THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :
Nicolas Brax

Le 27/11/2013

Titre :

Self-Adaptive Multi-Agent Systems for Aided Decision-Making:
An Application to Maritime Surveillance

ED MITT : Domaine STIC : Intelligence Artificielle

Unité de recherche :

Institut de Recherche en Informatique de Toulouse - UMR 5505

Directeur(s) de Thèse :

Marie-Pierre GLEIZES
Eric ANDONOFF

Rapporteurs :

Pierre CHEVAILLIER,
Giovanna Di MARZO SERUGENDO & Paolo GIORGINI

Autre(s) membre(s) du jury :

Henric JOHNSON
Michel MOREL

THESIS

presented at

Université Paul Sabatier - Toulouse III

U.F.R. MATHÉMATIQUES, INFORMATIQUE ET GESTION

to obtain the title of

DOCTEUR DE L'UNIVERSITÉ DE TOULOUSE

delivered by

UNIVERSITÉ PAUL SABATIER - TOULOUSE III

Mention INFORMATIQUE

by

NICOLAS BRAX

Doctoral school: Informatique et Télécommunication
Laboratory: Institut de Recherche en Informatique de Toulouse
Team: Systèmes Multi-Agents Coopératifs

Self-Adaptive Multi-Agent Systems for Aided Decision-Making: An Application to Maritime Surveillance

---

JURY

Pierre CHEVAILLIER	*Associate Professor, Ecole Nationale d'Ingénieurs de Brest*	(Reviewer)
Giovanna DI MARZO SERUGENDO	*Professor, University of Geneva*	(Reviewer)
Paolo GIORGINI	*Associate Professor, University of Trento*	(Reviewer)
Henric JOHNSON	*Professor, Blekinge Tekniska hogskola*	(Examiner)
Michel MOREL	*DCNS*	(Guest)
Marie-Pierre GLEIZES	*Professor, Université de Toulouse III*	(Supervisor)
Éric ANDONOFF	*Associate Professor, Université de Toulouse Capitole*	(Co-Supervisor)

To the two of you,
without whom this work could not have been done.
May you know it one day.

Many Thanks...

NEARLY four years since I begun this work. Quite a long time but I have reach an achievement. Before proceeding to the next stage, I have to thank many people for their help, their support, their advice and all the little things that contributed to this thesis.

To begin with, I would like to thank the members of my jury. First, Pr. Giovanna Di Marzo Serugendo for accepting to be the president of my jury, but also for your evaluation of my work. This evaluation and the review of my thesis has also been done by A.Pr. Pierre Chevaillier and A.Pr. Paolo Giorgini. Thank you for the remarks, the questions that helped to clarify my works and also for the hints for future works.

I would also thank Pr. Henric Johnson for your participation in my jury and your comments on my works that day. Also, with Pr. Davidsson, I would thank you for the three months I spent at the BTH, in Karlskrona. That was a great time, both academic and personal.

Finally, I would like to thank one last member of my jury in the person of Michel Morel. Thank you for accepting the invitation and your observations on my work. But also, thank you for giving a chance to the multi-agent systems and integrate them into the I2C project. And of course, thank you to all of you in DCNS for the practical point of view you brought in my works.

If a thesis reach an end with the defence, it could not have been achieved without the people around me all this time. First, every person in the IRIT lab, always here and helpful when needed. It is also up to you if I was able to work into a pleasant environment.

Of course, I should give notable thanks to the SMAC team and its members. First, Marie-Pierre, Pierre and Eric. You have supervised my works all these years and you were present when I needed. From each of you, I was able to learn and to apprehend various and different aspects of the academic life, of research and teaching. And also, some special thanks to Pierre who made me learn the benefits of disagreement.

And then, still within SMAC, I would thank Frédéric A., your supervising of my Master works was great and encourage me to pursue. Also, thank you to Noélie, Christine and Frédéric for the coffee breaks and the "asocial" refuge. And I can not name all of you, professors, teachers and fellow PhD students, but I have to thank you as well for the warm feeling within the team.

Also, while I was not part of your research group, I would like to thank all of you in the BTH, I did and do not forget. Marie, Martin, Niklas, Stefan, Bengt, Ewa, Edgar, Anton and

Rawand. You were all welcoming and make me feel good while I was there.

And what could have been done without the people around me, all of my friends and family. Jérémy, Maroussia, Arnaud, Raphael and Rémi for the ever funny and much needed role-playing games. Likewise, Cédric, Fabrice, Germain, Greg, Niko, Terry and Tiff for the drinks but also for the LoL games we had, they were saving during the writing process. The "Ours", I should not forget you, even if we don't see each other much. And finally, Pierre, thank you for who you are, for what you are.

My aunts and uncles, Christine and Bernard, Christine and Paco, Maryse and Michel, Monique and my godparent Michel, but also you my cousins, Natacha, Sandrine, Benoit, Rémi, Robert-André, Pierre-Jean, thank you all for your support and the marks of attention, even the smallest one. It helped a lot. This is even more true for you, my parents, Philippe and Françoise, Anne et Bertrand. You were always behind me, supporting me and you made me go forward. Really, thank you for everything you have done.

Finally, last but not least, Laura, Jade, Ethan. You just came up when I began this work and you are still here. I can not say how you changed my life but one sure thing, I could not imagine it without you now. You supported me, my doubts and my temper which is not always good. Your presence was my beacon all these years, it still is. This thesis is also yours for I could not have finished without you. You are an entire part of my life.

Contents

Introduction Générale

The introduction in English starts page 5

Une introduction...

La prise de décision est un *processus complexe qui amène à sélectionner une suite d'actions parmi plusieurs scénarios différents* duquel résultera un choix final, ce qui peut être une action ou une opinion sur une action. A l'origine, c'est un mécanisme biologique que toute entité sentiente peut accomplir pour choisir la meilleure alternative en fonction de ses besoins et de ses objectifs. Au début, il s'agissait de trouver un abri et de la nourriture pour pouvoir survivre, ensuite, avec l'évolution de la société, ces besoins n'ont plus lieu d'être et la prise de décision concerne maintenant de nombreux autres domaines, de plus en plus difficile et complexe à représenter. Ainsi, plusieurs contextes sont concernés : quelles sont les actions qu'un agent de change doit suivre pour maximiser ses bénéfices ? Que doit faire un étudiant pour obtenir son doctorat ? Comment le chirurgien peut-il pratiquer une opération sans risquer la vie de son patient ? Et de nombreuses autres questions peuvent se poser, par tout le monde et à chaque instant.

La société actuelle fournit une quantité astronomique d'informations disponible en un clin d'œil et cela les rend difficiles à traiter en vue de prendre la bonne décision. Si ce n'est pas grave quand cela concerne l'organisation d'un week-end, cela peut devenir vital quand il s'agit de faire atterrir un avion. Pour cette raison, de plus en plus de logiciels de prise de décision sont mis en place pour aider l'utilisateur à traiter les informations selon un certain objectif et sont utilisés dans plusieurs domaines, allant du médical au boursier. Malgré tout, il est quelque peu illusoire de proposer un unique logiciel qui soit capable de prendre en compte les différences selon les cas d'application. C'est une des raisons pour laquelle nous avons choisis, lors de ces travaux, de nous intéresser à une sous-partie de la prise de décision. En effet, outre le choix d'une série d'actions pour accomplir un objectif, cela peut aussi être utilisé pour traiter des informations et mettre en évidence un point particulier qui sera utile à un utilisateur qui prendra alors la décision finale. De tels systèmes d'aide à la prise de décision sont largement utilisés dans les systèmes de surveillance.

La surveillance, c'est *l'observation des comportements, des activités et d'autres informations,*

concernant généralement des personnes, dans le but de les influencer, les diriger ou les protéger. Si on met à part les problèmes de vie privée et l'aspect politique de la surveillance, un tel système regroupe différents outils et propose des moyens variés pour réaliser la surveillance d'une zone particulière, ou de quelque chose. Encore une fois, les systèmes de surveillance sont nombreux et utilisent différents outils : des réseaux de caméra pour la surveillance routière, des radars et des senseurs pour le contrôle aérien, des capteurs pour examiner le corps humain,... Tous ces systèmes ont un point commun, leur fonctionnement. Ils traitent l'information pour isoler les parties considérées comme critiques pour prendre une décision. La plupart du temps, un ou plusieurs opérateurs sont en charge du système et utilisent ensuite ces morceaux d'information pour décider de la marche à suivre.

Cependant, de nos jours, de plus en plus de systèmes sont conçus pour alléger le travail des utilisateur. De la détection d'une intrusion sur un réseau à la détection de ralentissement dans le trafic en passant par la surveillance fluviale, les systèmes sont capables de gérer ces tÃ¢ches automatiquement car ce sont des objectifs relativement simples à remplir. Mais quand il s'agit de systèmes cruciaux, les résultats de l'ordinateur sont vérifiés par un utilisateur car ils peuvent être sujets à des erreurs et la différence entre un missile et un avion inconnu peut être de la plus haute importance.

