métrique d'analyse d'impact et formuler des consignes de déploiement de contre-mesures et de maintien à jour des connaissances. Nous avons, à plusieurs reprises, utilisé des exemples très simples afin d'illustrer ces différents apports.

Toutefois, la validation de chaque notion, de chaque outil, de chaque phase du processus de gestion de correctifs – puis, *in fine*, du processus dans son ensemble – a été menée en nous appuyant sur des systèmes de test, conçus *ad-hoc* ou existant.

Le chapitre qui va suivre aura pour objet de présenter un des cas de test fictifs ayant servi à ces fins, de lui appliquer une itération du processus décrit au chapitre 1 pour l'illustrer par un exemple plus conséquent que ceux utilisés précédemment, et enfin d'exposer les conclusions que nous pouvons tirer de la mise à l'épreuve de nos propositions sur ce cas pratique.

4.1 Description du système de test fictif

Afin de valider les différentes tâches du processus – hors tâches nécessitant un déploiement sur un système réel –, nous avons conçu un cas d'étude inspiré de différentes classes de bâtiments civils, à partir de données publiques (documentations de fournisseurs de solutions d'automatisation pour les systèmes navals [NOR], description de l'appareil propulsif d'un paquebot de ligne [Ele04] ou de ferries utilisant les propulsions CODOG [Soa14], documents de spécification d'un simulateur de propulsion navale [Nyl16], description de l'architecture réseau d'un système de contrôle et de commande d'un navire [Fen+15], etc.).

Ce cas d'étude consiste en deux sous-systèmes – un sous-système de propulsion et un sous-système de gouverne – qui concourent aux chaînes fonctionnelles

éponymes, parmi les plus critiques d'un navire.

4.1.1 Sous-système de gouverne – analyse fonctionnelle

Le sous-système de gouverne vise à permettre l'orientation du navire. Il offre à ses opérateurs trois fonctions principales : premièrement, le sous-sytème permet d'établir une consigne d'angle de barre par le biais d'une IHM ; ensuite, il exécute la consigne fixée par les opérateurs en s'appuyant sur un PLC régulant l'angle du safran, à travers une boucle de rétroaction alimentée par des capteurs mesurant cet angle (*cf.* figure 4.1) ; enfin, il rend compte aux opérateurs de l'angle de barre réalisé en restituant sur l'IHM les informations fournies par les capteurs.

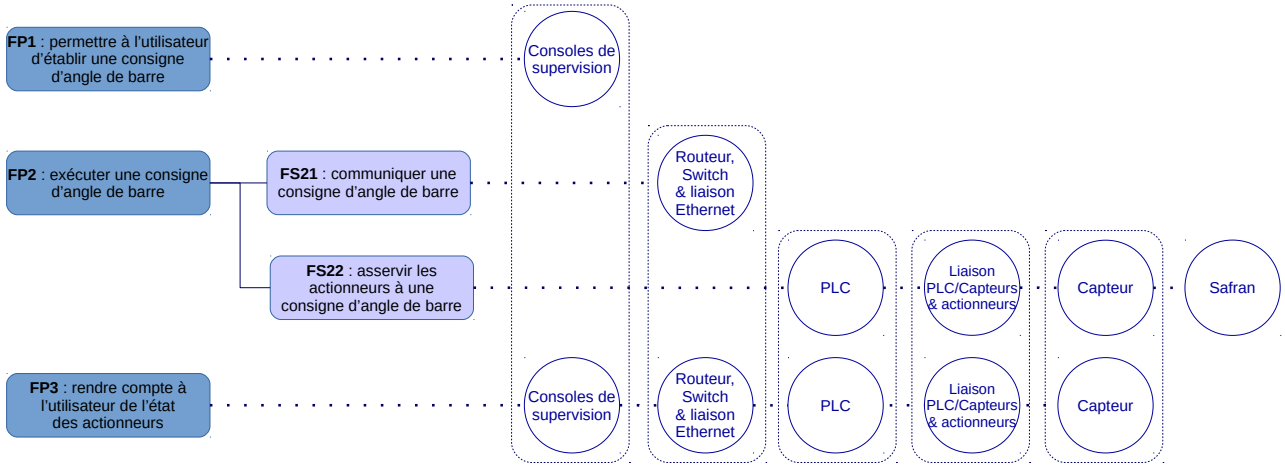


FIGURE 4.1 – Analyse fonctionnelle du sous-système de gouverne

4.1.2 Sous-système de propulsion – analyse fonctionnelle

Le sous-système de propulsion fournit l'énergie mécanique nécessaire au déplacement du navire dans son environnement et offre aux opérateurs la possibilité d'asservir la vitesse de rotation de l'arbre moteur, à travers celle des différents moteurs et les rapports d'inversion/réduction. Il est typiquement constitué :

— d'une console de supervision (IHM) et de PLC permettant d'asservir les moteurs et le boîtier inverseur/réducteur aux consignes fixées par les opérateurs via l'IHM ;

— d'un ou plusieurs moteurs ;
— d'un boîtier inverseur/réducteur permettant de transmettre aux lignes d'arbre l'énergie produite par les moteurs ;
— de capteurs permettant d'assurer les boucles de rétroaction pour l'asservissement de la vitesse de rotation des moteurs, et de permettre la supervision des moteurs par les opérateurs ;
— d'une ou plusieurs lignes d'arbre aboutissant à une ou plusieurs hélices.

La propulsion navale motorisée repose classiquement sur trois grandes technologies : le moteur Diesel, la turbine à gaz et la propulsion nucléaire. Plusieurs bâtiments civils ou militaires mettent également en oeuvre une propulsion mixte associant turbine à gaz et moteur Diesel deux temps – l'intérêt de la première étant de permettre des accélérations vives et une vitesse élevée ; celui du second étant de propulser le navire en régime de croisière tout en offrant un rendement supérieur à celui de la turbine à gaz. L'énergie mécanique produite par les moteurs est ensuite transmise aux lignes d'arbre via un boîtier inverseur/réducteur (« l'embrayage » du navire) qui a trois visées : d'accoupler et désaccoupler les lignes d'arbres des moteurs ; de transmettre, à travers un système d'arbres et d'engrenages, l'énergie aux lignes d'arbres en leur assurant une vitesse de rotation réduite par rapport à la vitesse de rotation des arbres moteurs ; et enfin, d'inverser le sens de rotation des lignes d'arbre pour permettre la « marche arrière » du navire.

Le sous-système de propulsion que nous avons retenu pour notre cas d'étude est du type CODOG (pour *COmbined Diesel Or Gas*), associant une turbine à gaz et deux moteurs Diesel [Soa14]. La décomposition fonctionnelle et les solutions techniques retenues sont synthétisées en figure 4.2.

4.1.3 Architecture physique

Les différents composants techniques des deux sous-systèmes, identifiés au cours de l'analyse fonctionnelle, sont intégrés au sein du système selon l'architecture représentée en figure 4.3.

Au sein du sous-système de gouverne, les composants suivants sont utilisés :
— Un safran ;
— Des capteurs permettant de mesurer l'angle de barre réalisé par le safran ;
— Un PLC permettant de « traduire » les consignes des opérateurs et d'asservir le safran à partir des consignes et des retours des capteurs ;
— Une console de supervision dédiée, permettant aux opérateurs de com-

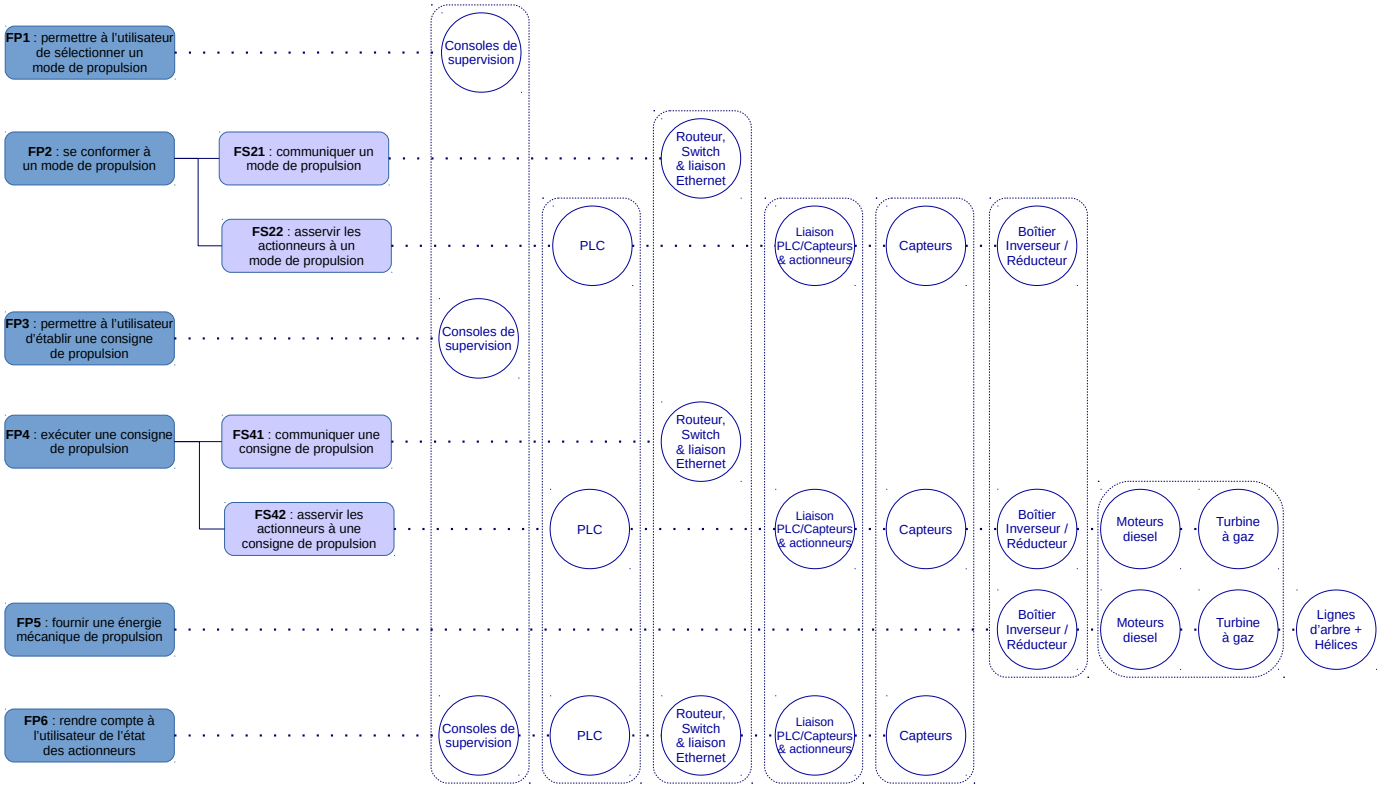


FIGURE 4.2 – Analyse fonctionnelle du sous-système de propulsion

mander et de contrôler le sous-système de gouverne ;

— Un commutateur réseau et une liaison Ethernet permettant la communication entre le PLC, la console dédiée et le reste du système.

Le sous-système de propulsion repose, quant à lui, sur les composants suivants :

— Deux moteurs Diesel ;
— Une turbine à gaz ;
— Un boîtier inverseur/réducteur ;
— Des capteurs permettant de mesurer la vitesse de rotation des différents moteurs et de rendre compte de l'état du boîtier inverseur/réducteur ;
— Trois PLC permettant de « traduire » les consignes des opérateurs et d'asservir les moteurs Diesel (resp. la turbine à gaz, le boîtier inverseur/-réducteur) à partir des consignes et des retours des capteurs ;
— Trois consoles de supervision dédiées, permettant aux opérateurs de commander et de contrôler les moteurs Diesel (resp. la turbine à gaz, le boîtier

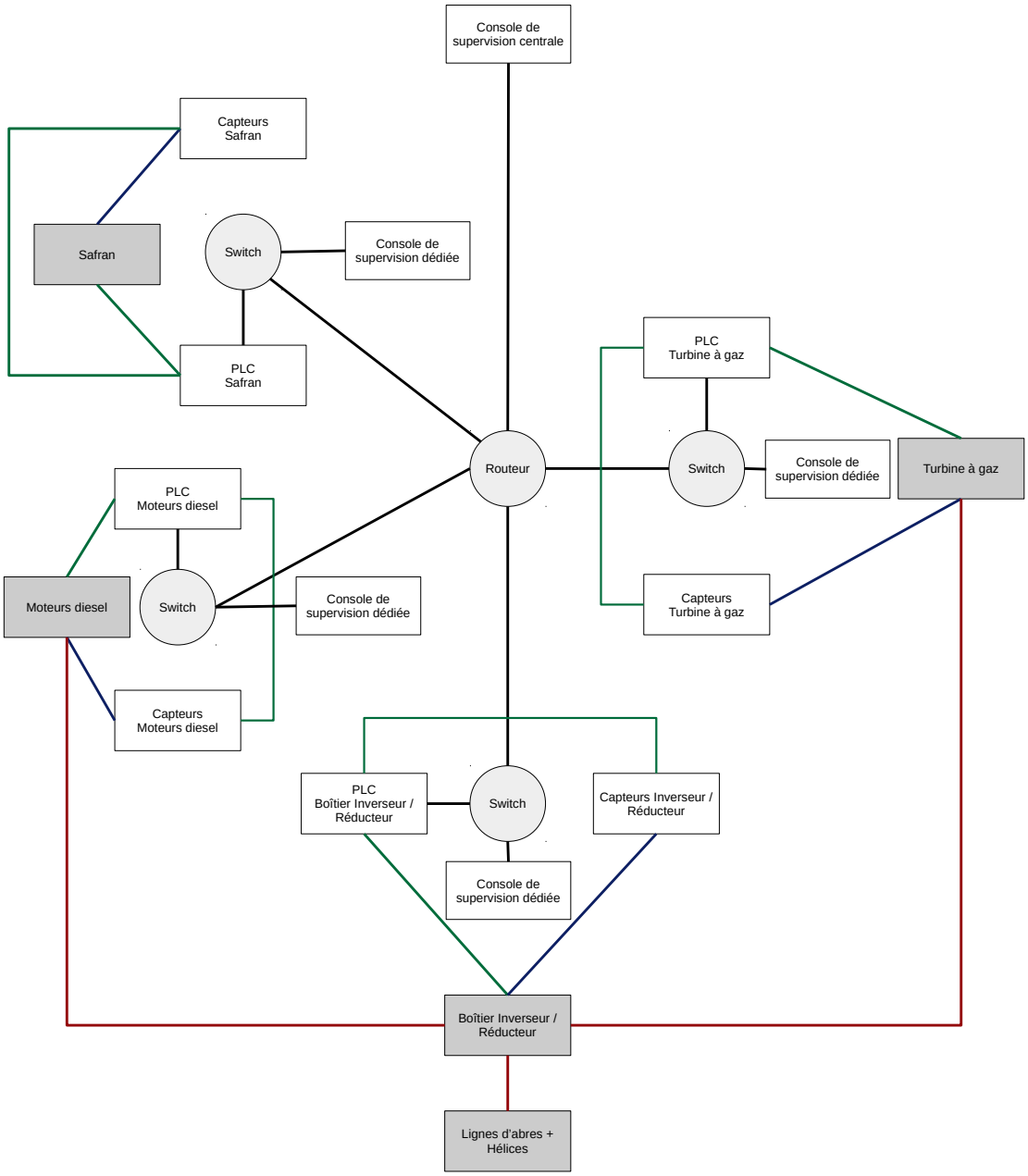


FIGURE 4.3 – Architecture physique du système. Les liaisons Ethernet sont représentées en noir ; les liaisons mécaniques en rouge ; les liaisons PLC/capteurs & actionneurs en vert ; les liaisons capteurs/actionneurs en bleu.

inverseur/réducteur) ;

— Trois commutateurs réseau et une liaison Ethernet permettant la communication entre les PLC, les consoles dédiées et le reste du système.

Nous avons par ailleurs choisi de concentrer la supervision de l'ensemble des actionneurs à partir d'une console IHM principale (située en passerelle), les quatre IHM de supervision dédiées étant présentes par souci de redondance. Cette console de supervision communique avec les autres composants du système via un commutateur réseau principal et une liaison Ethernet reliant celui-ci aux quatre autres commutateurs.

4.1.4 Architecture réseau

Les différents composants de ce système communiquent entre eux à travers un réseau organisé selon une topologie en étoile et inspiré de [Fen+15], et constitué de quatre sous-réseaux :

1. Le sous-réseau « gouverne » regroupant le PLC, le commutateur réseau et la console de supervision dédiés au sous-système de gouverne ;

2. Le sous-réseau « moteurs Diesel » regroupant le PLC, le commutateur réseau et la console de supervision dédiés au contrôle/commande des moteurs Diesel ;

3. Le sous-réseau « turbine à gaz » regroupant le PLC, le commutateur réseau et la console de supervision dédiés au contrôle/commande de la turbine à gaz ;

4. Le sous-réseau « inverseur/réducteur » regroupant le PLC, le commutateur réseau et la console de supervision dédiés au contrôle/commande du boîtier inverseur/réducteur.

Les interactions fonctionnelles, physiques et logiques des différentes entités du système, mobilisées dans le cadre du processus de gouverne, sont illustrées par le diagramme de flux présenté en figure 4.4.