C'est pourquoi le processus de prise de décision dans les systèmes de surveillance reste le plus simple possibles et que des experts, les utilisateurs, sont en bout de chaîne. Certains travaux proposent d'améliorer la prise de décision en ajoutant des mécanismes d'apprentissage, un apprentissage qui est le plus souvent fait a priori et en se basant sur des données de tests représentant des situations connues. Le problèmes de ces données, c'est qu'elles sont spécifiques au domaine d'application, au système et au type d'apprentissage.

Ainsi, plus qu'une prise de décision, ces systèmes sont conçus pour détecter des comportements anormaux et les soumettre aux opérateurs. Mais encore une fois, il n'y a pas d'approche générique du problème car le domaine et le contexte jouent un rôle fondamental.

Contributions

En conséquence, le travail de cette thèse concerne la conception d'un système capable de traiter les information et d'identifier des anomalies dans un domaine qualifié de générique. De plus, pour être efficace, le système doit être capable d'apprendre des ses erreurs. L'objectif final n'est pas de remplacer les opérateurs mais de proposer un système suffisamment efficace pour les aider dans leur travail. Pour prendre en compte l'apprentissage dans un système d'aide à la prise de décision, nous proposons un modèle générique à base d'agents en accord avec la théorie des Adaptive Multi-Agent Systems (AMAS). Le modèle proposé est conu pour être facilement adapté à différents types de systèmes, du moment que la représentation de l'environnement est compatible. De plus, le système est tel que l'utilisateur n'a pas besoin d'être un expert en AMAS pour l'utiliser.

Les agents qui composent notre systèmes ont une connaissance locale, ainsi qu'une représentation locale, du problème et de leur environnement, qu'ils peuvent percevoir, ainsi que leurs voisins, d'autres agents. Ils sont ensuite capable de coopérer et de s'adapter à leur environnement pour satisfaire à leur propres objectifs. Chaque agent est capable de calculer une valeur de satisfaction et l'atteinte d'un équilibre pour chaque agent constitue une solution au problème. La différence principale avec les autres approche est qu'il n'y a pas besoin de retranscrire le problème selon un certain formalisme ou selon un certain type de données. En effet les agents sont aussi proche de la définition du problème que possible et une définition générique nous assure des agents génériques.

Finalement, le système est conçu pour préserver la coopération et l'adaptation identifiées dans la théorie des AMAS. Cela peut se résumer comme suit :

- ▷ un traitement local en temps réel des événements et des changement dans l'environnement ;
- ▷ une robustesse et un passage à l'échelle améliorés ;
- ▷ la possibilité de fonctionner dans un environnement dynamique ;
- ▷ un système ouvert où des agents peuvent apparaître et disparaître pendant le fonctionnement.

Pour évaluer le système, nous avons pratiqués des expérimentations en utilisant une plateforme de simulation capable de représenter et de simuler différents composants d'un système de surveillance dans le but de remplir deux rôles principaux :

- ▷ simuler les comportements de plusieurs entités en créant et en suivant différents scénarios ;
- ▷ simuler les opérateurs ayant une réelle connaissance sur les comportements et les entités.

Ainsi, il est possible de combiner cette plateforme avec un système de détection de comportements anormaux pour pouvoir tester ce dernier.

Organisation du manuscrit

Les chapitres suivant composent cette thèse :

Chapitre 1. Dans ce chapitre, le **contexte spécifique** à cette étude est présenté : les **systèmes de surveillance** sont globalement décrits, et plus spécifiquement en ce qui concerne la surveillance maritime. Dans une seconde partie, nous présentons **l'aspect multi-agent et la méthodologie** que nous avons suivie. Nous y détaillons la notion d'agent et de système multi-agent avant de présenter la théorie des AMAS.

Chapitre 2. Ce chapitre concerne **l'état de l'art** concernant les méthodes d'apprentissage automatique et les systèmes de surveillance. Tout d'abord, les principales techniques d'apprentissage sont détaillées car elles sont utilisées dans les processus de détection d'anomalies et de prise de décision. Ensuite, nous présentons différents types de systèmes de surveillance qui ont été développés et appliqués dans de nombreux domaines.

Chapitre 3. Ce chapitre décrit **le modèle agent générique** que nous avons conçu en accord avec la problématique autour de la détection d'anomalies : la levée d'alerte et l'apprentissage pour améliorer le système. Nous décrivons les agents, leurs comportements et leurs interactions, toujours en accord avec la théorie des AMAS.

Chapitre 4. Ce chapitre présente **une application du système multi-agent** décrit précédemment. Nous y présentons les modifications et les ajouts pratiqués sur le système pour pouvoir l'utiliser dans le domaine de la surveillance maritime et plus spécifiquement dans le cadre du projet I2C[1]. De plus, nous y décrivons un outil développé pour simuler divers composants d'un système de surveillance.

Chapitre 5. Ce dernier chapitre est une discussion autour **l'évaluation du système multi-agent adaptatif** dans son contexte d'application. Différents critères sont utilisés pour souligné l'efficacité du système dans le domaine de la surveillance.

[1] *Integrated System for Interoperable sensors & Information sources for Common abnormal vessel behaviour detection & Collaborative identification of threat* – EU Project

General Introduction

Introduction to...

Decision-making is *a complex process leading to the selection of a course of action among several alternative scenarios*. This results in a final choice, which can be an action or an opinion on action. Originally, it is a biological process that every sentient entity can perform in order to choose a set of actions to follow according to a need or an objective. At the beginning, the first processes were about the finding of a shelter and food in order to survive. With the evolution of society, these needs are fulfilled and the decision making concerns a lot of other domains, becoming more and more difficult to express and represent.

But more than that, it is also used in several contexts: what is the course of actions to follow for a trader to improve its benefits? What have a student do in order to achieve a Doctorate? How can a surgeon perform an operation without endangering its patient life? And many others. All of this shows that decision-making is used everyday, more or less consciously, by everyone.

With the current society and the huge amount of information available at any time, it can be hard to process all of it in order to make the good decision. If it is not of a prime importance when it comes to the organisation of the week-end, it can be vital when it comes to land a plane. For this reason, more and more decision-making software are set up and help the users to process all the information according to a given objective. This is applied in several domains, from stock exchanges to medical diagnosis.

It is quite illusive to tackle all of these domains with one stone. This is the main reason why, in this work, we focus on a particular part of decision-making. Indeed, if decision-making can be used to select actions according to objectives, it can also be used to process information and highlight a particular information leading to the decision by the user. Such systems helping the decision process are widely used in surveillance systems.

The surveillance is *the monitoring of behaviours, activities or other changing information, usually of people for the purpose of influencing, managing, directing or protecting*. Set apart the privacy problems and the politic aspects of surveillance, a surveillance system is a system that regroups several tools and means to achieve the surveillance of a particular area or thing.

Again, surveillance systems are numerous and use various tools: camera networks to monitor traffic, radars and sensors to monitor planes, sensors for the monitoring of the human body,... The common point of these systems is their running. They process information and provide a piece of information that is considered critical in order to take a decision. Most of the time, one or more operators are in charge of such systems and use the information given to choose the course of actions by themselves.

However, nowadays, more and more systems are designed in order to alleviate the users of these concerns, especially simple ones. Detecting network intrusion, monitoring water overflow, spotting traffic jamming, these are simple tasks that can be handled by an automatic system. But for more crucial systems, the computer result is verified by a user. The system can be subject to error and the difference between a missile and an unknown plane can be of the utmost importance.

This is why the decision-making process in surveillance systems is kept as simple as possible and the users are at the end of the process. Some works propose to improve the decision-making by the addition of learning in the system. This learning is mainly made a priori, using training data set representing known situations. But these data set are very unique according to the domain of application and the kind of systems and learning used.

Thus, more than decision-making, these system are designed to detect abnormal behaviours, also called anomalies in this thesis, and propose them to the operators. But once again, there is no generic approach about anomaly detection in surveillance systems since it is so dependant on the domain and the context.

Contributions

Therefore, the work of this thesis is the design of a system able to process information and identify anomalies in a generic domain. Besides, in order to be efficient, this system must be able to learn from its errors. The final objective is not to replace the operator but to propose a system efficient enough to help the operator in its charge.

To address the learning in a aided decision-making system, we propose a generic agent model based on the Adaptive Multi-Agent System (AMAS) theory. The proposed model is designed to be easily adapted to various kinds of decision-making system, as long as the environment representation is equivalent. Besides, the multi-agent system is built so the user has not to be an AMAS expert to use it.

The final multi-agent system is divided into two multi-agent systems composed of several types of agents. These agents have a local knowledge and a local representation of the problem. They also possess a local environment, and means to perceive it as well as their neighbours, meaning the other agents. The agents are then able to cooperate and adapt themselves to their environment in order to satisfy their own objectives. Each agent is able to compute a satisfaction value and by reaching a satisfaction equilibrium among the

different agents a global solution of the problem is obtained.

The main difference with the other approaches is that there is no need to translate the problem into a specific framework or input data. Indeed, the agents are built as close as possible as to the definition of the problem. Taking the problem from a generic point of view ensures that the resulting multi-agent system is generic too.

Finally, the system is designed so that it preserves the cooperation and the adaptation features identified in the AMAS theory. This can be sum up by:

- A local and real-time treatment of the events and changes in the environment at runtime.
- An increased level of robustness and the scalability of the number of agents.
- An ability to run in a dynamic environment.
- An openness of the system, where new agents can appear and disappear during the running.

To evaluate the system, we have conducted experiments using a simulation platform. This platform is able to represent and simulate various components of a surveillance system and it aims at fulfilling two roles:

- Simulate the behaviours of several entities by creating and reading several scenarios.
- Simulate the operators with a real knowledge on the behaviours of the entities.

Thus, it is possible to plug this platform with a abnormal behaviour detection system in order to test it.

Manuscript Organisation

This thesis is divided into the following chapters:

Chapter 1. In this chapter, the **specific context** of this study is presented: surveillance systems are globally described and a focus then is done on maritime surveillance. In a second part, we present the **multi-agent aspect** and methodology of this work. Here, we detail the notion of agent and multi-agent system before we present the *Adaptive Multi-Agent System Theory*.

Chapter 2. This chapter concerns **the state of the art** of existing machine learning methods and surveillance systems. First, the main machine learning techniques are detailed since they are used in decision-making and anomaly detection processes. Second, we present several kinds of surveillance systems that have been designed and applied to numerous domains.

Chapter 3. This chapter describes **the generic agent model** we have designed according to the anomaly detection problem: the alert triggering and the learning to improve the system. We describe the agents, their behaviours and their interactions, accordingly to the Adaptive Multi-Agent System Theory.

Chapter 4. This chapter presents **an application** of the multi-agent system described in the previous chapter. We present the modification and the addition to the system in order to be used in a maritime surveillance system, called I2C[2]. We also describe a tool developed in order to simulate different component of a maritime surveillance system.

Chapter 5. This last chapter discusses **the evaluation** of the adaptive multi-agent system in its application context. Different criteria are used to underline the efficiency of the system within the surveillance domain.

[2] *Integrated System for Interoperable sensors & Information sources for Common abnormal vessel behaviour detection & Collaborative identification of threat* – EU Project

1 Context, Theory and Tools of the Thesis

« You cannot impose ideologies on people who do not embrace it wholeheartedly. »

Peter F. Hamilton

Contents

The chapter in English starts page 11.

Résumé général du chapitre

c*Ainsi qu'exposé en introduction, ce travail se place dans le contexte des systèmes de surveillance, et plus spécifiquement la détection d'anomalies, soit de comportements anormaux, dans une zone donnée. Ces comportements sont capturés par divers capteurs et senseurs et sont noyés au milieu de milliers de comportements dits normaux. De plus, ils peuvent être de différentes natures en fonction des entités que l'on souhaite surveiller. En revanche, le point commun de nombreux systèmes de surveillance est qu'ils sont conçus pour aider à la détection et l'identification des anomalies et des menaces conséquentes.*

A un instant donné, une situation peut être isolé et définie comme la séquence d'état qui y a mené. Elle peut ensuite être analysée et qualifiée en tant que suspecte ou non. Bien que cette décision puisse être automatisée, elle est plus souvent le résultat d'une intervention et d'une réflexion humaine. En effet, de nombreux facteurs entrent en jeu et en fonction de son contexte, une situation peut être normale alors qu'elle était qualifiée d'anormale dans un contexte différent.

Dans le domaine particulier de la surveillance maritime, les problèmes concernant l'automatisation sont exacerbés tant le nombre de variables à prendre en compte à un instant donné est important. Les opérateurs sont considérés comme des experts dans leur domaine et l'évaluation des différentes situations est le fruit d'une longue expérience. Lors de nos travaux, nous nous sommes attaqués à deux aspects : la représentation des entités surveillés, des navires, et de leurs comportements dans un premier temps, puis la possibilité pour le système d'apprendre de ses erreurs dans un second temps, de manière à proposer des résultats de plus en plus pertinents et corrects.

Pour y parvenir, nous avons choisi d'utiliser la technologie et les méthodes multi-agents. D'inspiration biologique, ces systèmes sont particulièrement efficaces quand il s'agit de représenter des modèles complexes et dynamique comme c'est le cas en surveillance. Un agent est une entité logicielle autonome capable d'interagir avec son environnement et les autres agents, qui possède des objectifs, des attributs et des capacités d'actions, de réflexion et de perception (ceci est une définition générique, les agents pouvant être adaptés et plus ou moins contraints en fonction du modèle). Lorsque ces agents sont plongés dans un environnement commun, on parle alors d'un système multi-agent (SMA).

Les interactions entre les agents en fonction de leurs objectifs, commun ou global, et de leurs définitions autorise la résolution de différents problèmes : optimisation, résolution, modélisation,... L'une des particularités les plus intéressantes des SMA concerne l'auto-organisation. Des systèmes capables de s'auto-organiser exhibe des structures ou des capacités sans intervention extérieure, c'est à dire que ce sont les agents eux-mêmes qui "créent" ces possibilités. Cette notion permet d'alléger la conception des agents et de laisser le SMA trouver la meilleure solution pour un problème donné.

Cet aspect des SMA est au coeur de la théorie des AMAS utilisée pour les travaux présentés

dans ce manuscrit. Cette théorie prône une définition locale des agents pour aboutir à l'émergence de la fonction globale. Ainsi, les agents d'un AMAS sont définis de manière à ne posséder qu'une perception limitée de leur environnement et à ne pouvoir interagir qu'avec leurs voisins proches en fonction d'un objectif local et propre à eux-mêmes. La définition des règles d'interaction permet ensuite de structurer les agents entre eux, et donc le système, pour aboutir à une fonction globale qui n'est pas intrinsèquement défini dans le SMA. La définition des règles locales est donc le point central de la théorie des AMAS. L'élégance de la chose vient d'une simple observation : plutôt que d'indiquer aux agents ce qu'ils doivent faire, il faut leur indiquer ce qu'ils ne peuvent pas faire et comment y remédier ou le prévenir. Ainsi, les agents sont libres de tester les différentes possibilités qui s'offrent à eux tout en tenant compte des interdictions définies par le concepteur. Ces interdictions doivent évidemment être en accord avec l'objectif voulu pour le système final.

Comme le système I2C concerne un environnement complexe, dynamique, ouvert et non totalement connu, nous avons choisi d'adopter cette philosophie. En effet, les agents que nous avons définis sont simples mais capable de représenter les différents éléments d'un tel système tout en intégrant les connaissances du domaine : les entités et les interdictions qu'elles ont (généralement issues de la législation ou de comportements anormaux connus et identifiés). Il faut enfin noter que notre travail s'appuie sur la téhorie des AMAS mais n'en adopte pas l'intégralité des principes et méthodes.

1.1 Surveillance Systems

1.1.1 Generality about Surveillance Systems

The context of this work is the surveillance domain and, more precisely, the anomaly detection in surveillance systems. Such systems use different kinds of sensors in order to capture the behaviours of entities in a monitored area. The entities and the area can be of abstract nature (for example the surveillance of agents in a computer networks) or of concrete one (for example the surveillance of pedestrians and vehicles). In a same way, various methods and techniques are used, see chapter 2 for more information on this.

The common point of the majority of surveillance systems is that they are designed to help the detection of anomalies or abnormal behaviours, like an intrusion in a computer network or a collision between several cars. The generic running of such a system is as follows.

Various sensors are set up to capture all the data and information on the entities in the monitored area. At a given time, the set of data allows to identify several situations: two vehicles stopped on the highway, a pedestrian lying down on the road, a computer node not responding, two computers with the same unique identification,...

These situations are then analysed in order to detect the true anomalies from noise and false or explainable situations. This analyse can be automated but is actually widely made by surveillance operators who are also considered as expert in their domain. This expertise

comes from experience, but also from the understanding of several underlying concepts (human behaviours, specificities of a network,...). These concepts are difficult to implement in a fully automated system, that is why the human presence is almost always needed in surveillance systems.

Finally, when a situation has been analysed and an anomaly is detected, it is broadcast to the concerned authorities in order to answer the anomaly (eradication of the false computer identification and its cause, sending assistance on highway,...). More than for the situation analysis, this phase is hard to automate and human operators and experts are mostly left in charge of these action decisions.

1.1.2 About Maritime Surveillance

Among the surveillance systems, a growing domain concerns the maritime activity. Indeed, maritime area are home to many activities, legal or illegal, and the huge size of the areas to monitor is a main challenge. Besides, a lot of parameters are involved in the maritime activities: weather and ship behaviours are the most common. Therefore, several projects take place into the maritime surveillance.