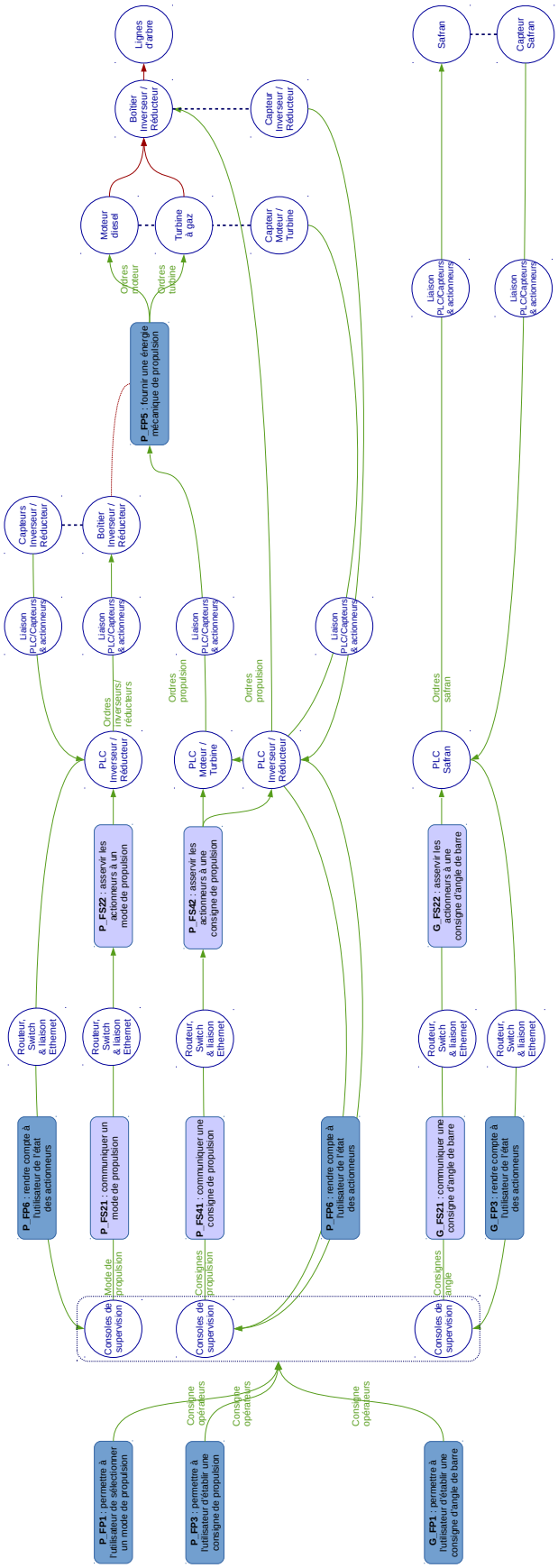


FIGURE 4.4 – Diagramme global des flux impliqués par les processus de propulsion et de gouverne de notre cas d'étude.

4.2 Modélisation du système

Nous avons modélisé notre cas d'étude fictif à travers un réseau de quarante-huit automates, issu de la composition de vingt-trois automates composant, quatre automates processus et vingt-et-un automates topologiques. Ce réseau permet de représenter l'ensemble des composants du système fictif, leurs échanges ainsi que les processus de commande et de supervision du safran, des moteurs Diesel, de la turbine à gaz et du boîtier inverseur/réducteur, comme nous le détaillerons dans ce qui suit.

4.2.1 Établissement de l'Automate Système Nominal

4.2.1.1 Automates composant

Le système imaginé étant constitué de quatre sous-systèmes identiques à celui que nous avons utilisé en exemple au cours du chapitre 2, nous avons sans surprise modélisé ses composants au travers d'automates très proches de ceux présentés en figure 2.8. Nous retrouvons par conséquent, pour chacun de ces sous-systèmes :

- un automate modélisant le comportement de la console de supervision, ou IHM, locale (*cf.* figure 4.5) ;
- un automate modélisant le comportement du contrôleur de l'actionneur afférent (*cf.* figure 4.6) ;
- un automate modélisant le comportement de l'actionneur afférent (*cf.* figure 4.7) ;
- un automate modélisant le comportement du capteur de la (des) grandeur(s) régulées par le sous-système (*cf.* figure 4.8) ;
- un automate modélisant le comportement du commutateur réseau du sous-système (*cf.* figure 4.9), que nous n'avions pas introduit dans les exemples du chapitre 2.

Notons que pour des raisons d'allègement du modèle, nous avons choisi de modéliser un seul des deux moteurs Diesel.

Par ailleurs, nous avons ajouté à ces vingt automates deux automates supplémentaires pour modéliser la console de supervision distante et le routeur central, ainsi qu'un automate représentant le comportement d'une hélice (*cf.* figure 4.10).

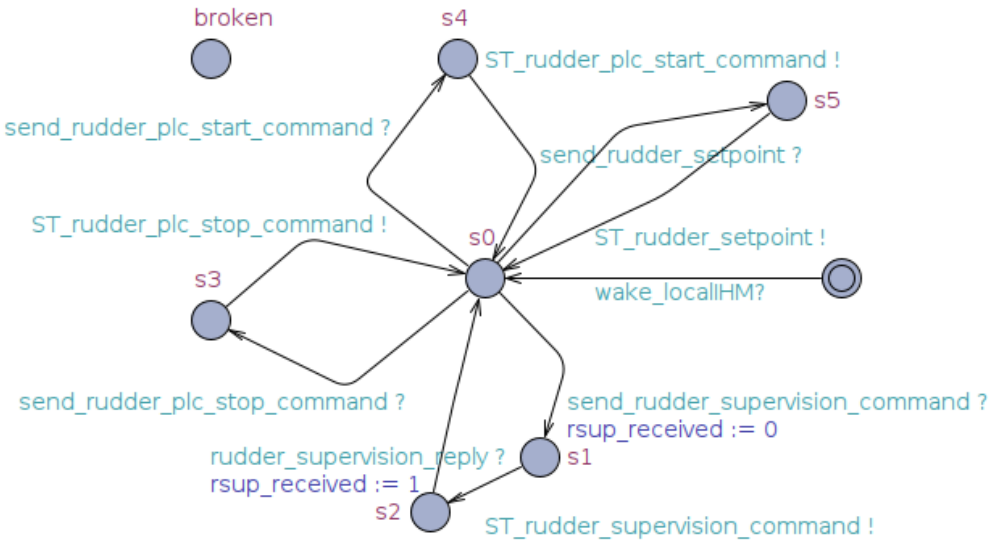


FIGURE 4.5 – Automate modélisant le comportement de l'interface de contrôle locale du sous-système de gouverne.

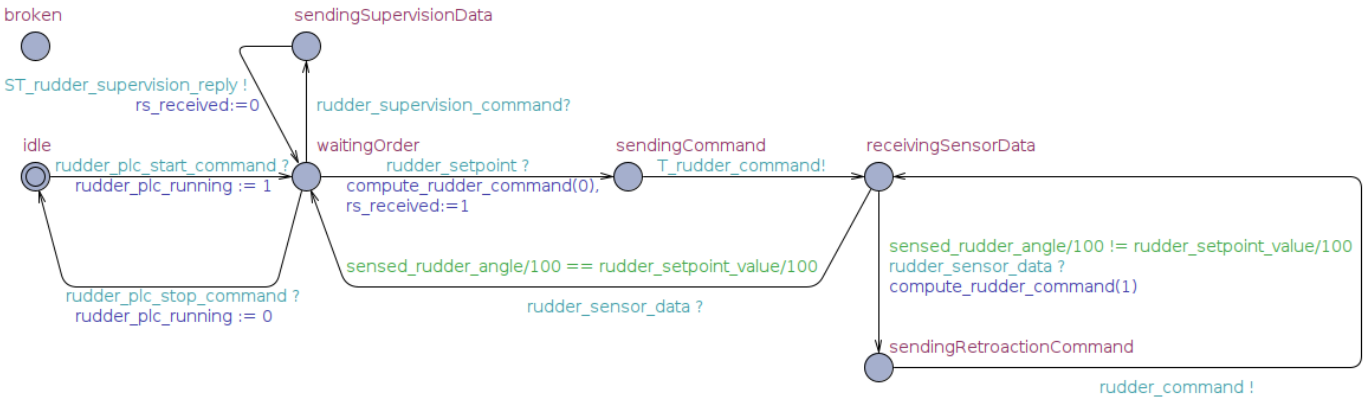


FIGURE 4.6 – Automate modélisant le comportement du PLC du sous-système de gouverne.

4.2.1.2 Modélisation des grandeurs physiques

Les grandeurs suivantes ont été modélisées :

- l'angle de barre `rudder_angle` ;
- l'angle de barre mesuré `sensed_rudder_angle` ;

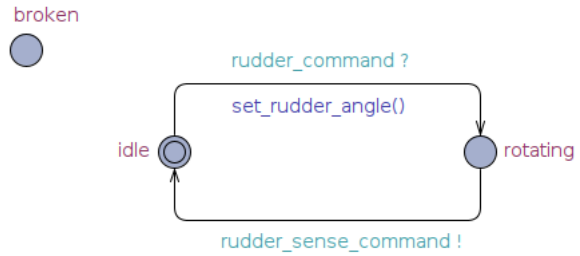


FIGURE 4.7 – Automate modélisant le comportement de l'actionneur du gouvernail.

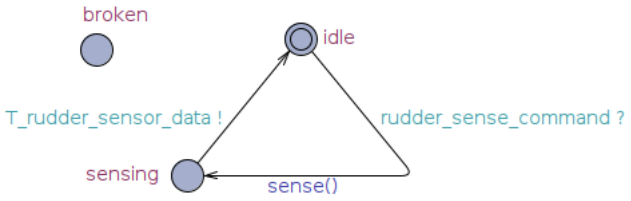


FIGURE 4.8 – Automate modélisant le comportement du capteur d'angle de barre.

— la vitesse de rotation du moteur Diesel `engine_speed` ;
— la vitesse de rotation mesurée du moteur Diesel `sensed_engine_speed` ;
— la vitesse de rotation de la turbine à gaz `turbine_speed` ;
— la vitesse de rotation mesurée de la turbine à gaz `sensed_turbine_speed` ;
— la vitesse de rotation de l'arbre moteur `shaft_speed` ;
— la vitesse de rotation mesurée de l'arbre moteur `sensed_shaft_speed` ;
— le rapport d'engrènement `gear_ratio` ;
— le rapport d'engrènement mesuré `sensed_gear_ratio`.

Nous avons également modélisé sous la forme d'une variable entière `clutched_engine` le moteur sur lequel l'arbre entraînant l'hélice était engrené – 0 s'il est débrayé, 1 pour le moteur Diesel, et 2 pour la turbine à gaz – ainsi que la « connaissance » de ce paramètre par le système `sensed_clutched_engine`. En complément, la valeur des consignes et des commandes relatives à chacun des quatre actionneurs du système a été modélisée sous la forme des dix variables suivantes :

— `rudder_setpoint_value` ;
— `rudder_command_value` ;
— `engine_setpoint_value` ;
— `engine_command_value` ;

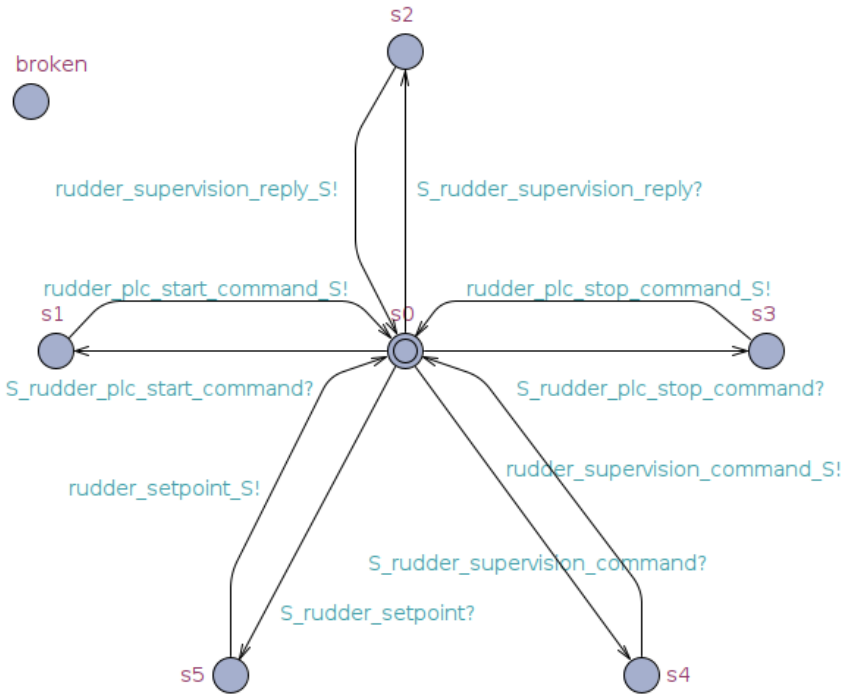


FIGURE 4.9 – Automate modélisant le comportement du commutateur réseau du sous-système de gouverne.

— `turbine_setpoint_value` ;
— `turbine_command_value` ;
— `gear_ratio_setpoint_value` ;
— `gear_ratio_command_value` ;
— `clutched_engine_setpoint_value` ;
— `clutched_engine_command_value` ;

Par ailleurs, nous avons retenu l'approche de modélisation simple présentée à la section 2.2.4 pour l'abstraction des lois de commande, transcrites au sein des fonctions `compute_actuator_command()` des automates représentant les PLC.

4.2.1.3 iii. Automates topologiques

Nous avons généré les automates topologiques et les canaux de synchronisation supplémentaires selon l'abstraction évoquée en section 2.2.1 (*cf.* figure 2.7). Toutefois, pour apporter de la lisibilité au modèle, nous avons composé ces automates

topologiques en fusionnant les états initiaux de certains d'entre eux, ainsi regroupés de manière thématique (échanges IHM ↔ Switch, Switch ↔ PLC, etc.) comme l'illustre la figure 4.11.

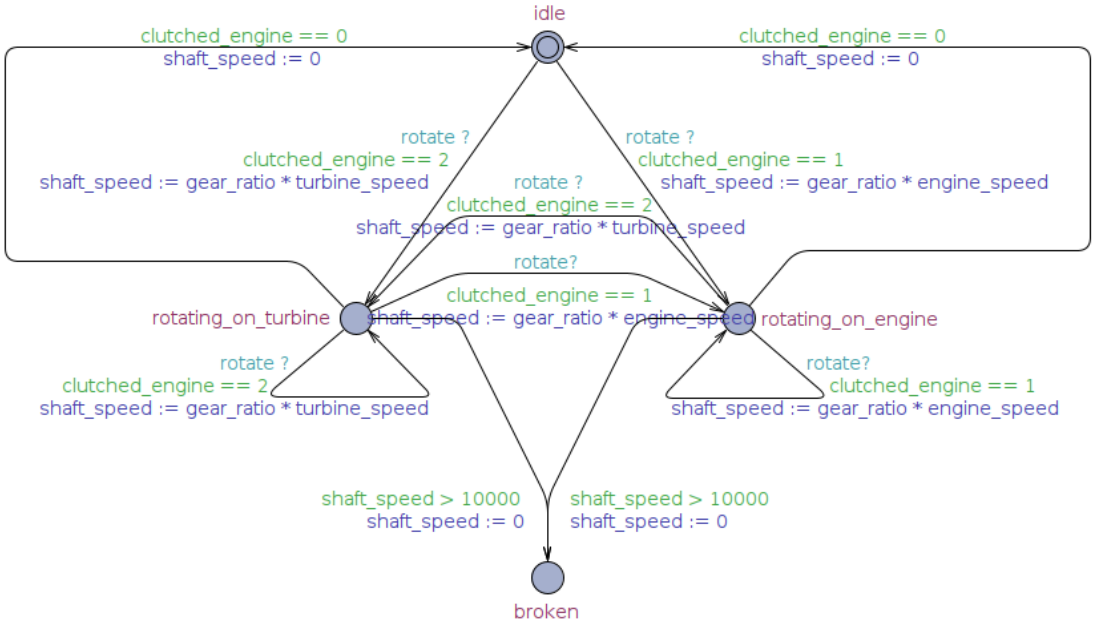


FIGURE 4.10 – Automate modélisant le comportement de l'hélice du bâtiment.

4.2.1.4 Automates processus

Outre les deux précédentes classes d'automates, nous avons également modélisé les processus suivants, permettant la mise en œuvre des fonctions de notre cas fictif :

— un processus de commande du safran ;
— un processus de commande du moteur Diesel ;
— un processus de commande de la turbine à gaz ;
— un processus de commande du boîtier inverseur/réducteur.

Ces processus sont représentés par des automates qui, après une première transition les « activant » par une synchronisation avec un automate mission, simulent l'envoi d'une consigne opérateur puis d'une requête de supervision à l'actionneur afférent, par le biais de l'IHM dédiée ou de l'IHM centralisée (*cf.* figure 4.12).

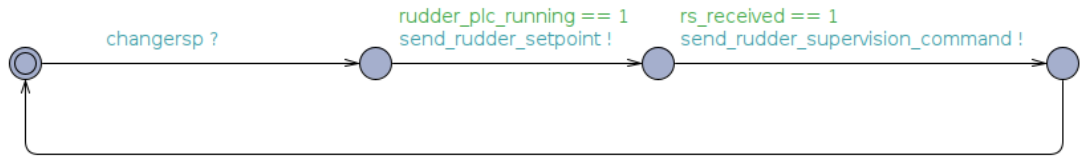


FIGURE 4.12 – Automate représentant le processus de commande du safran.

4.2.2 Élaboration et modélisation des missions

Sur la base de ce cas d'étude, nous avons imaginé quatre missions simples, mettant chacune en œuvre, partiellement ou totalement, les processus mentionnés ci-dessus. Les deux premières décrivent une séquence de navigation en croisière, et suivent les étapes suivantes :

1. activation des systèmes logiciels du bord (IHM et PLC) ;
2. communication d'une consigne d'angle de barre ;
3. communication d'une consigne au boîtier inverseur/réducteur (embrayage de la ligne d'arbre sur le moteur Diesel, et définition d'un rapport d'engrènement) ;
4. communication d'une consigne de vitesse (modérée) au moteur Diesel ;
5. communication d'une consigne d'angle de barre ;
6. communication d'une consigne de vitesse (élevée) à la turbine à gaz, représentant l'accélération pour atteindre la vitesse de croisière du bâtiment ;

7. communication d'une consigne au boîtier inverseur/réducteur (embrayage de la ligne d'arbre sur la turbine à gaz) ;
8. communication d'une consigne de vitesse (nulle) à la turbine à gaz, représentant la fin de la croisière ;
9. extinction des systèmes logiciels du bord.

Ces deux premières missions suivent rigoureusement les mêmes étapes ; la seule différence entre elles est que la première décrit une supervision centralisée de l'ensemble des processus du système (et donc l'IHM centralisée est activée en début de mission), tandis que la seconde décrit une supervision décentralisée de ces processus (et donc les IHM locales sont activées en début de mission).