To be efficient, a maritime surveillance system must be able to permanently track and monitor all types of maritime activities in order to detect abnormal behaviours, to analyse them and to early identify threatening situations. However, currently there does not exist such a system that can handle all the necessary information needed to detect all abnormal behaviours. More precisely, this kind of system must provide [Ince et al., 1999]:

- A permanent and all weather coverage of border maritime areas.
- A continuous collection and fusion of heterogeneous data provided by various types of sensors deployed on coastlines and offshore, but also information from external sources (for example open data [Kazemi et al., 2013]).
- An automatic detection of abnormal ship behaviours and the triggering of alerts if these abnormal behaviours represent potential threats.
- The understanding of these threats in order to allow decisional authorities to deal with them.
- An adapted interface intended to human operators involved in maritime surveillance.

Therefore, using passive or active means, the maritime surveillance systems aim at seeking and identifying various activities: illegal fishing, drug dealing, ship in distress,... In such system, the operation is similar to the one in general surveillance systems:

1. Sensors capture the tracks of the ships in the monitored area.
2. The tracks are associated to all relevant data available.

3. The behaviours of the ships are then analysed by operators in order to identify situation worth an intervention

4. The situation information is transmitted to the concerned authorities for action: rescue, military response,...

Such systems are difficult to design since they have to deal with a great number of ships at the same time and only a little part of them exhibits abnormal behaviours. For this reason, the automation of abnormal behaviour detection is a main challenge in maritime surveillance.

To sum up, just like the maritime surveillance systems, nowadays surveillance systems are deployed in **wide areas** and monitor **great numbers of entities**. Besides, the **behaviours of the entities are complex and difficult to represent**. Moreover, the monitored system are often **open and dynamics**: indeed, the entities are moving in and out of the system and they are interacting with each others. Ultimately, to be efficient for the operators, the system has to **be reactive** and raise relevant alerts in a limited amount of time. The current work proposes to address these issues using an Adaptive Multi-Agent System.

1.2 Multi-Agent Systems

In this thesis, we advocate the use of a multi-agent system in order to design an alert triggering system. Multi-agent systems are coming from the Artificial Intelligent domain and are now presented.

A multi-agent system is a system which can be artificial or not and which is composed of several autonomous entities, the agents. Such systems are coming from the Distributed Artificial Intelligence, which have developed to tackle the growing complexity of the systems by the utilisation of distributed computation techniques. Three fields of study then appear: the first concerns the distributed solving of problems using several communicative entities; the second concerns the parallel artificial intelligence tackling the performance constraints and issues by the utilisation of parallel computation and related algorithms; third, the multi-agent approach proposes to solve problems using the coordination between intelligent autonomous entities called agents. In these three fields, the processing are distributed and the control is decentralized. Usual systems from the AI are fragmented and the agents are the basis of these new systems.

One consequence of this phenomenon is the disappearance of the direct resolution of one complex problem for the benefit of splitting of the problem into several sub-problems being solved by the agents. From the beginning and until now, the MAS have improved greatly both in their conception and in the understanding of their dynamics. In the nineties, the MAS took advantages of disciplines like Biology and Sociology. Indeed, they integrates characteristics like the self-organisation, the emergence and agent interactions[Di Marzo Serugendo et al., 2011].

More precisely, the works of the team SMAC[1]), from the IRIT[2], within which this thesis takes place, focus on the solving of complex problems thanks to the utilisation of MAS and the notion of emergence. The use of MAS allows the design of new solutions for complex problems. Finally, the decomposition of a problem in several smaller problems easier to solve has been of a great benefits for the MAS when the global solution is hard to reach with common algorithms.

Hereafter, we detail the notion of agents and the interaction between agents leading to multi-agent systems. Then, the Adaptive Multi-Agent Systems (AMAS) Theory is presented since it is the basis of our work in this thesis.

1.2.1 The Notion of Agent

Agents are the basis of a Multi-Agent System. From the computer science point of view, the agent concept is an evolution from the concept of object and the Agent-Oriented Programming gradually becomes a new programming paradigm. This new approach takes a great advantage of the strong connections of this field with social sciences and biological phenomenons.

Several definitions of an agent exist, all focusing on different aspect of the agent. One of the most commonly spread is the definition given by Ferber [Ferber, 1995]. *An Agent is a virtual or physical entity:*

- ▷ *Able to act in an environment;*
- ▷ *Able to communicate with other agents;*
- ▷ *Possessing objectives to reach or satisfaction functions to optimise;*
- ▷ *Possessing its own resources;*
- ▷ *Able to perceive its local environment;*
- ▷ *Eventually able to represent its environment;*
- ▷ *Possessing skills and offering services;*
- ▷ *Eventually able to reproduce;*
- ▷ *Whose behaviour aims to satisfy its objectives according to its resources, its skills, its perception and its representation of its environment and neighbourhood.*

Woolridge [Wooldridge and Jennings, 1995] offers a more accurate way to distinguish an agent. It is a *hardware or software-based computer system that enjoys the following properties:*

[1]Systèmes Multi-Agent Coopératifs – Cooperative Multi-Agent Systems
[2]Institut de Recherche en Informatique de Toulouse – Toulouse Research Institute of Computer Science

- *autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*

- *social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;*

- *reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*

- *pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.*

This definition integrates the autonomy and the social dimension between several agents. This sociability also includes the user who can communicate with the agents using, for example, messages. These two definitions allow to define an agent following four axes.

First, the autonomy as a main characteristic for an agent as it is a main difference from the object. The agent is able to decide by itself and is not a passive entity waiting for a message or a stimulus.

Second, the local aspect of an agent participates in its definition. An agent acts in regard to its own skills as well as it local perceptions and knowledge. This also indicates one major advantage of the multi-agent systems: the agents have no need to know the global problem to solve while they can act and perform local solving.

The third axe is the social characteristic of an agent. Indeed, an agent is able to communicate with other agents, not necessarily of the same nature.

Fourth, the agent takes place into an environment. It can perceive its environment and act on it. Perception and action allow the agent to adapt itself or its local environment.

Several definitions of an agent lead to several types of agents and their precise classification is hard to determine. From abstract genericness to application techniques, several possibilities arise and it is common to categorise the agents according to their cognition level. Agents can be separated into cognitive agents and reactive agents. A cognitive agent possesses a knowledge database used to reason on its environment and its neighbourhood. A cognitive agent is also able to plan its actions in order to reach its own objective. A reactive agent only react to what it perceives and in regard to a predefined behaviour. The reactive agents are interesting when used in mass, their simplicity allowing a huge number of parallel processing.

In fine, an agent is an entity that can have various roles and can thus be used in several different systems. Communication between the agent is the premise of multi-agent systems.

1.2.2 The Multi-Agent Systems

However, communication and social aspect if not sufficient to define a MAS. The creation of a MAS is not a simple creation of a collection of agents but it needs a precise description of their interactions. In order to create a MAS, several elements have to be taken into account [Demazeau, 1995]: the agents and their behavioural models, the environment where they agents are set (spatially or not), the interactions and the tools to manage them between the agents, the structural organisation of the agents from hierarchy to simple relationships.

More than a group of agents, even with the definitions above, the definition of a multi-agent system also requires the definition of mechanisms for its organisation, allowing it to self-organise[Drogoul, 1993] and self-adapt[Dalpiaz et al., 2011].

Again, several MAS organisation are available. The first one is called AGRE[Ferber et al., 2005], for Agent, Group, Role, Environment. This organisation is based on the agent –an active communicative entity having one or several roles within a group–, the group –a set of agents with common characteristics and restraining the communication of the agents within the group–, the role –an abstract representation of the agent function within its group– and the environment –the abstract or concrete location of the group and its neighbourhood. This organisation is widely spread because of its simplicity and its generic aspect. It is a natural extension of the definition of a MAS and does not add abstract description of the relationships between the agents.

Another example is the self-organisation, coming from the biological field. Self-organised systems are systems showing the ability to exhibit forms or structures without external intervention. This phenomenon is also observable in crowds and is the focus of several sociological works. Actually, self-organisation can be found in percolation, liquid crystals, star and galaxy formation among many other examples. From a computer science point of view, the self-organisation is the ability for a system to modify its organisation while it runs and without external stimulus. This self-organisation is made by simple reactive agents and it lead to the global behaviour of the system. Inspired by several natural systems, there are several methods to reach self-organisation: swarm intelligence from the flocks of birds or shoals, stigmergy from ant colonies,...

The self-organisation is a way to delegate the organisational constraints to the agents themselves and alleviate the design of the multi-agent system. However, this impact the design of the agents which can be hard to obtain if the MAS is complex to organise. Thus, the designer can focus on the local behaviours of the agents and the global organisation of the MAS will emerge from all these behaviours.

1.2.3 The Adaptive Multi-Agent System Theory

The Adaptive Multi-Agent System Theory [George et al., 2003; Glize et al., 1998] proposes to develop agents focusing on the well-discussed advantages of cooperation. This

theory is based on the theorem of functional adequacy. This theorem is demonstrated in [Glize et al., 1998] and states as follows:

Theorem: For any functionality adequate system, there exists at least one cooperative internal medium system that fulfils an equivalent function in the same environment.

The cooperation is the ability of the agents to work together towards a common global objective. It implies complementarity, dependencies and solidarity between the agents and their activities. When designing the agents of such systems, we have to be sure that they:

- ▷ try to overcome the difficulties, called *Non Cooperative Situations* (NCS). A NCS is a failure that the agent is able to detect and repair. These failures are either related to the agents or to their interactions with their environment.
- ▷ chose the actions that have the least chance to disturb the other agents as well as themselves. This involves an anticipation of the NCS.
- ▷ are cooperative towards the system and other agents. This means that agents try to help the agents encountering a NCS.

The functional adequacy theorem means that we only have to understand a subset of a system in order to obtain a functionally adequate system in a given environment. We concentrate on systems with the following properties.

- ▷ the system is cooperative and adequate to its environment, its parts have no knowledge of the global function of the system.
- ▷ the system has no defined goal and acts according to its perception seen as feedbacks used in order to adapt the function towards the global goal. The adaptation is performed by the agents skills and their own representations of the environment.
- ▷ each part of the system only focuses on its point of view.

The main idea is that it is easier to focus on the agents and their local interactions when it comes to designing a complex system. Several applications proved that the local cooperation is relevant to tackle the problems without having an explicit knowledge of the objective and how to reach it [Capera et al., 2004; George et al., 2003].

Figure 1.1 shows how the local agents are able to perform a global objective by adapting and organizing themselves using cooperation. Each part P_i of a system S is able to achieve a partial local function f_{P_i} of the global function f_S. This function is the result of the combination of the partial functions f_{P_i}. The combination is noted by the operator $\bigcirc$. The combination being determined by the current organisation of the pars, we can deduce

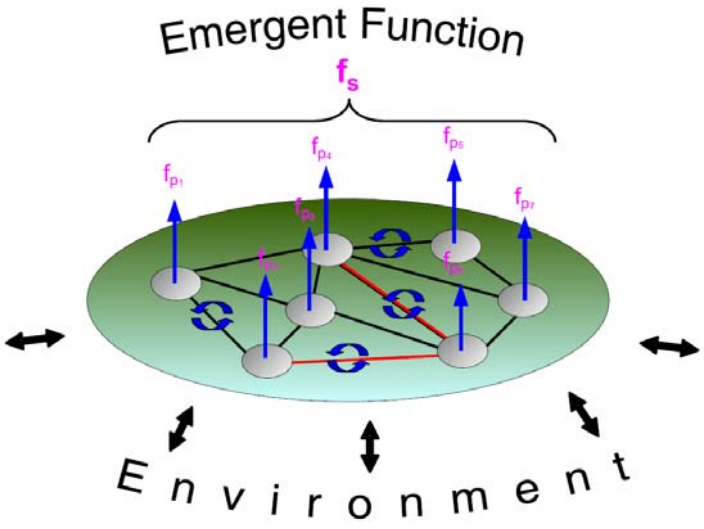

Figure 1.1: Adaptation and emergence of the function of the system.

$f_S = f_{P_1} \bigcirc f_{P_2} \bigcirc \cdots \bigcirc f_{P_n} \bigcirc$. As $f_{P_1} \bigcirc f_{P_2} \neq f_{P_2} \bigcirc f_{P_1}$, the changes in the organisation leads to a change in the combination of the partial functions and thus in the global function. This is very efficient to adapt a system to its environment and Adaptive Multi-Agent System are relevant to build such systems. Finally, AMAS are systems composed of several autonomous agents set into a common environment and trying to solve a common task using local interactions and knowledge.

The cooperative agents used in these systems are autonomous –they have the ability to decide to take an action or not–, unaware of the global function of the system –this function emerges from the local function of the agents–, can detect Non Cooperative Situations and act to fix them and they are benevolent –they want to achieve their goals and are cooperative at the same time.

1.2.3.1 On Multi-Agents Systems

Ultimately, multi-agent systems have been used to solve several distinct problems. Nonetheless, we can identify several common characteristics:

- **Modular applications** are well suited for MAS use due to the concept proximity of agents and objects.
- The **decentralisation** embeds the notion of **autonomy** linked to the decision making abilities of the agents.
- A highly dynamic problem can be easily tackled using a MAS approach, due to the two previous points: it is more **robust and easier to act locally** when subject to frequent and quick modifications.

- ▷ **Open systems**, where all the data is not necessarily available, can be represented by agents since the lack of information does not prevent the agent action.
- ▷ Due to the variety of behaviours and possible interaction, the MAS are **well suited for complex systems**. Besides, the variety of possible organisations ensures the possibility to represent several different systems.

Given the definition of a multi-agent systems and the related notions of self-organisation using local interactions and behaviours, the use of a MAS seems well suited in the domain of surveillance systems. Indeed, these systems are complex to represent due to the number of entities to monitor and their possible interactions, they are also open since new entities can appear at any time, or disappear. However, other techniques from mathematical fields or artificial intelligence domain are applied to tackle the surveillance system problems. The next chapter aims to presents these solutions in order to determine their advantages and drawbacks regarding the surveillance as considered in this thesis.

2 State of the Art

« The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...' »

Isaac Asimov

Contents

The chapter in English starts page 24.

Résumé général du chapitre

Des sentinelles du moyen-âge, et même avant, aux drones de surveillance actuels, les systèmes de surveillance prennent maintenant une place à part entière dans nos vies. Avec les avancées en informatique et en électronique, ces systèmes sont de plus en plus automatisés et différentes techniques ont été mises au point pour gérer les différents aspects de la surveillance. Au début, cela concernait la mise en réseau et l'enregistrement automatique des diverses observations possibles. Actuellement, les principaux travaux tournent autour de l'analyse des information et la reconnaissance d'intentions : suivre une personne dans une foule, analyser un comportement pour identifier une menace,...

Dans nos travaux, nous nous sommes concentrés sur la ***prise de décision****. En effet, dans la plupart des systèmes, la la décision finale revient aux utilisateurs considérés comme des experts dans le domaine. Ils sont ainsi capable d'identifier une menace en fonction des observations faites par le système et de prendre les mesures nécessaires. Pour automatiser une telle identification, l'un des moyens est de rendre le système capable d'apprendre les situations à risques, voire d'apprendre les différents évènements qui peuvent aboutir à ce genre de situations. Ainsi, en plus d'une levée d'alerte quand une situation est jugée anormale par le système, nous avons pris en compte la nécessité d'un apprentissage pour qu'il soit capable de se corriger en fonction des retours des utilisateurs.*

En matière d'apprentissage automatique, il existe plusieurs famille de techniques comme ***l'apprentissage supervisé****,* ***non supervisé*** *ou* ***par renforcement****. Dans le cas d'un apprentissage supervisé, le système s'appuie sur une* ***base de connaissances*** *construite par un expert du domaine considéré pour associer des données fournies en entrée à une des classes définie dans la base de connaissance. Si le système ne trouve aucune classe, il associe ces nouvelles données à une classe qualifiée d'inconnue. Cette technique est très utilisée en reconnaissance de formes.*

Dans le cadre d'un apprentissage non supervisé, le système ***découvre, à partir de données fournies en entrée, les classes par lui-même****, c'est-à-dire sans l'aide d'un expert. Le système analyse les données qu'il reçoit en se basant sur des groupes d'attributs ou de fonctions qu'il est capable d'observer. L'utilisateur de la base de connaissance doit ensuite analyser les résultats pour déterminer la nature de chaque classe alors identifiée. L'apprentissage non supervisé est souvent utilisé lorsqu'il s'agit de traiter de vastes corpus de données qu'il faut évaluer et classer, comme par exemple les données relatives à l'évaluation du trafic sur des réseaux informatiques ou les données relatives à la classification des protéines.*

Enfin, nous distinguons l'apprentissage par renforcement dans lequel ***les effets d'une action sur l'environnement sont pris en compte****. La base de connaissances est capable de se réorganiser en fonction de ces actions et de modifier les classifications établies. L'algorithme d'apprentissage qui sert de support à cette réorganisation va utiliser les retours en provenance de l'environnement pour améliorer la classification de la base de connaissances. Bien souvent les systèmes à base d'agents*

sont utilisés pour mettre en œuvre ce type d'apprentissage. En effet, les agents peuvent prendre des décisions en fonction de leurs objectifs, communs ou individuels, et observer l'impact direct sur leur environnement. En fonction des bénéfices ou des inconvénients qu'ils en retirent, les agents peuvent choisir de modifier leur classification ou non.