Les deux dernières missions, différant également l'une de l'autre par leur mode de supervision, décrivent une séquence de navigation en eaux resserrées, et suivent les étapes suivantes :

1. activation des systèmes logiciels du bord (IHM et PLC) ;
2. communication d'une consigne d'angle de barre ;
3. communication d'une consigne au boîtier inverseur/réducteur (embrayage de la ligne d'arbre sur le moteur Diesel, et définition d'un rapport d'engrènement) ;
4. communication d'une consigne de vitesse au moteur Diesel ;
5. communication d'une consigne d'angle de barre ;
6. communication d'une consigne de vitesse au moteur Diesel ;
7. communication d'une consigne d'angle de barre ;
8. communication d'une consigne au boîtier inverseur/réducteur (débrayage) ;
9. extinction des systèmes logiciels du bord.

4.2.3 Établissement des propriétés de sûreté

Les propriétés choisies pour le calcul d'impact se répartissent en trois catégories, parmi celles présentées au chapitre précédent :

— les propriétés d'atteignabilité de l'état final des quatre missions ;
— les proprietés d'intégrité des vingt-trois composants ;
— les propriétés de conformité des vingt-trois composants.

Les propriétés d'atteignabilité de l'état final de la mission s'expriment simplement en ce qu'elles consistent à vérifier que cet état final, appelé `end` dans les quatre automates mission, est nécessairement atteint lors de l'exécution du modèle : nous les écrivons `A<> Mission.end` dans le langage de vérification d'UPPAAL.

Les propriétés d'intégrité des composants sont également exprimées de manière simple, étant donné que nous avons modélisé leur défaut potentiel d'intégrité par l'ajout d'un état `broken` dans chacun des automates les représentant. Il convient ici de vérifier que cet état ne peut aucunement être atteint ; soit, dans le langage de vérification d'UPPAAL, `A[] not Component.broken`.

Les propriétés de conformité, quant à elles, nécessitent un peu plus d'opérations pour leur expression et leur calcul. Elles nécessitent en effet la vérification de l'atteignabilité ou la non-atteignabilité (en fonction du comportement normalement attendu du composant lors de l'exécution de la mission avec l'automate de laquelle l'automate système est composé) de l'automate composant – `A<> Component.state` ou `A[] Component.state` pour chaque état `state` de l'automate `Component`. Or le langage de requêtes d'UPPAAL ne permet pas la conjonction de plusieurs requêtes, qui doivent par conséquent être vérifiées indépendamment [Sek+14] : il est par conséquent de vérifier une à une les propriétés d'atteignabilité atomiques de chaque automate composant, puis de calculer leur conjonction, qui exprimera la propriété de conformité du composant, de manière externe.

Au final, 159 propriétés ont été établies pour chacune des quatre missions ; elles sont ainsi hiérarchisées au sein des matrises d'impact, mission par mission :

Mission 1	**Mission 2**	**Mission 3**	**Mission 4**
a_Mission	a_Mission	a_Mission	a_Mission
i_Propeller	i_Propeller	i_Propeller	i_Propeller
i_Rudder	i_Rudder	i_Rudder	i_Rudder
i_Engine	i_Engine	i_Engine	i_Engine
i_Turbine	i_Turbine	i_Gearbox	i_Gearbox
i_Gearbox	i_Gearbox	i_PLC_R	i_PLC_R
i_PLC_R	i_PLC_R	i_PLC_E	i_PLC_E
i_PLC_E	i_PLC_E	i_PLC_GB	i_PLC_GB
i_PLC_T	i_PLC_T	i_IHM	i_IHM_R
i_PLC_GB	i_PLC_GB	i_Router	i_IHM_E
i_IHM	i_IHM_R	i_Switch_R	i_IHM_GB

i_Router	i_IHM_E	i_Switch_E	i_Switch_R
i_Switch_R	i_IHM_T	i_Switch_GB	i_Switch_E
i_Switch_E	i_IHM_GB	i_Sensor_R	i_Switch_GB
i_Switch_T	i_Switch_R	i_Sensor_E	i_Sensor_R
i_Switch_GB	i_Switch_E	i_Sensor_GB	i_Sensor_E
i_Sensor_R	i_Switch_T	c_Propeller	i_Sensor_GB
i_Sensor_E	i_Switch_GB	c_Rudder	c_Propeller
i_Sensor_T	i_Sensor_R	c_Engine	c_Rudder
i_Sensor_GB	i_Sensor_E	c_Gearbox	c_Engine
c_Propeller	i_Sensor_T	c_PLC_R	c_Gearbox
c_Rudder	i_Sensor_GB	c_PLC_E	c_PLC_R
c_Engine	c_Propeller	c_PLC_GB	c_PLC_E
c_Turbine	c_Rudder	c_IHM	c_PLC_GB
c_Gearbox	c_Engine	c_Router	c_IHM_R
c_PLC_R	c_Turbine	c_Switch_R	c_IHM_E
c_PLC_E	c_Gearbox	c_Switch_E	c_IHM_GB
c_PLC_T	c_PLC_R	c_Switch_GB	c_Switch_R
c_PLC_GB	c_PLC_E	c_Sensor_R	c_Switch_E
c_IHM	c_PLC_T	c_Sensor_E	c_Switch_GB
c_Router	c_PLC_GB	c_Sensor_GB	c_Sensor_R
c_Switch_R	c_Switch_R	i_IHM_R	c_Sensor_E
c_Switch_E	c_Switch_E	i_IHM_E	c_Sensor_GB
c_Switch_T	c_Switch_T	i_IHM_GB	i_IHM
c_Switch_GB	c_Switch_GB	i_Turbine	i_Router
c_Sensor_R	c_Sensor_R	i_PLC_T	i_Turbine
c_Sensor_E	c_Sensor_E	i_Switch_T	i_PLC_T
c_Sensor_T	c_Sensor_T	i_Sensor_T	i_Switch_T
c_Sensor_GB	c_Sensor_GB	i_IHM_T	i_Sensor_T
i_IHM_R	c_IHM_R	c_IHM_R	i_IHM_T
i_IHM_E	c_IHM_E	c_IHM_E	c_IHM
i_IHM_T	c_IHM_T	c_IHM_GB	c_Router
i_IHM_GB	c_IHM_GB	c_Turbine	c_Turbine
c_IHM_R	i_IHM	c_PLC_T	c_PLC_T
c_IHM_E	i_Router	c_Switch_T	c_Switch_T
c_IHM_T	c_IHM	c_Sensor_T	c_Sensor_T

c_IHM_GB	c_Router	c_IHM_T	c_IHM_T

Notons que :
— *a_Mission* désigne la propriété d'atteignabilité de l'état final de la mission ;
— le préfixe *i_* désigne la propriété d'intégrité relative au composant dont le nom suit ;
— le préfixe *c_* désigne la propriété de conformité relative au composant dont le nom suit ;
— le suffixe *_R* désigne un composant appartenant au sous-réseau « gouverne » ;
— le suffixe *_E* désigne un composant appartenant au sous-réseau « moteur Diesel » ;
— le suffixe *_T* désigne un composant appartenant au sous-réseau « turbine à gaz » ;
— le suffixe *_GB* désigne un composant appartenant au sous-réseau « inverseur/réducteur ».

4.3 Modélisation de la vulnérabilité retenue, des attaques et des contremesures

Nous avons imaginé une vulnérabilité affectant les deux PLC contrôlant le moteur Diesel et la turbine à gaz de notre bâtiment fictif, et permettant à un attaquant l'exploitant d'opérer un déni de service ou une exécution arbitraire de code.

4.3.1 Scénarii d'attaque

Deux scénarii d'attaque simples, exploitant cette vulnérabilité et visant à dégrader les capacités opérationnelles du système, ont été retenus :

1. une attaque par déni de service $\mathtt{att}_1$ sur le sous-système de supervision du moteur Diesel, au cours de laquelle un attaquant provoque l'arrêt du PLC afférent dans le but de faire perdre aux opérateurs le contrôle du navire ;

2. une attaque par modification du régulateur de la consigne de vitesse communiquée à la turbine à gaz $\texttt{att}_2$, au cours de laquelle l'attaquant modifie à la hausse la valeur de la consigne transmise lorsqu'elle dépasse un certain seuil x, dans le but de provoquer une fatigue mécanique prématurée, voire la casse, des composants mécaniques concourant à la transmission de puissance.

4.3.2 Modélisation de la vulnérabilité et des attaques

Par conséquent, la présence de cette vulnérabilité sera modélisée par le biais des deux mutations suivantes, appliquées aux deux automates représentant les PLC concernés :

— création d'un canal de synchronisation `attack_engine`, et ajout à l'automate `PLC_E` modélisant le PLC contrôlant le moteur Diesel d'une transition liant les états `waitingOrder` et `idle`, synchronisée par la co-action `attack_engine?` – l'état `idle` étant ici équivalent, dans le contexte de notre modélisation et une fois l'attaque survenue, à un état « puits » ;

— création d'un canal de synchronisation `attack_turbine`, et ajout à l'automate `PLC_T` modélisant le PLC contrôlant la turbine à gaz d'une transition bouclant sur l'état `sendingCommand`, synchronisée par la co-action `attack_turbine?`, ayant pour garde `turbine_setpoint_value > x` (`x` étant le seuil évoqué ci-dessus) et effectuant la mise à jour suivante : `turbine_setpoint_value = 6000` (6000 rpm étant la borne supérieure de la plage d'opération typique des turbines à gaz navales donné par [MK15]).

Les attaques ont été simplement modélisées par deux automates possédant un état unique et une transition bouclant sur cet état, effectuant l'action de synchronisation `attack_engine!` ou `attack_turbine!` à la condition que la garde `bridge == 1` soit satisfaite, `bridge` étant une variable entière modifiée par l'automate représentant le fonctionnement du routeur et indiquant, si elle vaut 1, l'établissement d'une connexion entre l'IHM centralisée et le routeur ; ou son non-établissement si elle vaut 0.

4.3.3 Modélisation des contre-mesures

Deux contremesures ont été retenues pour tempérer la vulnérabilité décrite précédemment :

1. une contremesure « topologique » p_1, consistant à couper la communication entre l'IHM de supervision centralisée, considérée dans nos scénarios comme étant le point d'entrée de l'attaquant dans le système, et le routeur. Elle est modélisée par une suppression du routeur via la conversion de son état `broken` en état initial ;
2. une contremesure logicielle p_2, consitant à déployer un correctif fourni par le constructeur et supprimant la vulnérabilité ; elle est modélisée par la suppression des mutations décrivant cette dernière.

4.4 Calcul d'impact : résultats et conclusions

Afin de permettre la vérification et la synthèse des résultats qui en sont issus, nous avons développé une série de scripts automatisant le traitement des 3816 requêtes [1] adressées au logiciel *verifyta* (permettant de vérifier via un outil en ligne de commande un ensemble de propriétés, enregistrées dans un fichier passé en argument, en se basant sur un système UPPAAL passé lui aussi en argument), ainsi que le traitement des résultats retournés et la génération des matrices d'impact. Ces scripts ont permis entre autres le calcul des propriétés de conformité de chaque automate composant en vérifiant chacune de ses propriétés d'atteignabilité atomiques, puis les synthétisant en calculant leur conjonction.

4.4.1 Matrices d'impact

L'exécution de ces scripts sur les automates système mutés (vulnérable, corrigés, vulnérable sous attaques et corrigés sous attaques) a permis l'établissement des mesures $\mathcal{I}^{\mathsf{v}}, \mathcal{I}_1^{\mathsf{v/att}}, \mathcal{I}_2^{\mathsf{v/att}}, \mathcal{I}_1^{\mathsf{p}}, \mathcal{I}_{1,1}^{\mathsf{p/att}}, \mathcal{I}_{1,2}^{\mathsf{p/att}}, \mathcal{I}_2^{\mathsf{p}}, \mathcal{I}_{2,1}^{\mathsf{p/att}}$ et $\mathcal{I}_{2,2}^{\mathsf{p/att}}$ dont l'expression est donnée ci-après.

1. Nous avons vérifié, comme nous le verrons par la suite, 159 propriétés $\times$ 4 missions $\times$ 6 mutations – soit 3816 d'entre elles.

$$\mathcal{I}^{\mathsf{v}} = \begin{pmatrix} 1&1&1&1\\ 1&1&1&1 \end{pmatrix}^{\top}, \mathcal{I}_1^{\mathsf{v/att}} = \begin{pmatrix} 0&0&0&0\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&0&1\\ 1&1&1&0\\ 1&1&1&1\\ 1&1&0&1\\ 0&1&1&0\\ 0&1&0&1\\ 1&0&1&1\\ 1&0&1&1\\ 0&1&1&1\\ 0&1&1&1\\ 1&0&1&1\\ 0&0&0&1\\ 1&1&1&1\\ 1&1&1&0\\ 1&1&1&1\\ 0&0&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 0&0&1&1\\ 0&0&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&0&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1 \end{pmatrix}^{\top}, \mathcal{I}_2^{\mathsf{v/att}} = \begin{pmatrix} 0&0&1&1\\ 0&0&1&1\\ 1&1&1&1\\ 0&1&1&1\\ 1&1&1&1\\ 1&0&1&1\\ 1&1&1&1\\ 0&1&1&1\\ 1&1&1&1\\ 0&0&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 0&0&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1 \end{pmatrix}^{\top}$$

$$\mathcal{I}_1^{\mathsf{p}} = \mathcal{I}_{1,1}^{\mathsf{p/att}} = \mathcal{I}_{1,2}^{\mathsf{p/att}} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}^{\top}, \mathcal{I}_2^{\mathsf{p}} = \mathcal{I}_{2,1}^{\mathsf{p/att}} = \mathcal{I}_{2,2}^{\mathsf{p/att}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{\top}$$

Soit $\mathcal{I}^{\mathsf{v}} = \mathcal{I}_2^{\mathsf{p}} = \mathcal{I}_{2,1}^{\mathsf{p}/\mathsf{att}} = \mathcal{I}_{2,2}^{\mathsf{p}/\mathsf{att}} >_{lex} \mathcal{I}_1^{\mathsf{p}} = \mathcal{I}_{1,1}^{\mathsf{p}/\mathsf{att}} = \mathcal{I}_{1,2}^{\mathsf{p}/\mathsf{att}} >_{lex} \mathcal{I}_2^{\mathsf{v}/\mathsf{att}} >_{lex} \mathcal{I}_1^{\mathsf{v}/\mathsf{att}}$; $\mathcal{E}_1^{\mathsf{p}} = \mathcal{E}_2^{\mathsf{p}} = 2$. Nous déduisons de ces résultats que :

- la présence seule de la vulnérabilité n'a pas de conséquences sur le système ;
- l'attaque att_1 menée sur le système vulnérable entraîne la perte de sa capacité à réaliser chacune de ses quatre missions ; aucun composant ne souffre de défaut d'intégrité mais plusieurs d'entre eux n'ont pas un comportement conforme à leur spécification dans le contexte opérationnel décrit par les missions (de trois à huit en fonction des missions) ;
- l'attaque att_2 menée sur le système vulnérable entraîne la perte de sa capacité à réaliser ses deux premières missions ; l'appareil propulsif peut subir une rupture mécanique et plusieurs composants n'ont pas un comportement conforme à leur spécification dans les cadres des deux premières missions (respectivement quatre et trois d'entre eux) ;
- le déploiement de la contremesure p_1 empêche le système d'accomplir sa première et sa troisième mission du fait de nombreuses non-conformités des composants entraînées par son déploiement, le routeur n'est par ailleurs plus intègre. Ceci étant, il permet d'amoindrir les effets des attaques att_1 et att_2 : deux missions peuvent toujours être accomplies par le système sous attaque, et la rupture mécanique de l'appareil propulsif n'est plus un risque ;
- la contremesure p_2 n'a pas d'effet délétère sur le système et permet de préserver ses missions et ses composants face aux attaques att_1 et att_2.

4.4.2 Discussion sur la pertinence de la métrique

Si les neuf mesures exprimées permettent de rendre compte et de hiérarchiser entre eux les impacts d'une manière paraissant cohérente, plusieurs défauts de la métrique – ou du processus d'analyse proposé au chapitre 1 – se dégagent toutefois.

Premièrement, force est de constater que si la hiérarchisation des missions entre elles et, au sein de chaque mission, des propriétés d'intégrité ou de conformité des différents composants, est aisée étant donné les quatre missions considérées, cela pourrait s'avérer plus difficile lorsque le système modélisé présente une diversité de missions et de composants plus forte. Il découle de ce constat que l'ordre lexicographique proposé au chapitre 3 peut être trop péremptoire, en ce qu'il repose sur une hiérarchisation nette des propriétés entre elles au sein de la matrice d'impact. Une piste d'amélioration pourrait à cet égard être de considérer

une méthode de comparaison plus fine, autorisant par conséquent la considération de certaines missions ou propriétés comme de criticités égales.

Une seconde faiblesse est relative à l'arbre de décision de la tâche 2-3 présenté au chapitre 1 et proposant un processus de décision indexé sur l'analyse des mesures d'impact. En effet, s'il est bien adapté lorsqu'une contremesure au moins parmi celles étudiées a un impact inférieur à celui de la vulnérabilité, il devient trop manichéen si toutes les contremesures à disposition, à l'instar de p_1, induisent une régression.