Parmi les méthodes d'apprentissage par renforcement, nous pouvons retenir les algorithmes de ***Q-Learning et de TD-Learning*** *ou encore les réseaux de neurones grâce aux* ***mécanismes de rétro-propagation de l'erreur*** *et d'élagage. L'inconvénient de ces méthodes est principalement le coût en temps de calcul. De plus, de tels systèmes peuvent être complexes à implémenter en raison des mécanismes de décision parfois mal connus.*

L'apprentissage par renforcement est le type d'apprentissage que nous utilisons dans le système I2C. En effet, les alertes déclenchées par le système multi-agent MAS4AT sont soumises à un opérateur impliqué dans la surveillance maritime. Le rôle de cet opérateur est de valider ou invalider ces alertes. Les alertes validées sont documentées et transmises aux autorités qui vont décider d'éventuelles interventions sur le terrain tandis que les alertes invalidées ou les alertes non relevées sont transmises à MAS4AT pour qu'il apprenne les valeurs quantifiant l'importance des anomalies impliquées dans ces alertes.

Dans le domaine de la surveillance, quelque soit le contexte (surveillance maritime, médicale, sociologique ou dans le domaine de la sécurité des réseaux, les systèmes peuvent être classés en deux catégories. Nous distinguons tout d'abord les systèmes de surveillance qui ***modélisent les comportements autorisés*** *à l'aide par exemple de réseaux de neurones, et qui indiquent comme anormal tout comportement qui n'est pas décrit dans le modèle. Ces systèmes considèrent les* ***comportements inconnus comme des comportements anormaux****, ce qui n'est pas nécessairement vrai dans le contexte de la surveillance maritime. La deuxième catégorie de système* ***modélise plutôt les comportements interdits*** *en se basant sur la législation du domaine dans lequel ils s'appliquent (par exemple la législation maritime). Cette législation est décrite sous forme de règles construites par des experts du domaine et qui traduisent les infractions relatives au domaine. Les systèmes adoptant cette philosophie identifient donc les comportements anormaux en fonction des événements observés dans les zones surveillées. Ces systèmes sont pertinents pour la détection d'anomalies correspondant à des comportements inattendus. Mais ils sont inadéquats et doivent être améliorés dans le contexte de la surveillance maritime pour trois raisons principales. Premièrement, la plupart du temps, le comportement anormal d'un navire est le résultat de la violation de plusieurs règles et chaque infraction considérée indépendamment n'est pas nécessairement une anomalie en soi et ne provoque pas le déclenchement d'alertes. Malheureusement, les systèmes basés sur des règles ne permettent pas cette distinction. Deuxièmement, la législation maritime est très complexe et les alertes peuvent de plus être dépendante du type du navire. Tout cela rend la réglementation maritime difficile à modéliser sous forme de règles. Troisièmement, en se basant uniquement sur une approche à base de règles, il n'est pas possible de découvrir les nouveaux comportements des criminels, qui sont de plus en plus imaginatifs quand il s'agit de contourner la loi. Un système de surveillance maritime doit donc être capable d'identifier de nouveaux comportements anormaux, et donc de découvrir les nouvelles stratégies mises en œuvre par les criminels.*

Concernant les systèmes de surveillance et de détection des comportements, il faut également citer les systèmes ***basés sur les techniques de reconnaissance de chroniques ou d'intentions****. La reconnaissance de chroniques est utilisée pour détecter des événements ponctuels qui sont liés par des contraintes temporelles se produisant dans un contexte donné.. Ces événements et les contraintes associées sont généralement représentés sous la forme de graphes qui constituent une base de connaissance pour le système. Bien que pertinente, cette technique est difficilement utilisable dans notre contexte car les événements se produisant dans le domaine de la surveillance maritime ne sont pas nécessairement connus à l'avance. Nous avons à faire à un système fortement dynamique dans lequel des événements peuvent apparaître ou disparaître et le système de surveillance doit pouvoir s'adapter et découvrir par lui-même ces changements. La reconnaissance d'intention porte elle sur l'analyse des actions d'un objet et des changements qu'il provoque dans l'environnement en vue de déterminer son objectif. Cette technique se base sur la formalisation de bibliothèques pour établir des comparaisons avec les traces observées. Cette formalisation exhaustive peut s'avérer difficile dans le domaine de la surveillance maritime en raison de la complexité de la législation associée.*

L'inconvénient majeurs de tels systèmes est le ***manque de généricité****. En effet, la plupart sont conçus pour un domaine et un contexte particulier et sont difficilement applicable dans d'autre cas. De plus,* ***l'apprentissage est coûteux à mettre en œuvre****, que ce soit en terme de temps ou de ressources matérielles. Nous proposons ainsi d'adresser ce problème en présentant un système multi-agent capable de lever* ***des alertes pertinentes dans un système de surveillance*** *sans prendre en compte le contexte, mais aussi* ***capable d'apprendre en temps réel*** *de ses erreurs.*

2.1 Introduction

In this thesis we propose self-adaptive multi-agent systems to help decision-making in a surveillance system. We also instantiate these systems in the context of maritime surveillance. Watchmen in the beginning, camera and drones nowadays, surveillance systems are part of our existence since the beginning, whether to prevent threats or for control. With the advent of computer science, such systems become more and more automated and several techniques were designed in order to automate the different parts of the surveillance. At the beginning, this was the video recording and the creation of recording units networks. Currently, there are several works on the information analysis and intention recognition [McLachlan, 2004; Ryoo and Aggarwal, 2009]: following a person in a crowd, analysing a behaviour in order to identify a threat, identify a crowd movement and identifying its origin, the reason for surveillance systems are numerous and various.

The current work focuses on the decision-making. In most of these systems, the final decision is the responsibility of the user of the system. Given the set of behaviours, the threat is identified and given by the surveillance operator [Auslander et al., 2011], or given the symptoms, several diseases are proposed to the doctor that can validate the more plausible [Kononenko, 2001]. Nowadays, several works exist on the automatic decision-making in surveillance systems, in order to alleviate the operator works and identify more

complex situations.

In order to work efficiently, these systems are able to learn the situations and how they arise. This learning can be from a training set of data or in real-time, but it improves the overall efficiency of a surveillance system [Valera and Velastin, 2005]. Indeed, when a system is wrong on a situation, it can learn and correct itself in order to avoid the error the next time it identifies the situation. The learning can also be used to introduce a flexibility in these systems which are then able to identify situation even with slight differences from what they have learnt.

This chapter is organised as follows. Section 2.2 concerns the state of the art in machine learning. We present the most common techniques used in machine learning, divided in three categories: the supervised learning 2.2.1.1, the unsupervised learning 2.2.1.2 and the reinforcement learning 2.2.1.3. Then we present multi-agent based learning 2.2.2, derived from the reinforcement learning and we bring out the most common methods. Our system is set up in a maritime surveillance system. As a consequence, we had to identify the global characteristics of these systems. Section 2.3 focuses on surveillance systems. More precisely, we propose an overview of such systems before we concentrate on decision-making in the surveillance domain 2.3.2. Then, we propose a specific survey about the maritime surveillance systems 2.3.3. Finally the section 2.4 discusses the points previously mentioned and provide an analysis of the detailed solutions.

2.2 Machine Learning

Machine learning is a field of study from the artificial intelligence and it represents the discipline concerning the development and the implementation of automatic methods that allow a system to evolve using learning processes. Since we were able to model computer systems, learning was needed to improve the autonomy of these systems from the users. The systems using machine learning are then able to complete tasks that they are not able to tackle using more common algorithms.

They are several machine learning approaches. In the beginning, the systems were based on expert knowledge in order to classify data according to the sets of example they were given, or use expert knowledge to identify the classified output of the system. Quickly, systems are designed according to natural observation and the genetic algorithms and neural networks were designed and are the precursors of reinforcement learning techniques. Reinforcement learning is now widely used and concerns the learning using the feedbacks, whereas they come from the environment of the systems or from internal functions of the system itself.

In the next section, we present different techniques of learning and we focus on the following three main families of algorithms: the supervised learning, the unsupervised learning and the reinforcement learning. We end with a section dedicated to the multi-agent

based learning.

2.2.1 Machine Learning Overview

As presented in the introduction, there are several techniques of machine learning from the supervised learning [Venturini, 1994] to reinforcement learning [Kaelbling et al., 1996] via unsupervised learning [Hinton and Sejnowski, 1999]. We now briefly present these techniques, their advantages and drawbacks.

2.2.1.1 Supervised Learning

The techniques of supervised learning are mainly used when the system has to classify and sort data, whatever they represent. When the different classes are known and can be modelled, the system learns to sort them according to a classifier. In order to accomplish this, the system compares its input to data labelled by an expert of the concerned domain. The input data are then associated to these examples and are given the same label. This kind of learning has several applications in form recognition [Pao, 1989; Jurafsky and James, 2000] or healthcare domains [Kononenko, 2001]. Indeed, it is possible to identify a shape or a disease by comparing it to examples and propose a diagnosis.