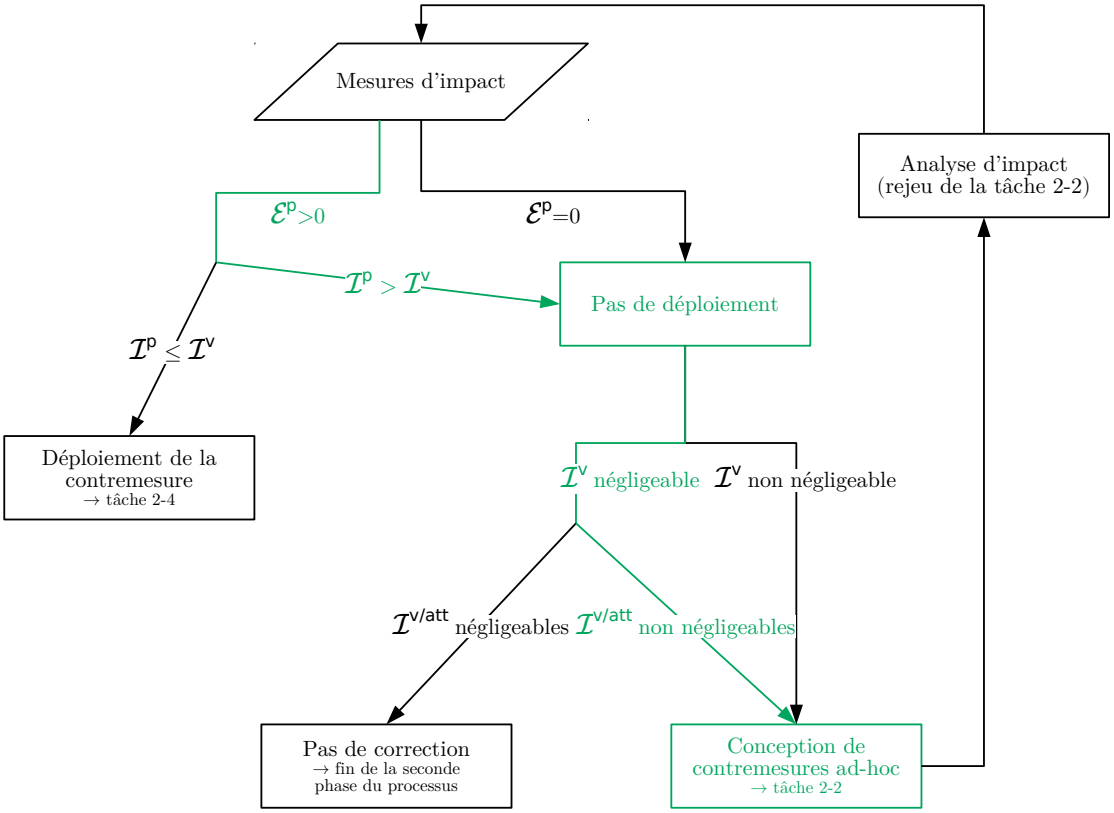


FIGURE 4.13 – Arbre de décision de la tâche 2-3 appliqué à la contremesure p_1.

En effet, son application stricte (*cf.* figure 4.13) mène à la conception de nouvelles contremesures ; or nous pouvons aisément imaginer une situation où l'urgence de la correction d'une vulnérabilité ne permet pas l'attente induite ; auquel cas, la contremesure de moindre régression et de meilleure efficacité devrait être choisie.

Une troisième piste d'amélioration naît du fait qu'une contremesure consis-

tant à isoler topologiquement uniquement les sous-systèmes « moteurs Diesel » et « turbine à gaz » aurait été traduite par un impact proche de celui de p_1 dans la mesure où les missions 1 et 3 seraient également apparues comme non-réalisables. Or il existe une nuance entre une telle contremesure et p_1 : en effet, il est possible dans un cas de continuer à opérer deux des quatre systèmes depuis l'IHM centralisée, et donc à accomplir la moitié des tâches impliquées par la mission. L'ajout de propriété d'atteignabilité d'états intermédiaires de l'automate mission, ou de missions subsidiaires (ici, une variante des deux missions où les sous-systèmes de gouverne et d'inversion/réduction sont supervisés seuls depuis l'IHM centralisée), permettrait d'introduire une nuance dans l'expression de la capacité du système à accomplir ses missions.

Enfin, l'analyse « humaine » de la matrice d'impact est complexe dès lors qu'un grand nombre de coefficients y figurent, ce qui est inévitable si elle s'applique à de grands systèmes cyber-physiques ; une interface utilisateur listant chaque propriété en regard d'un résultat traduit sous forme de code couleur, par exemple, apporterait une lisibilité accrue aux résultats.

4.4.3 Discussion sur les temps de calcul

Le calcul des matrices d'impact,réalisé sur un ordinateur de bureau (processeur Intel Core i7-6500U exécutant ses instructions à une fréquence de 2,5 GHz, 16 Go de mémoire) ; nous a permis de constater les temps d'exécution suivants :

	Mission 1	**Mission 2**	**Mission 3**	**Mission 4**	**Total**
$\mathcal{I}^{\mathsf{v}}$	4 min 10 s	10 min 16 s	3 min 35 s	10 min 6 s	28 min 6 s
$\mathcal{I}_1^{\mathsf{v/att}}$	3 min 17 s	7 min 39 s	3 min 27 s	8 min 51 s	23 min 17 s
$\mathcal{I}_2^{\mathsf{v/att}}$	3 min 9 s	8 min 11 s	3 min 42 s	10 min	25 min 2 s
$\mathcal{I}_1^{\mathsf{p}}$	$<$ 1 s	4 min 21 s	$<$ 1 s	4 min 25 s	8 min 47 s
$\mathcal{I}_{1,1}^{\mathsf{p/att}}$	$<$ 1 s	4 min 24 s	$<$ 1 s	4 min 23 s	8 min 48 s
$\mathcal{I}_{1,2}^{\mathsf{p/att}}$	$<$ 1 s	4 min 23 s	$<$ 1 s	4 min 23 s	8 min 27 s

NB : nous n'avons pas calculé les matrices $\mathcal{I}_2^{\mathsf{p}}, \mathcal{I}_{2,1}^{\mathsf{p/att}}$ et $\mathcal{I}_{2,2}^{\mathsf{p/att}}$: étant donné que le déploiement de la contremesure p_2 consiste à revenir à l'Automate Système Nominal, les résultats des trois mesures d'impact étaient triviaux. Par ailleurs, les statistiques d'utilisation de la mémoire et du processeur ont été constantes pendant les calculs, à 25% d'utilisation CPU (vraisemblablement un seul des quatre cœurs, à pleine charge) et entre 0,96 et 1,28 Go mobilisé.

Ces mesures ayant été calculées les unes après les autres, le temps de calcul global s'est élevé à une heure et quarante-deux minutes, soit un temps d'évaluation moyen de 1,6 seconde par propriété. Nous pouvons toutefois remarquer que la mesure d'impact la plus coûteuse ($\mathcal{I}^{\mathsf{v}}$, Mission 2) a demandé un peu plus de 10 minutes de calcul ; par conséquent, une parallélisation de ces vingt-quatre séries de vérifications sur vingt-quatre ordinateurs distincts aurait permis de faire chuter le temps d'évaluation moyen à 0,16 seconde par propriété.

Par ailleurs, nous n'avons pas été confrontés à une explosion combinatoire, étant donné que la mémoire nécessaire au *model-checking* n'a pas dépassé les 1,28 Go. Précisions ici que *verifyta* n'a été exécuté sans aucune de ses options de réduction de l'espace d'état, permettant de réduire la mémoire nécessaire à l'exploration du modèle (contre un temps de calcul supérieur toutefois).

Enfin, nous pouvons constater que les choix de modélisation permettent de réguler les temps de calcul dans la mesure où les missions 1 et 3 sont beaucoup moins coûteuses à la vérification que les missions 2 et 4 ; les automates composant le modèle étant plus fortement synchronisés à la composition du réseau avec les missions 1 et 3 (supervision depuis l'IHM centralisée) qu'à la composition avec les deux autres (supervisions parallèles depuis quatre IHM).

4.5 Conclusion

Les différentes expérimentations menées dans le cadre de ces travaux de recherche, dont le cas que nous nous sommes attachés à présenter dans le chapitre qui se clôt, nous ont permis de mettre à l'épreuve différentes tâches du processus de gestion de correctifs proposé au chapitre 1.

Les résultats exposés ci-dessus ne permettent pas de discuter finement de la pertinence et des limites des tâches de la première phase visant à la collecte des modèles du système étudié et à l'établissement des automates modélisant ce dernier, trivialement validées par le cas d'étude ici présenté dans la mesure où celui-ci a été spécialement conçu pour les besoins de ces travaux de recherche ; toutefois, d'autres expérimentations, sur des cas d'étude réels (navire civil, système d'information d'entreprise) dont nous n'avons pas pu rendre compte pour des raisons de confidentialité, ont permis de confirmer la pertinence du choix du corpus de modèles d'entrée ainsi que des paradigmes de modélisation retenus et présentés au chapitre 2, y compris pour la modélisation de systèmes dépourvus d'actionneurs,

de contrôleurs et d'asservissements. Deux limitations se dégagent toutefois de ces mises à l'épreuve : d'une part, la démarche de modélisation « simple », toute bénéfique qu'elle soit en termes de simplicité de modélisation et de réduction des temps d'exploration des modèles lors du calcul d'impact, demeure limitée et le choix des automates hybrides paraît donc plus indiqué dès lors que des lois physiques et l'évolution de grandeurs continues doivent être modélisées ; d'autre part, en fonction de la granularité choisie les modèles comportementaux et leur temps de conception peuvent devenir particulièrement conséquents. En demeurant à une granularité moyenne comme celle présentée dans ce chapitre, le temps de modélisation reste contenu (il a fallu deux jours de travail pour établir les modèles UPPAAL à partir du cas d'étude fictif présenté) ; par ailleurs, ce coût de modélisation est à mettre en regard avec la durée de vie des systèmes étudiés, et se « dilue » d'autant plus aisément que les navires modernes sont le plus souvent construits par classes. Cela constitue un avantage réel de l'approche de modélisation que nous proposons et permet de valider son applicabilité.

L'expérimentation présentée permet en revanche à elle seule de tirer des conclusions sur la seconde phase du processus, visant à analyser l'impact d'une vulnérabilité, d'attaques et de contremesures associées et à enrichir le modèle comportemental du système après une prise de décision relative à la correction de la vulnérabilité découverte. Nous avons pu constater que la modélisation des vulnérabilités, attaques et contremesures à travers un même formalisme (celui des mutations) permettait une transcription suffisamment riche, au regard de nos objectifs, de leurs effets sur le système modélisé. Nous avons également pu, à travers les scénarios d'attaque fictifs et les deux contremesures évaluées, estimer la pertinence et les limites de la métrique proposée au chapitre 3 ainsi qu'établir des temps de calcul pouvant servir de référence pour une estimation concrète du coût calculatoire de l'analyse d'impact. Notons que si les impacts estimés paraissent cohérents d'un point de vue intuitif, nous n'avons toutefois pas eu l'occasion de les confronter avec des impacts réels à l'occasion de nos différentes expérimentations, et une incertitude demeure à cet égard.

Au regard des cas d'étude étudiés à l'occasion des travaux présentés à travers ces pages, nous pouvons observer de manière plus générale que le formalisme de modélisation retenu est suffisamment puissant, adaptable et expressif pour fonder des analyses pertinentes au-delà de celles proposées dans le seul cadre du processus proposé au chapitre 1. Une importante marge d'amélioration et des perspectives d'utilisation de ces modèles et méthodes existent donc, que ce soit dans un contexte

d'analyse des événements d'origine cyber ou d'analyse de modifications de pure sûreté de fonctionnement.

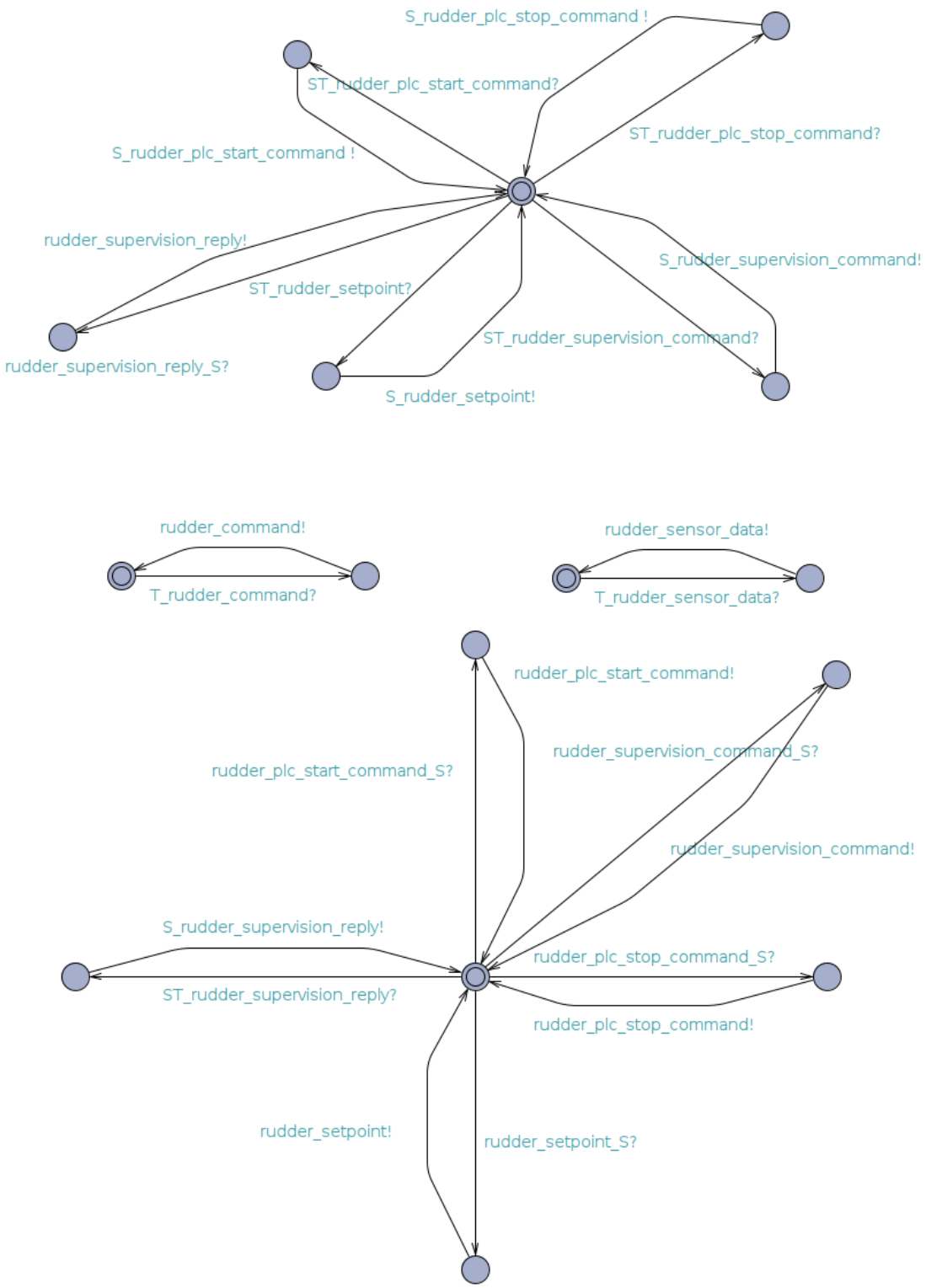


FIGURE 4.11 – Automates topologiques du sous-système de gouverne.

Conclusion générale et perspectives

Depuis que nous avons posé, en introduction, les cinq questions de recherche découlant de notre objectif final – établir un processus et un ensemble d'outils, métriques et modèles permettant une gestion maîtrisée des contremesures de sécurité appliquées à un système complexe –, les chapitres qui se sont succédés se sont attachés à leur apporter une réponse en présentant les contributions que nous avons développées. La conclusion qui s'ouvre ici nous permettra de les synthétiser, ainsi que de rappeler leurs limites et d'esquisser les perspectives de recherche qui en découlent.

Rappel des questions de recherche

Étant donné que la synthèse de nos travaux établira une correspondance entre les apports formulés et les questions de recherche, nous les rappelons ici afin de faciliter la lecture des pages qui vont suivre.

QR1 : Un besoin méthodologique : comment architecturer un processus de gestion de correctifs adapté ?

QR2 : Un besoin de modélisation : comment modéliser un système aussi complexe et aussi hétérogène qu'un navire ?

QR3 : Un besoin de conception de métrique : comment exprimer l'impact d'une vulnérabilité, de l'exploitation d'une vulnérabilité, d'une contremesure, sous forme de mesure comparable contenant suffisamment d'information ?

QR4 : Un besoin méthodologique subsidiaire : comment mener les calculs

aboutissant à ces mesures ?

QR5 : Un besoin de modélisation subsidiaire : quelle modélisation des vulnérabilités, des attaques et des contremesures est appropriée dans la visée de QR3 et QR4 ?

1. Travaux réalisés

Définition d'un processus de gestion de correctifs de sécurité

La première contribution de ce travail de recherche est la définition d'un processus de gestion de correctifs appliqué au contexte des grands systèmes cyber-physiques. Structuré par deux phases s'inscrivant dans des durées distinctes, nous l'avons conçu dans un but d'adaptabilité forte au système qu'il analyse : ainsi, ses différentes tâches peuvent être appliquées à n'importe quel navire, voire à n'importe quel système cyber-physique pour peu que suffisamment d'informations soient disponibles pour le modéliser selon notre approche. La clé de cette modularité réside dans la première phase du processus, s'inscrivant dans le temps long du cycle de vie du système.

Apport méthodologique 1 : une méthode et des outils de collecte et de génération de modèles du système

Cette première phase vise à établir une modélisation formelle du système étudié, en s'appuyant sur un ensemble d'informations d'entrée. Or ces informations sont hétérogènes à deux titres : elles peuvent varier d'un système (ou d'une classe) à l'autre, et elles sont exprimées dans des formalismes différents. Nous avons donc conçu cette phase en nous appuyant sur la fédération de modèles [Guy+13]. Cette approche nous a permis de proposer une méthode de modélisation « sur-mesure », s'adaptant aux modèles existants et informations collectées auprès des différents acteurs impliqués dans la conception, la maintenance et l'opération du système.

Nous avons, en appui de cette phase, proposé et défini des outils et des formalismes de modélisation intermédiaires. Ces formalismes servent à la synthèse des informations collectées, et s'appuient sur des langages de balisage que nous avons définis *ad-hoc*. Ces langages permettent une interprétation aisée et servent à l'automatisation des opérations de génération de la modélisation du système ; leur

détail est présenté dans l'annexe A.

La définition de cette première phase, la proposition d'outils la soutenant et la définition de langages et de formalismes de modélisation intermédiaire constituent un premier apport méthodologique de notre travail, permettant de répondre partiellement à QR1.