When it comes to solving a problem using supervised learning, there are several steps to follow:

- The first step is to identify the examples on which the system will base its classification. The users or the experts have to choose an accurate enough set of data for the system to be efficient. For instance, if the system has to identify diseases, it can base itself on examples of the malady in a whole or it can take into account each symptom of the disease one by one.

- When the set of data used as examples by the system is identified, the second thing to do is to gather training data. These data have to represent the real world according to what we want to observe. The training data is a set of pairs with the observable input, for example the symptoms, and the related outputs, for example the disease related to the symptoms.

- The next step is common to the majority of learning algorithms: the definition of the function to learn. This function is a representation of the inputs of the algorithm that is usable by the system. It is usually a vector containing the characteristics of the object. The difficulty is to model these characteristics such as there is enough of them to be relevant, but not too much in order to avoid dimension issues. Finally, depending on the system used, these structures have to be translated in a form understandable by the system: decision tree, equalities system,...

- ▷ Finally, in order to adjust the parameters of a supervised learning algorithm, it is possible to run it on the training set defined previously. The user is now able to validate the outputs of the system. Once the adjustment is done, the accuracy of the learning must be measured on a data set different from the one used for the training. It is possible to go back on one of these steps in order to improve these measures.

It exists a lot of supervised learning algorithms and for numerous domains: shape recognition [Turk and Pentland, 1991], bioinformatics [Tan and Gilbert, 2003], information classification [Manevitz and Yousef, 2000], pattern recognition [Duda et al., 2012],... However when it comes to supervised learning, there are issues to consider.

First, a given learning algorithm can fit the data almost exactly and give relevant outputs only if the inputs are highly similar to the training set, but such a system is not able to efficiently process data that does not fit exactly. On the contrary, a learning algorithm can process all kind of data but then give inaccurate results. The design of such an algorithm has to take into account the trade-off between the accuracy –in the provided results– and the flexibility –about the input representations– of the system [Geman et al., 1992]. Currently used algorithms used automatic adjustment of this bias according to the expected results.

Second, as said before, the function to learn has a major impact on the learning algorithm. Indeed, if the function is simple, it is easy to learn it with a small amount of data. but if the function is complex, a huge amount of data can be used before it is learnt. The function is the translation of the real-world and thus represents the different interactions between components in different parts of the context. This complexity can be taken into account by the algorithms, but it is better to design the function as simple as possible.

The third issue concerns the dimension of the input space. Indeed if the characteristic to take into account by the algorithm are numerous, the learning of the function can be difficult, especially if it depends on a small number of these. On one hand, if the overage characteristics are numerous, they can confuse the learning algorithm and cause it to propose different results for a same input. On the other hand, if the input data in a whole is useful, it will take time to process it and to propose a result. Practically, it improves the accuracy of the learnt function to remove the irrelevant features which it represents, this can be done manually or automatically [Aha, 1998]. This problem is a sub-part of the search space reduction [Srinivas and Patnaik, 1991], where one seeks to map data to a lower dimensional space. This is not treated in this thesis.

The fourth problem concerns the output space. If there are several errors on the outputs, proven it is due to error on the inputs (human, sensor,...), the learning algorithm will not be able to find a function matching exactly with the training data. This can lead to an over-fitting of the system because too much precision is wanted related to the lack of precision of the input data. Over-fitting is also possible if the function to learn is too complex according to the learning data, introducing noise in the output of the algorithm. To tackle this problem, the learning algorithm can be made more flexible to the detriment of the accuracy.

Besides these four main issues to take into account, there are other factors that can intervene in the application of a learning algorithm. There are methods that require numerical input data (e.g. neural networks, support vector machines, linear regression) while some algorithms are better used to process data of different kinds at the same time (e.g. discrete and continuous values). Also, if the input data are redundant, that can cause numerical instabilities disrupting some learning algorithms requiring a regularisation of the data. More generally, depending on the input data, the complexity of the function and the wanted accuracy of the results, some algorithms perform better than others. The choice of an algorithm is also dependant on the resources and the time allocated for the algorithm to process the data [Mitchell, 1997].

However, such algorithms are well suited when they come to identify situations already known. One good example is the handwriting recognition where the number of letters is finished and the usual writing is well known [Plamondon and Srihari, 2000]. Another example is an automatic diagnosis system [Saito and Nakano, 1988] which analyses the symptoms of a patient and proposes a corresponding disease and the medication to counter it. In both of these examples, the training data are well known and defined and the input are likely to fit one of the class. Indeed, it is not everyday that we discover a new disease, let apart a new letter.

To sum up, a supervised learning algorithm is based on a knowledge base built by an expert of the considered domain. The algorithm is designed to associate input data to a class and the function representing it. It can be more or less flexible according to the wanted results. This is mainly used when a pool of situations have to be evaluated according to another pool of known situations. One of the main drawbacks of such an algorithm is its initialisation requiring an accurate evaluation of the domain and the problem in order to design well the input data and the learning function. Besides, these systems are not easily adaptable to change except by the use of statistical methods [Cohn et al., 1996]. Either way, these algorithms are time and resources consuming.

Supervised learning techniques are well known and widely used in several systems. However, they need a **training phase that is costly** for it to be efficient. Besides, they require an important **global knowledge** and are **not efficient when applied to an open environment**. Therefore, when applied to surveillance systems, the supervised learning techniques have difficulty to handle on the fly the dynamics and complexity of the system.

2.2.1.2 Unsupervised Learning

Compared to the supervised learning, the unsupervised learning is more a clustering of unlabelled input data in order to find hidden structures inside it [Hinton and Sejnowski, 1999].

Starting from a pool of input data with no label or indication, an unsupervised learning system is able to identify the relations and the interactions between the different sets of data.

Thus it is able to regroup then in a same cluster and build an organised knowledge database. This is widely used in data mining or whenever it is needed to process a huge amount of data in order to evaluate and classify them. Common examples are the classification of computer networks data [Zander et al., 2005] or proteins classification [Andrade et al., 1993].

The main advantage of such algorithms is the discovery of interactions and hidden structures among a pool of data. This can help for further or more accurate processing of it. Neural networks are widely used when unsupervised learning is needed but the drawbacks are the same as for the supervised learning: the time and resource costs can be very high [Mitchell, 1997]. Besides, if the system is able to identify the clusters of data, it is not able to label them, this phase requiring another analysis or the intervention of an expert.

When supervised learning needs an accurately designed function and input data, the unsupervised learning can classify data and propose the relevant attributes or information to take into account to analyse the given data. For example, in handwriting recognition, an unsupervised learning system should be able to identify the words of a same language when supervised learning is able to give them a sense.

Unsupervised learning is widely used in order to discover the hidden structures in unlabelled data and to sort data according to unknown criteria. After a **training phase**, unsupervised learning systems are able to sort data into cluster within which the each piece of data has similar properties to the others. However, such systems have **high resource costs** and are **not efficient when facing novelty**. When applied, to surveillance systems, the unsupervised learning techniques compare the behaviours to analyse to a knowledge database and try to relate them to a known cluster. When no association can be done, the behaviours is set as unknown. This poses the issue of unknown behaviours which are not necessarily abnormal. Besides, these techniques are **depending on the function used to compare** the data. Its definition is a major challenges when defining such systems since a small variation can give totally different results. As one of the main issues in surveillance systems is **the detection of abnormal behaviours from normal ones,** such methods are hard to use in this domain.

2.2.1.3 Reinforcement Learning

Inspired by natural behaviour, the reinforcement learning is "how an agent ought to take actions in an environment so as to maximize some notion of cumulative reward". The main difference with the supervised learning is that there is no pair made between input data and output data.

When it comes to reinforcement learning, the learning system must be able to perceive and evaluate the changes that are provoked by its learning in its environment. This evaluation is highly dependant on the objectives of the system. To put it simply, if the changes allow the system to draw near its objectives, the actions that lead to it are rewarded;

on the contrary, if the system rolls away from its objective, the actions receive a penalty. In the end, we have a learning way towards the objectives. The most well-known and main algorithms of reinforcement learning are the TD-Learning [Watkins and Dayan, 1992] and the Q-Learning [Sutton, 1988] as well as the neural networks with the addition of error back-propagation and pruning mechanisms [Coulom, 2002]. We now present these techniques.