Apport méthodologique 2 : une méthode et des outils de collecte et de génération de modèles du système

La seconde phase du processus que nous avons proposé, s'inscrivant dans le temps plus contenu de la réaction à la découverte d'une vulnérabilité affectant le système, vise à évaluer l'efficacité et l'innocuité des contremesures de sécurité disponibles et à fournir des indicateurs concrets à cet égard afin de faciliter la prise de décision subséquente à la découverte d'une vulnérabilité.

Dans le cadre de cette seconde phase, nous avons formulé des propositions méthodologiques quant aux opérations de veille permettant de déceler les vulnérabilités affectant les composants du système. Afin d'automatiser une partie de ces opérations, nous avons également proposé un outil de collecte continue des vulnérabilités, VulneRobot, dont l'implémentation a été réalisée par deux étudiants d'IMT Atlantique dans le cadre d'un projet de troisième année.

Nous avons ensuite défini une méthode de calcul et de comparaison des impacts des vulnérabilités, attaques et correctifs. Cette méthode permet, étant donné le modèle formel du système établi lors de la première phase et les informations collectées lors des opérations de veille, d'obtenir un ensemble de mesures d'impact comparables entre elles.

Enfin, nous avons conçu et proposé une approche fondée sur des arbres de décision permettant d'orienter la réflexion en fonction de ces mesures, et un processus subsidiaire pour guider le déploiement des contremesures sur le système.

L'ensemble de ces propositions constitue le second apport méthodologique de notre travail, permettant de compléter la réponse apportée à QR1.

Définition d'un formalisme de modélisation d'un grand système cyber-physique et des événements [2] l'affectant

Apport théorique 1 : un formalisme de modélisation d'un système naval et de ses missions

Nous avons proposé, dans le cadre de ces travaux de recherche, un formalisme de modélisation des systèmes cyber-physiques complexes. Ce formalisme est spécialement conçu pour servir de base au calcul d'impact des événements cyber et à ce titre, répond à trois impératifs : il doit (1) être vérifiable (puisque nous souhaitons *calculer* les impacts), (2) permettre de représenter la dépendance entre le système et ses missions (puisque nous souhaitons calculer les impacts sur la capacité du système à accomplir ses missions) et (3) être exhaustif (puisque nous souhaitons calculer ces impacts sur *l'ensemble* du système).

Nous avons par conséquent conçu un formalisme de modélisation fondé sur l'utilisation de réseaux d'automates finis hybrides. Ce formalisme permet de représenter les missions, les processus, les fonctions, les composants d'un système et les liens topologiques qui les unissent, retranscrivant ainsi les différentes strates de la hiérarchie d'abstraction de Jens Rasmussen [Ras85] structurant les informations synthétisées en son sein. Par ailleurs, ce formalisme a été conçu de sorte à intégrer la dynamique des grandeurs physiques du système modélisé, grâce à deux approches proposées : soit une approche « simple » abstrayant certaines lois dynamiques, soit une approche plus complète intégrant les équations différentielles d'évolution de ces grandeurs au réseau d'automates. Nous avons par ailleurs, afin de valider l'intégration de ces grandeurs physiques, établi une modélisation simple de la dynamique d'un navire propulsé en étendant le modèle proposé par Luc Jaulin [Jau04].

Ce formalisme de modélisation permettant d'abstraire des objets de nature très diverse constitue notre premier apport théorique, permettant de répondre à QR2.

2. Découverte d'une vulnérabilité, survenue d'une attaque, déploiement d'une contremesure.

Apport théorique 2 : un formalisme de modélisation des vulnérabilités, contremesures et attaques

Notre second apport théorique réside en la conception d'un formalisme de modélisation des événements cyber affectant le système modélisé. Ce formalisme de modélisation est complémentaire à celui utilisé pour représenter le système et ses missions : nous proposons ainsi un formalisme unifié permettant d'abstraire le système, ses missions et les événements.

Il permet en effet, d'une part, de représenter les attaques de manière séquentielle, de manière homologue aux missions ; et d'autre part, de modéliser les vulnérabilités et l'application des contremesures via l'altération du modèle formel du système. En effet, la découverte d'une vulnérabilité ou le déploiement d'un correctif est soit une évolution du système, soit une évolution de la connaissance que l'on en a : le modèle préexistant devient obsolète et il convient de le faire évoluer afin qu'il devienne représentatif du système réel. Nous avons à ces fins choisi de décrire les vulnérabilités et contremesures à travers un ensemble de mutations, modifications atomiques d'un automate ou d'un réseau d'automates, permettant de retranscrire leur survenue par l'altération des modèles du système.

En complément, nous avons défini un ensemble de mutations génériques décrivant la plupart des modifications pouvant être entraînées par une vulnérabilité ou le déploiement d'un correctif.

Enfin, nous avons défini et proposé un langage de modélisation des mutations, permettant leur abstraction dans un formalisme proche du langage naturel et l'automatisation de leur application au cours de la tâche 2-2 du processus de gestion de correctifs.

Cet ensemble de propositions constitue le second apport théorique de nos travaux, et permet de répondre à QR5.

Apport méthodologique 3 : une méthode de calcul d'impact des événements cyber

Ces travaux nous ont également permis de concevoir une méthode de calcul d'impact, dont l'objectif est de fournir des mesures objectives et comparables entre elles afin de hiérarchiser les différents correctifs à disposition lors de la découverte d'une vulnérabilité. Ces mesures permettent en effet l'estimation de deux

paramètres cruciaux : la non-régression de chaque contremesure, et son efficacité.

Cette méthode, déclenchée à l'apparition d'une vulnérabilité, se déroule en quatre étapes successives : (1) calcul de l'impact de la vulnérabilité ; (2) calcul de l'impact de chaque contremesure à disposition ; (3) calcul de l'impact d'un jeu d'attaques exploitant la vulnérabilité sur le système vulnérable ; (4) calcul de l'impact d'un jeu d'attaques exploitant la vulnérabilité sur chaque système corrigé. Cette méthode est fondée sur des opérations de *model-checking* : à chacune de ses quatre étapes, la conformité du modèle muté du système à ses propriétés de sûreté et de sécurité est vérifiée. Chaque étape aboutit donc à un ensemble de valeurs binaires illustrant l'impact de l'événement considéré sur la capacité du système à accomplir ses missions, sur l'intégrité ou la conformité de ses composants, ou encore sur la confidentialité des informations qu'il traite.

La conception de cette méthode constitue notre troisième apport méthodologique et notre réponse à QR4.

Apport théorique 3 : une métrique d'expression des impacts

En complément de notre méthode de calcul d'impact, nous avons conçu une métrique permettant d'exprimer les impacts à partir des valeurs binaires issues des opérations de *model-checking* successives. Cette métrique permet une restitution exhaustive de l'information produite lors du calcul d'impact, et fournit des mesures aisément comparables entre elles. Cette métrique exprime les impacts de manière matricielle, chaque coefficient de la matrice étant le résultat de la vérification d'une des propriétés vérifiées au cours des opérations de calcul d'impact. Ces coefficients étant ordonnés entre eux, nous avons de surcroît défini un ordre adapté permettant de comparer et hiérarchiser aisément les différentes matrices d'impact entre elles. Cette métrique, adaptable en fonction des propriétés de sûreté et de sécurité, du contexte opérationnel du système et des criticités relatives de ses missions, fournit une base efficace à la prise de décision dans le cadre du processus que nous proposons mais peut également fonder des analyses de sûreté dans un cadre plus large.

La définition de cette métrique constitue notre troisième apport théorique, et nous permet de répondre à QR3.

2. Limitations et perspectives

En marge de la présentation de nos apports, nous avons pu évoquer au cours des trois premiers chapitres et au sein du chapitre 4, les limites inhérentes aux solutions proposées dans le cadre de nos travaux. Nous nous attacherons ici à en faire la synthèse, ainsi qu'à évoquer des limites non-soulevées précédemment.

i. Améliorer la finesse de modélisation

Comme nous avons pu l'évoquer au terme du chapitre 2, le formalisme de modélisation des systèmes demeure perfectible. Conformément à ce que nous avons alors expliqué, « des travaux futurs pourront notamment s'intéresser à diverses pistes d'enrichissement des modèles produits, par exemple par l'utilisation des plans du système (pour la prise en compte des risques liés à l'émission de signaux parasites par les composants, ou pour la modélisation de contremesures organisationnelles par restriction d'accès à une zone donnée du navire pour éviter l'interaction avec un composant vulnérable, ...) ou par une reflexion plus poussée sur la modélisation de la confidentialité des données utilisées par le système. Ces enrichissements permettraient une expression d'autant plus fine de l'impact des vulnérabilités, attaques et contremesures. »

Nos travaux appellent par ailleurs une amélioration de la prise en compte de la dynamique des systèmes au sein du modèle proposé au chapitre 2. Une perspective intéressante pourrait être de renforcer le lien entre la modélisation du système dynamique et le réseau d'automates l'abstrayant, afin notamment de mettre en regard une simulation réaliste de ses grandeurs physiques et celle de l'évolution, par essence discrète, de ses composants logiques.

ii. Améliorer la finesse d'analyse des impacts

Une piste d'enrichissement connexe réside en l'extension des propriétés considérées pour le calcul d'impact. Nous nous sommes en effet bornés dans ces travaux à l'expression des propriétés suivantes :

- A : atteignabilité de l'état final des missions ;
- T : atteignabilité de l'état final des missions dans des délais données ;
- C : conformité du comportement de chaque composant du système à sa spécification dans le contexte de chacune des missions ;

— I : intégrité de chaque composant ;
— Pr : confidentialité des données traitées par le système.

Nous pourrions imaginer d'enrichir ces cinq catégories par des propriétés ayant trait par exemple à des atteintes partielles des objectifs des missions, ou à la qualité d'atteinte de ces objectifs.

De manière plus générale, le processus que nous avons proposé fonde exclusivement la prise de décision sur les mesures issues de notre méthode de calcul d'impact basée sur la vérification des modèles comportementaux du système. L'analyse des impacts gagnerait à être enrichie par des outils et méthodes complémentaires tels que ceux proposés par Wael Kanoun *et al.* [Kan+08 ; Kan+07] ou Gustavo Gonzalez Granadillo *et al.* [GG+17].

iii. Améliorer l'efficacité algorithmique

Une des limites importantes des travaux présentés dans ces pages réside dans le coût calculatoire inhérent à notre méthodologie de calcul d'impact. Nous avons évoqué au cours du chapitre 3 différentes pistes d'amélioration pour limiter les temps de calcul d'impact et les ressources nécessaires pour le mener ; toutefois, nous n'avons pas pu les éprouver et des perspectives de recherche sont donc ouvertes à cet égard. De surcroît et de manière connexe, la fréquence de découverte des vulnérabilités peut être élevée sur un système aussi complexe et hétérogène qu'un bâtiment moderne. À titre d'exemple, toutes versions confondues, Windows 10 a par exemple vu entre 257 et 357 vulnérabilités êtres révélées chaque année depuis 2017 : des travaux ultérieurs pourraient se consacrer à l'estimation de la fréquence de découverte de vulnérabilités sur un système naval, et à proposer en fonction des améliorations du processus de gestion de correctifs.

iv. Améliorer les interactions processus/utilisateurs

Une limitation supplémentaire du processus que nous proposons a trait à l'implémentation de plusieurs outils présentés. En effet, les chaînes de compilation définies au chapitre 1 n'ont été que partiellement implémentées : seuls existent pour le moment les scripts de génération (et de modification) des modèles – les outils d'interprétation et de compilation intermédiaire des langages d'entrée, que nous avons proposé en annexe A et au chapitre 3, restent à développer afin de parfaire l'automatisation du processus.

Enfin, l'aide à la prise de décision pourrait être étendue par la conception d'interfaces utilisateur explicites comportant par exemple des diagrammes de synthèse permettant une visualisation aisée des fonctions, missions et composants affectés par les vulnérabilités, attaques et contremesures. Dans l'état actuel des choses, seule l'analyse des matrices d'impact et l'interface sommaire de VulneRobot permettent une telle visualisation : des travaux futurs pourraient se focaliser sur la sélection et la restitution d'éléments d'intérêt afin de faciliter la décision en cas de crise et augmenter la conscience de la sécurité ainsi que de la sûreté du système.

Notons que les pistes d'amélioration que nous venons de présenter n'ont pas l'ambition de l'exhaustivité ; si elles répondent aux limites identifiées ici, d'autres limites existent et appellent des réponses scientifiques, s'intégrant ou non au sein de ces quatre axes d'amélioration.

Annexe

A Formalisme des modèles intermédiaires

nécessaires à la génération semi-automatisée de la modélisation d'un système naval proposée dans le contexte de ces travaux

Table des matières

A.1 Formalisme du modèle d'architecture système

Le modèle d'architecture système est exprimé à travers de deux documents. Le premier est un fichier XML qui réunit les informations ayant trait aux variables, composants, liens topologiques, flux

et chaînes fonctionnelles, selon le formalisme exposé ci-dessous :

<variable> $variable\_type, variable\_name$ </variable>
<node> $component\_name$ </node>
<link> $component_i\_name, component_j\_name$ </link>
<flow> $flow\_id, flow\_name, component_i\_name, component_j\_name$ </flow>
<function> $function\_name, flow\_id_k, ..., flow\_id_l$ </function>

À titre d'exemple, le modèle suivant décrit de manière schématique un sous-système de gouverne, composé d'une interface de contrôle, d'un PLC, d'un actionneur contrôlant le gouvernail et d'un capteur mesurant les angles de barre réalisés. Un flux bi-directionnel lie l'interface de contrôle au PLC (afin de transmettre les consignes choisies par l'opérateur, et lui fournir les informations de supervision), et deux flux mono-directionnels lient le PLC à l'actionneur (à des fins de commande) et le capteur au PLC (dans le but d'opérer une boucle de rétroaction, et de fournir des informations qui seront remontées à l'opérateur). Ces flux se succèdent au sein de la chaîne fonctionnelle de gouverne («*steering*») :

```
1  <variable> int, position_x </variable>
2  <variable> int, position_y </variable>
3  <variable> int, course </variable>
4  <variable> int, speed </variable>
5  <variable> int, rudder_angle </variable>
6  <variable> int, rudder_setpoint </variable>
7
8  <node> interface </node>
9  <node> rudder_PLC </node>
10 <node> rudder_actuator </node>
11 <node> rudder_sensor </node>
12
13 <link> interface, rudder_PLC </link>
14 <link> rudder_PLC, rudder_actuator </link>
15 <link> rudder_PLC, rudder_sensor </link>
16
17 <flow> 0, interface, rudder_PLC, crew_command </flow>
18 <flow> 1, rudder_PLC, interface, crew_supervision </flow>
19 <flow> 2, rudder_PLC, rudder_actuator, command </flow>
20 <flow> 3, rudder_sensor, rudder_PLC, feedback </flow>
21
```

```
22 <function> steering , 0, 2, 3, 2, 3, 1 </function>
```

Le second document complétant le modèle d'architecture système détaille quant à lui :

- pour chaque chaîne fonctionnelle, les modifications des grandeurs physiques liées aux actionneurs du système et impliquées par son exécution, en fonction des entrées de la chaîne fonctionnelle (grandeurs observées et consignes fixées par les opérateurs du système). Cela peut être, par exemple, la loi de commande liant l'angle de barre courant mesuré à la consigne d'angle de barre ;
- les équations décrivant la dynamique du système, ou l'évolution des grandeurs physiques qui y sont liées (position, cap, vitesse, etc.).

Ce document sert notamment de base, au sein de la tâche 1-2, à l'élaboration des formules de mises à jour des variables liées aux transitions des automates modélisant le comportement des composants.

A.2 Formalisme de modélisation de la description comportementale des composants

Les automates temporisés modélisant le comportement de chaque composant [1] sont également exprimés par le biais de fichiers XML (un par automate) qui suivent les règles syntaxiques suivantes :

<states> $state_0\_name, \ldots, state_n\_name$ </states>
<initial_state> $initial\_state\_name$ </initial_state>
<clocks> $clock_0\_name, \ldots, clock_p\_name$ </clocks>
<variable> $variable\_type, variable\_name$ </variable>
<invariant> $state\_name, clock\_name <= integer$ </invariant>
<transition> $source\_state\_name, destination\_state\_name, guard,$
$update_0, \ldots, update_q, action$ </transition>

NB : les actions correspondant à la réception ou à l'envoi d'un message entre deux automates devront suivre le formalisme suivant :

1. Notons que la construction de ces automates est basée sur les modèles du corpus ① mais aussi sur le modèle d'architecture système : en effet, la description des fonctions dans lesquelles un composant est impliqué peut être partiellement transcrite dans la description de son comportement.

$\{receive, send\}\_message\_name$; l'absence d'un champ (garde ou mise à jour d'horloges ou de variables) dans la balise *transition* est signifiée par la chaîne *none*.

Considérons l'automate modélisant le comportement du PLC mentionné dans l'exemple ci-dessus. Une représentation simplifiée de son fonctionnement peut être de considérer six états dont un état initial (*idle*) où le composant est éteint (ou dans une configuration ne lui permettant pas d'effectuer les tâches de commande de l'actionneur), un état d'attente des ordres de l'opérateur (*waitingOrder*), un état d'envoi à l'actionneur de la commande initiale issue de la consigne reçue de l'opérateur (*sendingInitialCommand*), deux états modélisant la boucle de rétroaction (*receivingSensorData* et *sendingRetroactionCommand*), et un état où le PLC rend compte à l'opérateur des données du composant supervisé (*sendingSupervisionData*).

Les transitions de cet automate sont les suivantes :

— *idle* → *waitingOrder* lors de la réception de l'ordre de démarrage ($receive\_start\_order$) ;

— *waitingOrder* → *sendingInitialCommand* lors de la réception de la consigne fixée par l'opérateur ($receive\_rudder\_setpoint$) ;

— *sendingInitialCommand* → *receivingSensorData* avec envoi de la commande issue de la consigne fixée par l'opérateur ($send\_rudder\_command$) ;

— *receivingSensorData* → *sendingRetroactionCommand* lors de la réception de données du capteur indiquant que l'angle de barre réalisé est en-dehors des tolérances spécifiées (conjonction de l'action $receive\_sensor\_data$ et de la garde $rudder\_angle \neq rudder\_setpoint$) ;

— *sendingRetroactionCommand* → *receivingSensorData* avec envoi de la commande issue de la consigne fixée par l'opérateur et des données remontées par le capteur ($send\_rudder\_command$) ;

— *receivingSensorData* → *waitingOrder* lors de la réception de données du capteur indiquant que l'angle de barre réalisé est dans la plage de tolérance spécifiée (conjonction de l'action $receive\_sensor\_data$ et de la garde $rudder\_angle = rudder\_setpoint$) ;

— *waitingOrder* → *sendingSupervisionData* lors de la réception de la demande de données de supervision de l'opérateur ($receive\_supervision\_command$) ;

— *sendingSupervisionData* → *waitingOrder* avec envoi des données de su-

pervision (*send_supervision_reply*) ;

— *waitingOrder* → *idle* lors de la réception de l'ordre d'arrêt (*receive_stop_order*).

Deux invariants sont définis aux états *sendingCommand* et *sendingRetroactionCommand* afin de modéliser les exigences temps-réel du PLC.

Dans le formalisme exposé ci-dessus, cet automate est ainsi exprimé :

```
1  <states> idle, waitingSetpoint, sendingInitialCommand,
       receivingSensorData, sendingRetroactionCommand </states>
2  <initial_state> idle </initial_state>
3
4  <clocks> x </clocks>
5
6  <invariant> sendingCommand, x < 4 </invariant>
7  <invariant> retroaction, x < 4 </invariant>
8
9  <transition> idle, waitingOrder, none, none,
       receive_start_command </transition>
10 <transition> waitingOrder, idle, none, none,
       receive_stop_command </transition>
11 <transition> waitingOrder, sendingInitialCommand, none,
       none, receive_rudder_setpoint </transition>
12 <transition> sendingInitialCommand, receivingSensorData,
       none, x:=0, send_rudder_command </transition>
13 <transition> receivingSensorData, sendingRetroactionCommand
       , rudder_angle != rudder_setpoint, none,
       receive_sensor_data </transition>
14 <transition> sendingRetroactionCommand, receivingSensorData
       , none, x:=0, send_rudder_command </transition>
15 <transition> receivingSensorData, waitingOrder,
       rudder_angle == rudder_setpoint, none,
       receive_sensor_data </transition>
16 <transition> waitingOrder, sendingSupervisionData, none,
       none, receive_supervision_command </transition>
17 <transition> sendingSupervisionData, waitingOrder, none,
       none, send_supervision_reply </transition>
```

A.3 Formalisme de la description séquentielle des missions

Le formalisme retenu pour la description séquentielle des missions est très proche de celui exposé précédemment pour l'expression des automates temporisés modélisant le comportement des composants du système. En effet, une mission est vue comme un ensemble d'étapes (*steps*) mobilisant en parallèle un ensemble de fonctions. Cet ensemble représente les fonctions dont l'exécution par le système est nécessaire afin qu'il passe d'une étape de la mission à une autre (via les *transitions* dans la formalisation de cette description séquentielle :

<steps> $step_0\_name, \ldots, step_n\_name$ </steps>
<initial_step> $initial\_step\_name$ </initial_step>
<final_step> $final\_step\_name$ </final_step>
<transition> $source\_step\_name, destination\_step\_name, function_0, \ldots, function_n$ </transition>

NB : les fonctions référencées dans les balises *transition* devront être nommées en cohérence avec les noms choisis lors de l'élaboration du modèle d'architecture système.

Considérons par exemple une mission de patrouille affectée à un sous-marin que nous nous efforcerons à modéliser de manière très simplifiée, non-exhaustive, et à partir de données publiques. Cette mission peut être résumée à la succession de ces étapes :

1. *Début de mission* : le bâtiment est à quai et l'équipage le prépare pour l'appareillage ;
2. *Appareillage* : le bâtiment quitte le quai ;
3. *Transit* : le bâtiment évolue en surface jusqu'à son point de plongée ;
4. *Plongée* : le bâtiment plonge puis évolue en immersion ;
5. *Transit* : à l'issue de sa patrouille en immersion, le bâtiment opère son retour en surface puis évolue jusqu'à son quai ;
6. *Fin de mission* : le bâtiment est à quai, sa mission prend fin.

Ces étapes mobiliseront les chaînes fonctionnelles suivantes :

— *Appareillage* : seules les chaînes fonctionnelles « vitales » du bâtiment sont mises en œuvre lors de cette phase, à savoir dans notre exemple ses capacités de production d'énergie, d'oxygène et d'eau douce ;

— *Transit* : en plus des fonctions précédemment citées, le sous-marin s'appuie lors de son transit en surface sur ses fonctions de propulsion, gouverne, navigation (c'est-à-dire l'établissement de sa route compte-tenu de ses objectifs opérationnels et de son environnement), perception de l'environnement, et écoute radio ;
— *Plongée* : aux chaînes mobilisées lors de l'étape de transit s'ajoutent les fonctions de variation d'immersion (utilisation des barres de plongée et des ballasts) et d'étanchéité du navire ;
— *Transit* : *cf.* ci-dessus ;
— *Fin de mission* : nous considérons, comme pour l'appareillage, que le bâtiment à quai n'a besoin que des fonctions « vitales » figurant dans cet exemple.

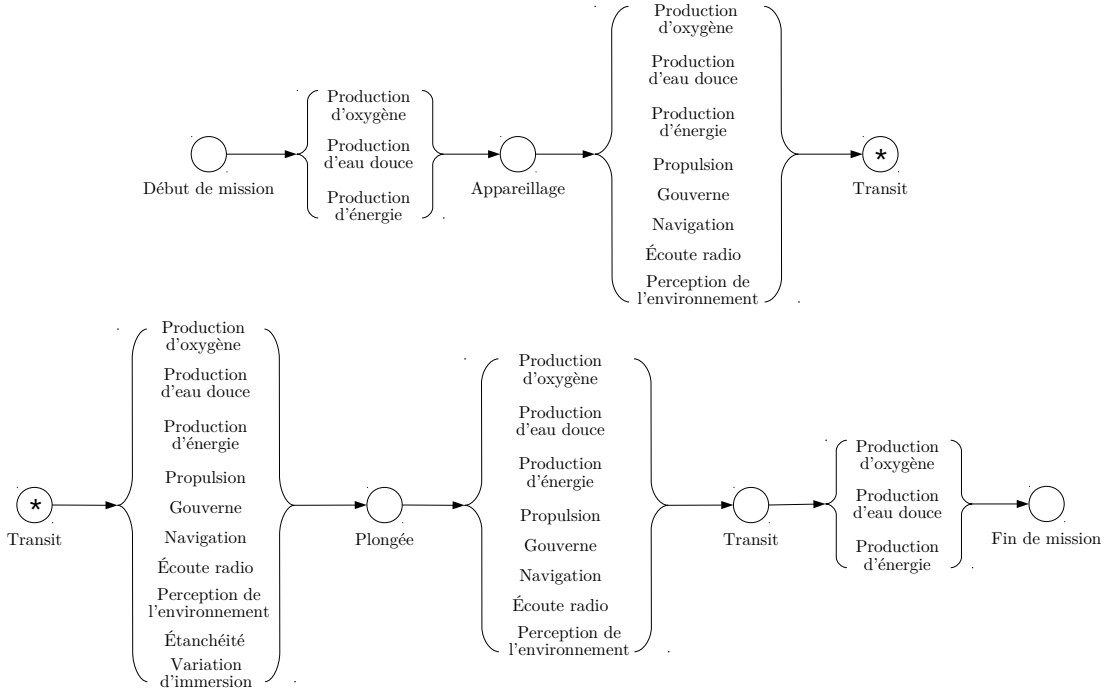


FIGURE A.1 – Exemple de description séquentielle d'une mission

La description séquentielle de cette mission est illustrée en figure A.1 ; sa transcription dans le formalisme proposé est la suivante :

```
1 <steps> missionStart, castingOff, transit0, diving,
    transit1, missionEnd </steps>
2 <initial_step> missionStart </initial_step>
```

```
3 <final_step> missionEnd </final_step>
4
5 <transition> missionStart, castingOff, oxygen_production,
    water_production, energy_production </transition>
6 <transition> castingOff, transit0, oxygen_production,
    water_production, energy_production, propulsion,
    steering, navigation, communication, environment_sensing
7 </transition>
8 <transition> transit0, diving, oxygen_production,
    water_production, energy_production, propulsion,
    steering, navigation, communication, environment_sensing
    ,
9 immersion_variation, waterproofness</transition>
10 <transition> diving, transit1, oxygen_production,
    water_production, energy_production, propulsion,
    steering, navigation, communication, environment_sensing
11 </transition>
12 <transition> transit1, missionEnd, oxygen_production,
    water_production, energy_production </transition>
```

A.4 Formalisme de la liste ordonnée des missions

Ce type de modèle est sans surprise le plus simple parmi les quatre classes élaborées au cours de la tâche 1-2 ; son formalisme consiste en une liste de noms de missions ordonnés selon une priorité décroissante, le nom de la mission la plus critique figurant en premier et celui de la moins critique prenant place en fin de liste. Les missions dont la priorité ne peut être départagée sont séparées par des virgules, tandis que celles auxquelles des niveaux de priorité différents sont affectés sont séparées par des point-virgule. Soit, pour $n + 1$ niveaux de priorité, 0 correspondant à la priorité la plus haute et n à la priorité la plus basse :

$mission_{\{priorite=0\},0}\_name, \ldots, mission_{\{priorite=0\},k}\_name;$
$\ldots;$
$mission_{\{priorite=i\},0}\_name, \ldots, mission_{\{priorite=i\},l}\_name;$
$mission_{\{priorite=i+1\},0}\_name, \ldots, mission_{\{priorite=i+1\},m}\_name;$
$\ldots;$
$mission_{\{priorite=n\},0}\_name, \ldots, mission_{\{priorite=n\},p}\_name;$

(Les indices signifiant $k+1$ missions de criticité 0, $l+1$ missions de criticité i, $m+1$ missions de criticité $i+1$ et $p+1$ missions de criticité n).

Dans le cas d'espèce, eu égard à la simplicité de cette classe de modèles, l'exemple est dispensable.

Annexe B

À propos de la dynamique d'un système naval

Table des matières

Une simplification de la dynamique du navire en évolution, inspirée du modèle proposé par Luc Jaulin [Jau04], amène à considérer qu'il est soumis aux forces suivantes (exception faite de celles s'exerçant parallèlement au champ de gravité) :

- la **poussée** f_p générée par ses hélices ; nous considérons que dans le cas d'un bâtiment possédant deux lignes d'arbre, elle s'applique au milieu du segment reliant les deux hélices étant donné leur distribution symétrique axiale. La poussée est orientée selon l'axe du navire ;
- la **portance du gouvernail** f_v, orientée perpendiculairement à la surface du gouvernail ;
- les **frottements** exercés par l'eau, possédant une composante tangentielle f_{rt} opposée à la marche du navire et une composante angulaire f_{ra} opposée à sa rotation. Ces frottements sont supposés visqueux, c'est-à-dire proportionnels à la vitesse du navire.

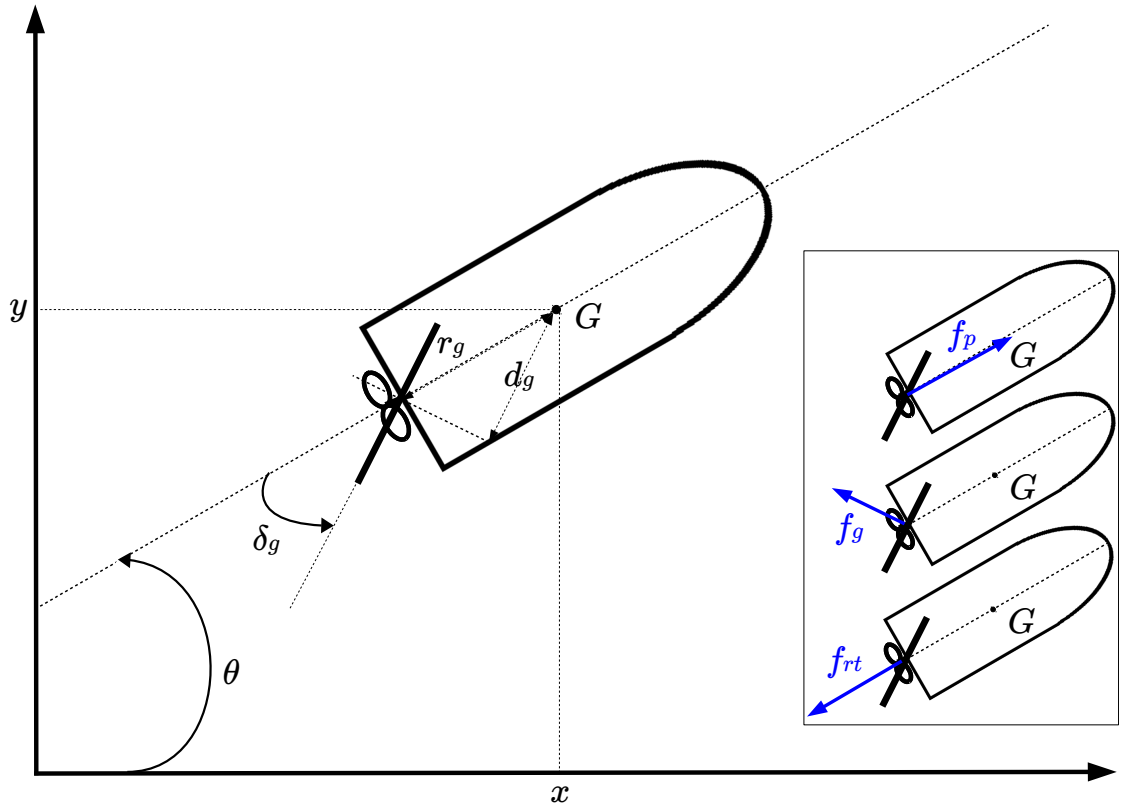


FIGURE B.1 – Dynamique simplifiée d'un navire en évolution dans le référentiel terrestre (inspiré de [Jau04]).

B.1 Étude de la dynamique en translation

La translation du bâtiment est permise par la poussée de l'appareil propulsif f_p et contrariée par la portance du gouvernail f_g et les frottements tangentiels f_{rt}. Ces trois forces peuvent s'exprimer ainsi, de manière simplifiée [Tec04 ; Jau04] :

$$f_p = K_t \rho n^2 D^4 \tag{B.1}$$

$$f_g = \alpha_g v \sin(\delta_g) \tag{B.2}$$

$$f_{rt} = \alpha_f v \tag{B.3}$$

où K_t est le coefficient de poussée de l'hélice, D est son diamètre et n sa vitesse de rotation, α_g est le coefficient de portance du gouvernail et α_f est le coefficient

de frottement tangentiel du navire.

L'application du principe fondamental de la dynamique en translation nous donne :

$$m\dot{v} = f_p - f_g \sin(\delta_g) - f_{rt} \tag{B.4}$$

où m est la masse du navire, soit :

$$m\dot{v} = K_t \rho n^2 D^4 - \alpha_g v \sin^2(\delta_g) - \alpha_f v \tag{B.5}$$

B.2 Étude de la dynamique en rotation

Similairement, les forces qui entraînent et entravent la rotation du navire autour de l'axe vertical passant par son centre de gravité sont la force f_g exercée par l'eau sur le gouvernail, et les frottements angulaires f_{ra}. La poussée de l'hélice f_p n'agit pas directement sur la rotation du bâtiment, son moment étant nul étant donné que sa droite d'action passe par un des points de l'axe de rotation. Nous introduisons le coefficient de frottement angulaire α_θ afin d'exprimer la force afférente [Jau04] :

$$f_{ra} = \alpha_\theta \omega \tag{B.6}$$

Le principe fondamental de la dynamique en rotation nous permet d'établir que :

$$J\dot{\omega} = r_g \cos\delta_g f_g - f_{ra} \tag{B.7}$$

Soit :

$$J\dot{\omega} = r_g \alpha_g v \cos(\delta_g)\sin(\delta_g) - \alpha_\theta \omega \tag{B.8}$$

B.3 Équations d'évolution des grandeurs liées au système

Nous en déduisons donc les équations suivantes, nous permettant de modéliser l'évolution des grandeurs descriptives de la dynamique du système :

$$\begin{cases} \dot{x} = v cos(\theta) \\ \dot{y} = v sin(\theta) \\ \dot{\theta} = \omega \\ \dot{v} = \frac{K_t \rho n^2 D^4 - \alpha_g v \sin^2(\delta_g) - \alpha_f v}{m} \\ \dot{\omega} = \frac{r_g \alpha_g v \cos(\delta_g)\sin(\delta_g) - \alpha_\theta \omega}{J} \end{cases} \tag{B.9}$$

Nous voyons à la lecture de ces équations que la dynamique du bâtiment est commandable à partir des entrées des actionneurs considérés dans le cadre de cette étude, à savoir :

— l'angle de barre δ_g ;
— la vitesse de rotation du moteur Ω_e ;
— le rapport de réduction r_e ;

ces deux dernières variables étant présentes dans la quatrième équation du système précédent étant donné que la vitesse n de rotation de l'hélice peut s'exprimer par la relation :

$$n = r_e \Omega_e \tag{B.10}$$

B.4 Axes d'amélioration

Si ce modèle offre l'avantage de la simplicité, il n'est pas exhaustif. Il a par exemple été construit en supposant nuls les angles de roulis et de tangage, pourtant utiles dans le contexte de nos travaux étant donné qu'une attaque pourrait consister à « jouer » sur le moment de chavirement du bâtiment (par le biais de l'équilibrage de sa cargaison, notamment) pour provoquer un déséquilibre du navire.

La prise en compte des dérives liées au vent et au courant serait également nécessaire afin d'atteindre un degré de fidélité plus élevé.

Annexe C

Modèles UPPAAL utilisés

Table des matières

C.1 Sous-système de gouverne (chapitres 1 & 2) : automates et déclarations

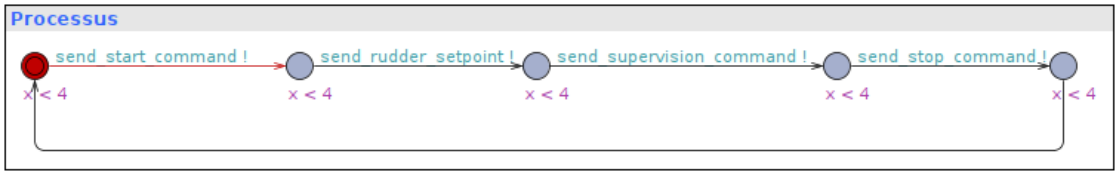


FIGURE C.1 – Automate modélisant un processus de gouverne élémentaire.

```
1 // Place global declarations here.
2
3 //synchronisations
```

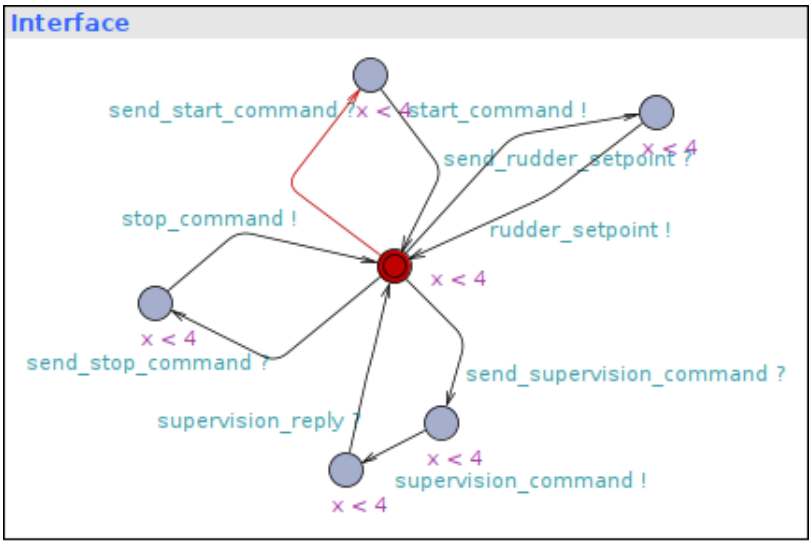


FIGURE C.2 – Automate modélisant l'interface utilisateur.

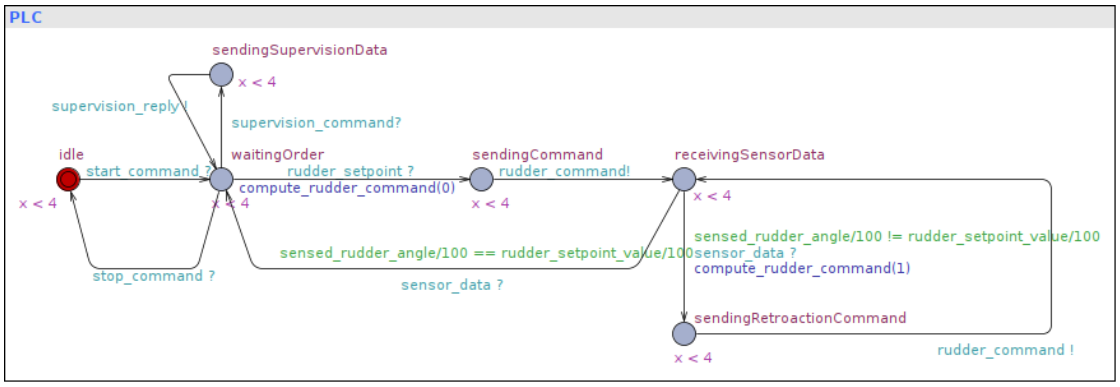


FIGURE C.3 – Automate modélisant le PLC commandant l'appareil à gouverner.

```
4  chan start_command;
5  chan stop_command;
6  chan rudder_setpoint;
7  chan rudder_command;
8  chan sensor_data;
9  chan sense_command;
10 chan supervision_command;
11 chan supervision_reply;
12 chan send_start_command;
13 chan send_stop_command;
```

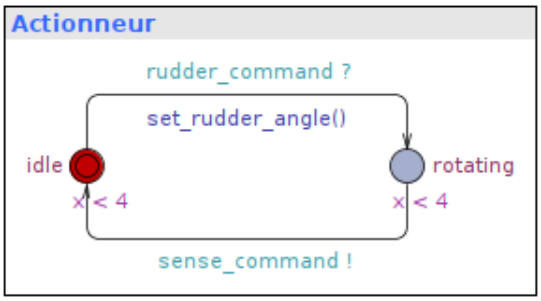


FIGURE C.4 – Automate modélisant l'actionneur du gouvernail.

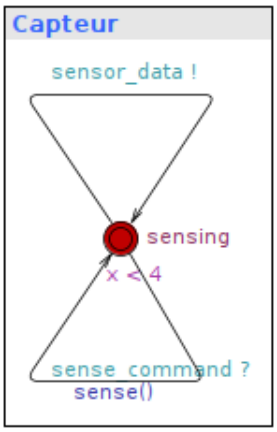


FIGURE C.5 – Automate modélisant le capteur d'angle de barre.

```
14 chan send_rudder_setpoint;
15 chan send_supervision_command;
16
17 //variables
18 int rudder_setpoint_value = 4500 ;
19 int rudder_command_value ;
20 int rudder_angle;
21 int sensed_rudder_angle;
22
23 //fonctions
24 void sense(){
25     sensed_rudder_angle = rudder_angle ;
26 }
27
28
```

```
29 //fonctions de commande : premiere version (abstraction d'
      un regulateur proportionnel)
30 /**
31 void compute_rudder_command(int p){
32     if (p == 0){
33         rudder_command_value = rudder_setpoint_value/2 ;
34     }
35     else {
36         rudder_command_value = rudder_setpoint_value + 85*(
              rudder_setpoint_value - sensed_rudder_angle)/100
               ;
37     }
38 }
39 **/
40
41 //seconde version (abstraction de l'effet final d'un
      regulateur PID)
42 void compute_rudder_command(int p){
43     if (p == 0){
44         rudder_command_value = rudder_setpoint_value ;
45     }
46     else {
47         rudder_command_value = 101*rudder_setpoint_value
              /100 ;
48     }
49 }
50
51 void set_rudder_angle(){
52     rudder_angle = rudder_command_value -
          rudder_command_value/100 ;
53 }
```

C.2 Automates topologiques

C.2.1 Automates

Les automates utilisés dans le cadre du modèle « nominal » sont représentés par les figures C.6 et C.7.

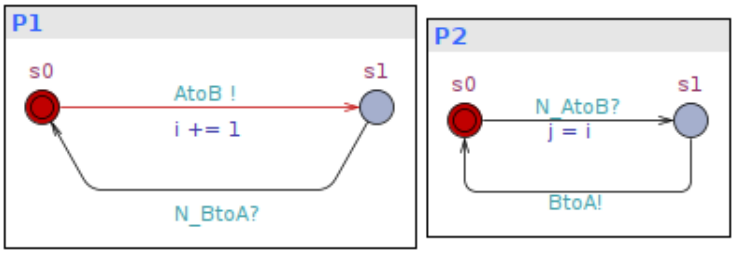


FIGURE C.6 – Automates modélisant deux composants s'échangeant une variable entière i.

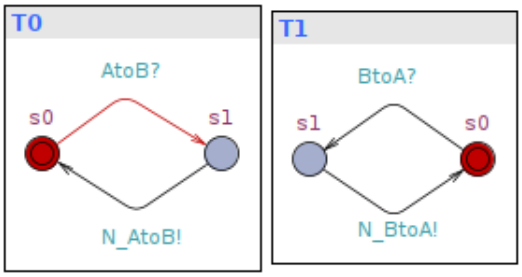


FIGURE C.7 – Automates modélisant les liens topologiques unissant les deux composants.

Après application des mutations décrites en 2.3.2.1, ces automates évoluent de sorte à obtenir les modèles représentés par les figures C.8 et C.9.

C.2.2 Déclarations

```
1 // Place global declarations here.
2 chan AtoB ;
3 chan BtoA ;
4 chan N_AtoB ;
5 chan N_BtoA ;
6 //canal de synchronisation servant a eviter les
    interblocages lors de la mutation presentee
```

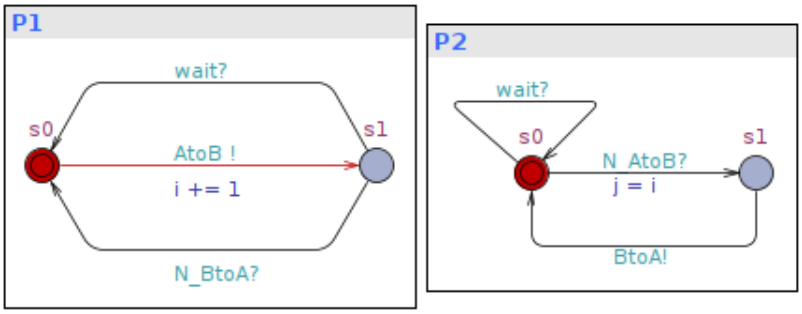


FIGURE C.8 – Automates modélisant deux composants s'échangeant une variable entière i, après mutation.

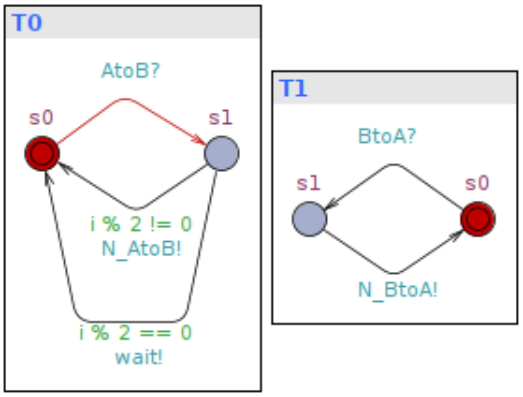


FIGURE C.9 – Automates modélisant les liens topologiques unissant les deux composants, après mutation.

```
7  broadcast chan wait;
8
9  int i = 0;
10 int j = 0;
```

C.2.3 Propriétés

```
1 A[] T0.s0 imply i == j
```

Bibliographie

[AEK09] A. Abou El Kalam. « Security of critical infrastructures and Networks ». Habilitation à Diriger les Recherches. Institut National Polytechnique de Toulouse - INPT, déc. 2009. URL : https://tel.archives-ouvertes.fr/tel-00440784.

[Aic+14] B. K. Aichernig, K. Hörmaier et F. Lorber. « Debugging with Timed Automata Mutations ». In : *Computer Safety, Reliability, and Security : 33rd International Conference, SAFECOMP 2014, Florence, Italy, September 10-12, 2014. Proceedings*. Sous la dir. d'Andrea Bondavalli et Felicita Di Giandomenico. Cham : Springer International Publishing, 2014, p. 49-64. ISBN : 978-3-319-10506-2. DOI : 10.1007/978-3-319-10506-2_4. URL : http://dx.doi.org/10.1007/978-3-319-10506-2_4.

[Air40] Sir G. B. Airy. « On the Regulator of the Clock-work for effecting uniform Movement of Equatoreals ». In : *Monthly Notices of the Royal Astronomical Society*. Mar. 1840.

[And15] E. André. « What's decidable about parametric timed automata ? » In : *International Workshop on Formal Techniques for Safety-Critical Systems*. Springer. 2015, p. 52-68.

[Beh+04] G. Behrmann, A. David et K. G. Larsen. « A tutorial on uppaal ». In : *Formal methods for the design of real-time systems*. Springer. 2004, p. 200-236.

[Ber38] D. Bernoulli. *Hydrodynamica, sive de Viribus et Motibus Fluidorum Commentarii*. Sumptibus Johannis Reinholdi Dulseckeri, 1738. URL : https://books.google.fr/books?id=uPXMSGwlTlQC.

[Ber+08] B. Berthomieu, J.-P. Bodeveix, P. Farail, M. Filali, H. Garavel, P. Gaufillet, F. Lang et F. Vernadat. « Fiacre : an Intermediate Language for Model Verification in the Topcased Environment ». In : *ERTS 2008*. Toulouse, France, jan. 2008. URL : https://hal.inria.fr/inria-00262442.

[Bis+01] A. M. Bisantz, E. Roth, B. Brickman, L. L. Gosbee, L. Hettinger et J. McKinney. « Integrating Cognitive Analyses into a Large Scale System Design Process ». In : *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 45.4 (2001), p. 434-438.

[Bis09] C. Bissell. « A History of Automatic Control ». In : *Springer Handbook of Automation*. Sous la dir. de Shimon Y. Nof. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009, p. 53-69. ISBN : 978-3-540-78831-7. DOI : 10.1007/978-3-540-78831-7_4. URL : https://doi.org/10.1007/978-3-540-78831-7_4.

[Bou60] G. Bouligand. « L'œuvre d'Euler et la mécanique des fluides au XVIIIe siècle ». In : *Revue d'histoire des sciences et de leurs applications* 13.2 (1960), p. 105 -113. URL : https://www.persee.fr/doc/rhs_0048-7996_1960_num_13_2_3806.

[BS03] B. Brykczynski et R. A. Small. « Reducing Internet-based intrusions : Effective security patch management ». In : *IEEE Software* 20.1 (2003), p. 50-57. ISSN : 0740-7459. DOI : 10.1109/MS.2003.1159029.

[Bur+05] C. M. Burns, D. J. Bryant et B. A. Chalmers. « Boundary, Purpose, and Values in Work-Domain Models : Models of Naval Command and Control ». In : *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans* 35.5 (2005), p. 603-616. ISSN : 1083-4427.

[Cha+05] C.-W. Chang, D.-R. Tsai et J.-M. Tsai. « A cross-site patch management model and architecture design for large scale heterogeneous environment ». In : *Security Technology, 2005. CCST '05. 39th Annual 2005 International Carnahan Conference on*. 2005, p. 41-46. DOI : 10.1109/CCST.2005.1594837.

[Che+03] S. Cheung, U. Lindqvist et M. W. Fong. « Modeling multistep cyber attacks for scenario recognition ». In : *Proceedings DARPA Information Survivability Conference And Exposition*. T. 1. IEEE. 2003, p. 284-292.

[Cla+01] E. Clarke, O. Grumberg, S. Jha, Y. Lu et H. Veith. « Progress on the state explosion problem in model checking ». In : *Informatics*. Springer. 2001, p. 176-194.

[Cla09] E. M. Clarke. « My 27-year Quest to Overcome the State Explosion Problem ». In : *2009 24th Annual IEEE Symposium on Logic In Computer Science*. 2009, p. 3-3.

[CC18] CMA-CGM. *Le CMA CGM ANTOINE DE SAINT EXUPERY a été inauguré par Bruno Le Maire, Ministre de l'Economie et des Finances, avec Elisabeth Borne, Ministre chargée des Transports*. 2018. URL : https://www.cma-cgm.fr/detail-news/2190/le-cma-cgm-antoine-de-saint-exupery-a-ete-inaugure-par-bruno-le-maire-ministre-de-l-economie-et-des-finances-avec-elisabeth-borne-ministre-chargee-des-transports (visité le 15/04/2020).

[Cos18] B. Coste. « Détection contextuelle de cyberattaques par gestion de confiance à bord d'un navire ». Thèse de doct. Ecole Nationale Supérieure Mines-Télécom Atlantique Bretagne-Pays de la Loire, déc. 2018.

[Dav+15] A. David, K. G. Larsen, A. Legay, M. Mikučionis et D. B. Poulsen. « Uppaal SMC tutorial ». In : *International Journal on Software Tools for Technology Transfer* 17.4 (2015), p. 397-415.

[Dha+11] P. Dhaussy, J. C. Roger et F. Boniol. « Reducing State Explosion with Context Modeling for Model-Checking ». In : *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering*. 2011, p. 130-137. DOI : 10.1109/HASE.2011.24.

[DIC14] DICOD. *Dernière cérémonie des couleurs de la frégate Georges Leygues*. 2014. URL : https://www.defense.gouv.fr/actualites/articles/derniere-ceremonie-des-couleurs-de-la-fregate-georges-leygues (visité le 15/04/2020).

[Div08] U.S. DHS National Cyber Security Division. *National Cyber Security Division, Recommended Practice for Patch Management of Control Systems*. 2008.

[DoD95] U.S. DoD. *Department of Defense Handbook : System Security Engineering – Program Management Requirements, MIL-HDBK-1785*. 1995.

[Dup19] F. DUPONT. *Commandant de sous-marins*. Autrement, 2019. ISBN : 9782746754409. URL : https://books.google.fr/books?id=MBezDwAAQBAJ.

[Ele04] General ELECTRIC. *GE's LM2500+ Gas Turbines Power Queen Mary 2 - The World's Largest Transatlantic Liner*. 2004. URL : https://www.geaviation.com/press-release/marine-industrial-engines/ges-lm2500-gas-turbines-power-queen-mary-2-worlds-largest (visité le 11/06/2020).

[Fen+15] D. FENG, J. XIAO, S. WEI et L. ZHANG. « Design of Integrated Ship Equipment Monitoring and Controlling System Based on Ethernet ». In : *Marine Engineering Frontiers* 3 (jan. 2015). DOI : 10.14355/mef.2015.03.002.

[Fri] C. FRIBOURG. *La technologie des réacteurs de propulsion navale*. URL : https://inis.iaea.org/collection/NCLCollectionStore/_Public/33/048/33048066.pdf (visité le 15/04/2020).

[GL92] R. GARDINER et A.D. LAMBERT. *Steam, steel & shellfire : the steam warship 1815-1905*. Conway's history of the ship. Naval Institute Press, 1992, p. 37. ISBN : 9781557507747. URL : https://books.google.fr/books?id=qHoSAQAAMAAJ.

[Gar18] C. GARNIER. *L'armateur français CMA CGM inaugure son nouveau navire amiral au Havre*. 2018. URL : https://www.usinenouvelle.com/article/l-armateur-francais-cma-cgm-inaugure-son-nouveau-navire-amiral-au-havre.N738199 (visité le 15/04/2020).

[Gau+17] A. GAUCI, S. MICHELIN et M. SALLES. « Addressing the challenge of cyber security maintenance through patch management ». In : *CIRED - Open Access Proceedings Journal* 2017.1 (2017), p. 2599-2601.

[GM17a] A. GIRARD et C. MANDA. *Projet Vulnerobot*. 2017. URL : https://github.com/linuxisnotunix/Vulnerobot (visité le 15/04/2020).

[GM17b] A. GIRARD et C. MANDA. *Projet Vulnerobot – Configuration*. 2017. URL : https://github.com/linuxisnotunix/Vulnerobot/wiki/Configuration-file (visité le 23/07/2020).

[GG+17] G. Gonzalez-Granadillo, J. Garcia-Alfaro et H. Debar. « A polytope-based approach to measure the impact of events against critical infrastructures ». In : *Journal of Computer and System Sciences* 83.1 (2017), p. 3 -21. ISSN : 0022-0000. DOI : https://doi.org/10.1016/j.jcss.2016.02.004. URL : http://www.sciencedirect.com/science/article/pii/S0022000016000192.

[Gro17] V. Groizeleau. *Charles de Gaulle : Plus qu'un arrêt technique, une refonte*. 2017. URL : https://www.meretmarine.com/fr/content/charles-de-gaulle-plus-quun-arret-technique-une-refonte-0 (visité le 15/04/2020).

[Gu+19] R. Gu, R. Marinescu, C. Seceleanu et K. Lundqvist. « Towards a Two-Layer Framework for Verifying Autonomous Vehicles ». In : *NASA Formal Methods*. Sous la dir. de J. M. Badger et K. Y. Rozier. Cham : Springer International Publishing, 2019, p. 186-203. ISBN : 978-3-030-20652-9.

[Guy+13] C. Guychard, S. Guerin, A. Koudri, F. Dagnat et A. Beugnard. « Conceptual interoperability through Models Federation ». In : *Semantic Information Federation Community Workshop*. 2013.

[Hol+08] J. Holsopple, S. Yang et B. Argauer. « Virtual terrain : a security-based representation of a computer network ». In : *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*. T. 6973. International Society for Optics et Photonics. 2008, 69730E.

[Iva+18] D. Ivanov, K. G. Larsen, S. Schupp et J. Srba. « Analytical Solution for Long Battery Lifetime Prediction in Nonadaptive Systems ». In : *Quantitative Evaluation of Systems*. Sous la dir. d'A. McIver et A. Horvath. Cham : Springer International Publishing, 2018, p. 173-189. ISBN : 978-3-319-99154-2.

[Jag] S Jagannathan. « PID Based Rudder Controller ». In : ().

[Jak11a] G. Jakobson. « Extending situation modeling with inference of plausible future cyber situations ». In : *2011 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. 2011, p. 48-55.

[Jak11b] G. JAKOBSON. « Mission cyber security situation assessment using impact dependency graphs ». In : *14th International Conference on Information Fusion*. 2011, p. 1-8.

[Jak13] G. JAKOBSON. « Mission-centricity in cyber security : Architecting cyber attack resilient missions ». In : *2013 5th International Conference on Cyber Conflict (CYCON 2013)*. 2013, p. 1-18.

[Jau04] L. JAULIN. « Modélisation et commande d'un bateau à voile ». In : *CIFA'2004 (Conférence Internationale Francophone d'Automatique)*. Douz, Tunisia, nov. 2004. URL : https://hal.archives-ouvertes.fr/hal-00861485.

[Jaw93] L. M. JAWORSKI. « Tandem Threat Scenarios : a Risk Assessment Approach ». In : *16th National Computer Security Conference : Proceedings*. 1993, p. 155-164.

[Kan+07] W. KANOUN, N. CUPPENS-BOULAHIA, F. CUPPENS et F. AUTREL. « Advanced reaction using risk assessment in intrusion detection systems ». In : *International Workshop on Critical Information Infrastructures Security*. Springer. 2007, p. 58-70.

[Kan+08] W. KANOUN, N. CUPPENS-BOULAHIA, F. CUPPENS et J. ARAUJO. « Automated reaction based on risk analysis and attackers skills in intrusion detection systems ». In : *2008 Third International Conference on Risks and Security of Internet and Systems*. IEEE. 2008, p. 117-124.

[KC13] I. KOTENKO et A. CHECHULIN. « A cyber attack modeling and impact assessment framework ». In : *2013 5th International Conference on Cyber Conflict (CYCON 2013)*. IEEE. 2013, p. 1-24.

[KT15] S. Narayanan KRISHNA et A. TRIVEDI. « Hybrid automata for formal modeling and verification of cyber-physical systems ». In : *arXiv preprint arXiv :1503.04928* (2015).

[LL96] L. WANG et R. LANGARI. « Complex systems modeling via fuzzy logic ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), p. 100-106.

[Lar+97] K. G. LARSEN, P. PETTERSSON et W. YI. « UPPAAL in a nutshell ». In : *International journal on software tools for technology transfer* 1.1-2 (1997), p. 134-152.

[LD94] A. Le Duff. *Un saut technologique majeur*. 1994. URL : https://lemarin.ouest-france.fr/sites/default/files/2015/03/27/page_10_13_mai_1994.pdf (visité le 15/04/2020).

[LRd70] J. Le Rond d'Alembert. *Traité de l'équilibre et du mouvement des fluides : pour servir de suite au traité de dynamique*. Briasson, 1770. URL : https://books.google.fr/books?id=rnAOAAAAQAAJ.

[LV17] S. Le Vey. *« Cyber sécurité», renforcer la protection des systemes industriels du navire*. Jan. 2017. URL : https://www.ecologique-solidaire.gouv.fr/sites/default/files/Guide%202%20-%20Cyber%20securit%C3%A9%20-%20Renforcer%20la%20protection%20des%20syst%C3%A8mes%20industriels%20du%20navire.pdf (visité le 15/04/2020).

[LL04] J. Leigh et J.R. Leigh. *Control Theory*. Control Theory. Institution of Electrical Engineers, 2004, p. 250. ISBN : 9780863413322. URL : https://books.google.fr/books?id=3P1zTw1HmyIC.

[Lin10] Michael B. Lindsey. « A Method for Estimating the Financial Impact of Cyber Information Security Breaches Utilizing the Common Vulnerability Scoring System and Annual Loss Expectancy ». In : (2010).

[Lio96] J.-L. Lions. *Ariane 5 Flight 501 Failure – Report by the Inquiry Board*. Juil. 1996. URL : https://esamultimedia.esa.int/docs/esa-x-1819eng.pdf (visité le 15/04/2020).

[Liu+09] S. Liu, R. Kuhn et H. Rossman. « Surviving Insecure IT : Effective Patch Management ». In : *IT Professional* 11.2 (2009), p. 49-51.

[LJ13] Z. Liu et H. Jin. « Extended radiated energy method and its application to a ship roll stabilisation control system ». In : *Ocean engineering* 72 (2013), p. 25-30.

[Mad19] H. R. Madala. *Inductive learning algorithms for complex systems modeling*. CRC press, 2019.

[Mar18] SIRPA Marine. *Aquitaine (D650)*. 2018. URL : https://www.defense.gouv.fr/marine/equipements/batiments-de-combat/fregates/fregates-multimissions/aquitaine-d-650 (visité le 15/04/2020).

[Mat+07] P. Matiæ, R. Antoniæ et I. Komar. « Marine diesel engine governor identification ». In : *IFAC Proceedings Volumes* 40.17 (2007), p. 385-389.

[Max68] J. C. Maxwell. « On governors ». In : *Proceedings of the Royal Society of London* (1868), p. 270 -283.

[ML17] P. Merino Laso. « Detection of dysfunctions and malveillant acts based on multi-sensor data quality models ». Thèse de doct. Ecole Nationale Supérieure Mines-Télécom Atlantique Bretagne-Pays de la Loire, déc. 2017. URL : https://tel.archives-ouvertes.fr/tel-01813616.

[MK15] J.P. Mittal et I. Kaur. *Encyclopaedia of Technical Education*. T. 19 – Naval Architecture and Marine Engineering. Encyclopaedia of Technical Education. Mittal Publications, 2015, p. 39. ISBN : 9788170994497. URL : https://books.google.fr/books?id=H1H54pr3Cr4C.

[Mui19] D. Muilenburg. *Boeing CEO Dennis Muilenburg Addresses the Ethiopian Airlines Flight 302 Preliminary Report*. 2019. URL : https://investors.boeing.com/investors/investor-news/press-release-details/2019/Boeing-CEO-Dennis-Muilenburg-Addresses-the-Ethiopian-Airlines-Flight-302-Preliminary-Report/default.aspx (visité le 15/04/2020).

[Mus+10] S. Musman, A. Temin, M. Tanner, D. Fox et B. Pridemore. « Evaluating the impact of cyber attacks on missions ». In : *International Conference on Cyber Warfare and Security*. Academic Conferences International Limited. 2010, p. 446.

[Mus+11] S. Musman, M. Tanner, A. Temin, E. Elsaesser et L. Loren. « Computing the impact of cyber attacks on complex missions ». In : *2011 IEEE International Systems Conference*. 2011, p. 46-51.

[New17] Insurance Marine News. *Marine accident round-up : 17th July 2017*. 2017. URL : https://insurancemarinenews.com/insurance-marine-news/marine-accident-round/ (visité le 25/04/2020).

[NOR] NORIS. *Solutions for Marine Automation*. URL : https://www.noris-group.com/fileadmin/cms_upload/en/Resources/BR-NGR-Marine-Overview-EN.pdf (visité le 11/06/2020).

[Nyl16] M. Nylund. « Construction of a hardware-in-the-loop simulator for Azipod control system testing ; Konstruktion av en hårdvara i loopen simulator för testning av Azipod-styrsystemet ». en. G2 Pro gradu, diplomityö. 2016-08-24, p. 90+9. URL : http://urn.fi/URN:NBN:fi:aalto-201608263084.

[Pat+08] Sandip C. Patel, James H. Graham et Patricia A.S. Ralston. « Quantitatively assessing the vulnerability of critical information systems : A new method for evaluating security enhancements ». In : *International Journal of Information Management* 28.6 (2008), p. 483-491.

[Pau+17] W. Pauquet, J. Bercy et M. Benedittini. *Cybersécurité dans le milieu maritime : défis et pistes de solutions*. CEIS, 2017.

[Pre15] J.-F. Preveraud. *Le plus gros porte-conteneurs du monde arrive en Europe*. 2015. URL : https://www.industrie-techno.com/article/le-plus-gros-porte-conteneurs-du-monde-arrive-en-europe.35593 (visité le 15/04/2020).

[Ras85] J. Rasmussen. « The role of hierarchical knowledge representation in decisionmaking and system management ». In : *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15.2 (1985), p. 234-243.

[Rob08] G. N. Roberts. « Trends in marine control systems ». In : *Annual reviews in control* 32.2 (2008), p. 263-269.

[Sam+13] L. Samarji, F. Cuppens, N. Cuppens-Boulahia, W. Kanoun et S. Dubus. « Situation calculus and graph based defensive modeling of simultaneous attacks ». In : *Cyberspace safety and security*. Springer, 2013, p. 132-150.

[Sch99] B. Schneier. « Attack trees ». In : *Dr. Dobb's journal* 24.12 (1999), p. 21-29.

[Sch15] B. Schneier. *Secrets and lies : digital security in a networked world*. John Wiley & Sons, 2015.

[Sek+14] T. Sekizawa, K. Okano, A. Ogawa et S. Kusumoto. « Verification of a control program for a line tracing robot using UPPAAL considering general aspects ». In : *International Journal of Informatics Society (IJIS)* 6.2 (2014), p. 79-87.

[SM14] S. Shah et B. Mehtre. « An overview of vulnerability assessment and penetration testing techniques ». In : *Journal of Computer Virology and Hacking Techniques* 11 (fév. 2014), p. 27-49. doi : 10.1007/s11416-014-0231-x.

[SB18] H. M. Z. A. Shebli et B. D. Beheshti. « A study on penetration testing process and tools ». In : *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. 2018, p. 1-7.

[SA16] P. S. Shinde et S. B. Ardhapurkar. « Cyber security analysis using vulnerability assessment and penetration testing ». In : *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*. 2016, p. 1-5.

[Smi10] V. Smil. *Two Prime Movers of Globalization : The History and Impact of Diesel Engines and Gas Turbines*. MIT Press, 2010, p. 71. isbn : 9780262014434. url : https://books.google.fr/books?id=nmDGlX0UncsC.

[Soa14] C. Soares. *Gas Turbines : A Handbook of Air, Land and Sea Applications*. Butterworth-Heinemann, 2014. isbn : 978-0-12-410461-7.

[Ste+13] R. Steenbergen, P.H.A.J.M. Gelder, S. Miraglia et T. Vrouwenvelder. *Safety, Reliability and Risk Analysis : Beyond the Horizon*. Sept. 2013, p. 1045. isbn : 9781138001237.

[Sul+16] B. Sultan, F. Dagnat et C. Fontaine. « Maîtrise des Correctifs de Sécurité pour les Systèmes Navals ». In : *CIEL 2016 : 5ème Conférence en Ingénierie du Logiciel*. 2016, p. 1-6.

[Sul+18] B. Sultan, F. Dagnat et C. Fontaine. « A Methodology to Assess Vulnerabilities and Countermeasures Impact on the Missions of a Naval System ». In : *Computer Security*. Sous la dir. de Sokratis K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J. Mylopoulos, A. Antón et S. Gritzalis. Cham : Springer International Publishing, 2018, p. 63-76. isbn : 978-3-319-72817-9.

[Sun+15] X. Sun, A. Singhal et P. Liu. « Who Touched My Mission : Towards Probabilistic Mission Impact Assessment ». In : *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*. SafeConfig '15. Denver, Colorado, USA : ACM, 2015, p. 21-

26. ISBN : 978-1-4503-3821-9. DOI : 10.1145/2809826.2809834. URL : http://doi.acm.org/10.1145/2809826.2809834.

[Tec04] A. H. TECHET. *Hydrodynamics for Ocean Engineers*. 2004. URL : http://web.mit.edu/13.012/www/handouts/propellers_reading.pdf.

[Tuo+16] M. TUO, X. ZHOU, G. YANG et N. FU. « An Approach for Safety Analysis of Cyber-Physical System Based on Model Transformation ». In : *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2016, p. 636-639.

[WF04] P.P. WALSH et P. FLETCHER. *Gas Turbine Performance*. Wiley, 2004, p. 25. ISBN : 9780632064342. URL : https://books.google.fr/books?id=zxRdacCvjVsC.

[Wan+13] Q. WANG, X. ZHOU, G. YANG et Y. YANG. « Behavior Modeling of Cyber-physical System Based on Discrete Hybrid Automata ». In : *2013 IEEE 16th International Conference on Computational Science and Engineering*. 2013, p. 680-684.

[Wie+18] O. WIEVIORKA, H. DREVILLON, X. HÉLARY, B. DERUELLE, A. CREPIN et B. GAINOT. *Histoire militaire de la France*. vol. 1. Place des éditeurs, 2018. ISBN : 9782262076818. URL : https://books.google.fr/books?id=6mNoDwAAQBAJ.

[WR15] R. WINFIELD et S.S. ROBERTS. *French Warships in the Age of Sail, 1786-1861 : Design, Construction, Careers and Fates*. Naval Institute Press, 2015. ISBN : 9781591146292. URL : https://books.google.fr/books?id=kUONrgEACAAJ.

[ZM18] B. ZOHURI et P. MCDANIEL. « Energy Resources and the Role of Nuclear Energy ». In : *Combined Cycle Driven Efficiency for Next Generation Nuclear Power Plants : An Innovative Design Approach*. Cham : Springer International Publishing, 2018, p. 85-104. ISBN : 978-3-319-70551-4. DOI : 10.1007/978-3-319-70551-4_5. URL : https://doi.org/10.1007/978-3-319-70551-4_5.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Maîtrise des correctifs de sécurité pour les systèmes navals

Mots clés : Analyse de risque – Cybersécurité – Vulnérabilités – Correctifs

Résumé : Évoluant dans des environnements contraints et rassemblant dans des espaces réduits des sous-systèmes fortement critiques et hétérogènes, les navires d'aujourd'hui font partie des objets les plus complexes qui soient. À l'heure où croît à leur bord le nombre de systèmes informatiques contrôlant parfois des actionneurs d'importance cruciale, leur maintien en condition de sécurité est une problématique majeure. Les travaux présentés dans cette thèse définissent un processus de gestion des vulnérabilités et des contremesures associées adapté au contexte des systèmes industriels complexes. Ce processus s'appuie sur une méthode et un formalisme de modélisation de ces systèmes permettant d'abstraire leur comportement, discret ou continu, et leurs évolutions éventuelles – apparition d'une vulnérabilité, déploiement d'une contremesure ou survenue d'une attaque – que nous avons définis dans le cadre de ces travaux. Il repose également sur une méthode de calcul des impacts associés aux vulnérabilités, attaques et contremesures aboutissant à leur expression sous la forme d'une métrique adaptée pour la prise de décision. Ce calcul étant permis par la modélisation du système mais aussi par celle des vulnérabilités, attaques et contremesures, nous introduisons par ailleurs une méthode et un formalisme adapté – les mutations d'automates et de réseaux d'automates – permettant leur abstraction. Ces propositions théoriques et méthodologiques sont enfin confrontées à une expérimentation sur un cas fictif représentatif d'un système de propulsion et de gouverne d'un bâtiment de type ferry ou paquebot, permettant de discuter de leur pertinence et de leurs limites ainsi que d'esquisser les perspectives de recherche naissant de nos travaux.

Title: Patch management applied to naval systems

Keywords: Risk analysis – Cybersecurity – Vulnerabilities – Patches

Abstract: Operating in constrained environments and composed of heterogeneous subsystems, today's ships are among the most complex objects that exist. Due to the increasing number of cyber assets among their components, patch and vulnerability management applied to naval systems is an essential process. The work detailed in this PhD thesis aims to define such a process tailored to complex cyber-physical systems. This process relies on a modelling method and formalism allowing to depict CPS behaviour and cyber events – a vulnerability discovery, a cyber attack occurrence or a patch deployment. It also relies on an impact assessment method, allowing to compute the effects of cyber events on CPS ability to fulfill their missions. These impacts are expressed through a specially designed metric aiming to help in decision-making. The process, methods, formalisms and metrics we propose in this work are then evaluated through an experimentation based on a fictitious case-study.