Q-Learning [Watkins, 1989]: this is one of the most known algorithm in reinforcement learning. The aim of Q-Learning is to find the optimal sequence of actions in order to reach an expected goal. When the problem can be modelled using a finite Markov Decision Process, the Q-Learning algorithm is able to find optimal actions for the agent running it. Q-Learning requisites at least an agent, a set of states S and a set of actions A for each state. Then the algorithm is able to compute a numerical value Q translating the quality of the combination state / action. Each time the agent selects an action, the value of Q is updated so to take into account the selected action, the previous and the new states and a reward. The reward is dependant on the objective of the agent and the distance between the current state and this objective. If the agent draws near its objective, the reward is a bonus; if the agent recedes from its objective, the reward is a penalty. This is an incremental algorithm that improves at each step its evaluation of the pairs state/action available to the agent. When the pairs are learnt, the agent is able to take the best decision, according to the problem and its current state, by taking the most rewarding one.

TD-Learning [Sutton, 1988]: the temporal difference learning algorithm is another mainly used technique in machine reinforcement learning. This is a prediction method that uses Monte Carlo [Metropolis and Ulam, 1949] ideas because the learning is based on samples and not the whole environment, but it also uses dynamic programming [Nemhauser, 1966] ideas because the current estimations of the algorithm are based on the previous ones. The TD-Learning takes into account the correlations between the current prediction and the previous ones. The operation of this algorithm is quite simple: (if it is run by an agent) the agent makes a prediction of its state and waits for its observation; when possible, the agent observes the real state and tunes its prediction accordingly; the agent repeats the operation as many time as needed until the observation and the predictions are the same. The algorithm 1 allows a simple implementation of the TD Learning.

The aim of this algorithm is the evaluation of a given policy of actions for the agent. There is no improvement of the said policy. V(s) represent the value given by the agent to each state it can have, Π is the policy to evaluate. When the agent has achieve this learning, it is able to give the reward value it has associated to each state it can reach and it is therefore able to evaluate the policy of action it follows.

An example of domain where TD-learning can be applied in meteorology. When you predict the weather for the next five days, each passing day can be used to refine the

Algorithm 2.1: A simple algorithm for TD Learning.

```
V(s) ← random initialisation
Π ← initialisation
foreach episode do
    s ← initialisation
    foreach time step in current episode do
        agent performs a given by Π
        observe reward r and next state s'
        V(s) ← V(s) + ∝ [r + γV(s') − V(s)]
        s ← s'
        End when s terminal
    end
end
```

prediction for the fifth day.

Neural networks [Fausett, 1994b]: Artificial neural networks are mathematical models inspired by biological neural networks and consists in a group of interconnected artificial neurons able to process an information. It is often an adaptive system able to change its structure allowing it to perform a learning. More on artificial neural networks can be found in [Mehrotra et al., 1997]. In this section, we focus on the interest in neural networks from the learning point of view. This is close to the Q-Learning algorithm: given an objective and a class of functions F allowing to reach it, the neural network provides the sequences of functions $f \in F$. The choice of the function is done using the same way as the Q-Learning. This means that a neuron is able to compute a cost function translating the quality of the chosen function according to the current state of the neuron, the past actions and the distance to the objective. In machine learning, the definition of this cost function is a central point of study [Baresel et al., 2002]. Mechanics of reinforcement learning are easy to add to neural networks by providing feedbacks to the neurons on the functions they have chosen and their costs. The feedbacks are provided either by the system itself or by the users of the system. Then the aim of each neuron is to maximize its own reward. Multi-agent systems are particularly well suited for the modelling of artificial neural networks as each agent can take the role of a neuron. Besides, if the agent is given a set of abilities allowing it to perceive the results of its action and compute its own reward. This has been successfully used in [Jolly et al., 2007] and [Xu et al., 1996].

To sum up these three approaches, the learning process can be represented using an agent which can move towards different states according to a set of actions. For each state, a reward or penalty is granted and the objective of the agent is to maximize its total reward. To say it otherwise, the knowledge base is able to reorganize itself and modify the set classifications. The learning algorithms indeed uses the feedbacks from its environment

(the user, the context,...) in order to improve the classification [Sutton and Barto, 1998].

Such methods can be time consuming and complex to implement because of the decision mechanisms that can be poorly known [Kaelbling et al., 1996]. Besides, not all the search space might be explored because if an agent reaches its objective, it will not search any more. This can lead to the discovery of local minima but not the real solution [Gori and Tesi, 1992]. Agent-based systems often use this kind of learning [Lauer and Riedmiller, 2000] as the agents are able to perceive their environment, to decide what is the most well suited action to take and to perform this chosen action. This cycle is made according to the objectives, both common and individual, of the agents. Finally, the agents can observe the feedbacks of their actions on their environment and are able to note the actions, thus modifying themselves or not to perform better.

When applied to surveillance systems, reinforcement learning is used by agents within the system in order to correct themselves. Therefore, they are able to become **more and more efficient when they trigger alerts**. This kind of system seems useful and is able to run in real time, using feedbacks as it operates to correct itself. However, one of the main issues in reinforcement learning concerns **the representation of the system to learn and its definition**. The more the environment is complex, the more it is difficult to find an efficient method because **it is hard to model all the interactions and synergies within it**. As surveillance systems are used in open and dynamics environment, a reinforcement based learning technique needs **an efficient representation of the system, yet a simple one**.

2.2.2 MAS-Based Learning

According to Panait and Luke [Panait and Luke, 2005], there are two major categories of multi-agent learning approaches. On one hand, there is one learner for a pool of agents. This is close to the traditional learning techniques but there is some issues when the number of agents increases and we lack the usefulness of the number of agents. This is what is called the team learning. On the other hand, there are several learners in concurrence. The objective is the reduction of the search space by using each agent as a learner. This is called the concurrent learning.

2.2.2.1 Team Learning

As said before, in team learning, there is only one agent involved in the process of learning. It is an easy approach to agent learning because the agent can use common machine learning techniques. However this is still challenging because there is few considerations about the interactions between the agents and the collective behaviour can be surprising. This is the emergent complexity of the multi-agent systems [Weiss, 1999]. On the advantages, there are no issues brought by the interactions between several learners and the unique learner has to improve the performance of the entire team and not of each agent individually. Actually, team learning only focuses on one agent and the multi-agent aspect

of the learning can be left aside.

There are also drawbacks to this team learning. The first one is common: the side of the search space. As a single agent has to search for each other agents, size of the search space grows in an exponential way. For example, if each agent has 50 possible actions available and if there is one hundred agents, the size of the whole search space is 50^{100}. The number of actions to explore can quickly be overwhelming [Calvez and Hutzler, 2005]. Another drawback is the resource cost. Indeed, as there is a single learner, all resources have to be gathered on a unique spot of the system, the agent that is learning. We can imagine that the use of a multi-agent system translates the need of a distributed system and this kind of learning somehow ignores this need.

When it comes to team learning, there are two main categories, the homogeneous and the heterogeneous team learning. In homogeneous team learning, what is learnt is then applied on each agent of the team. On the contrary, in heterogeneous team learning, the learner agent can propose a unique behaviour for each of its team-mate. This last kind of learning offers best solutions but more complexity in their design. Eventually, it is possible to combine these two approaches by dividing the pool of agents in several teams [Panait and Luke, 2005].

The choice of what learning will be used depends on the need of specialists. Indeed, the heterogeneous team learning proposes a unique behaviour for each agent of the team, so they can be considered as specialists among their team-mates. In a few words, if a unique agent can perform well, homogeneous team learning is well suited [Balch, 1998a] but when some specializations are needed, the heterogeneous team learning is preferable [Potter et al., 2001].

2.2.2.2 Concurrent Learning

The concurrent team learning makes a better use of the multiple agents part of a multi-agent system. Indeed, several learning processes can work in the team in order to improve it as a whole. The most common way to embody concurrent learning is to give a learning ability to each agent, meaning that each agent is able to modify its own behaviour according to what it perceives and what it chooses to do. According to Jansen and Wiegand [Jansen and Wiegand, 2003], concurrent learning is the best to use when a decomposition of the problem is possible and moreover if it is possible to solve each sub-problem one by one. Indeed, in concurrent learning, each learner has a smaller search space to consider than the conjoint space. Thus, if each agent can solve its part of the problem in a way that it helps the whole system to reach its objective, the search space is reduced to the individual one and so is the complexity. Besides, when each agent has the ability to learn its own behaviour in a smaller search space, it reduces the resources needed for the learning. More precisely, these resources can be distributed equally among all the agents for a better repartition and use.

The main issue when designing a concurrent multi-agent learning system is to take into

account the other agents. Indeed, when an agent modifies its behaviour, it is according to what it perceives of its environment, of its context. But the other agents are adapting themselves as well. If the behaviour of an agent modifies the learning context, this can be problematic for the other agents. To put it in another words, the learnt behaviour of an agent A can modify the environment in a way that the learnt behaviour of an agent B becomes useless [Sandholm and Crites, 1996a]. Thus, such system need mechanisms to handle it, either by taking into account the other agents or by the modification of the agent roles. This way, the agents are always able to perform an action useful for the system.

We expose four approaches to tackle these issues: first the credit assignment learning that focuses on the reward system; second the focus on the dynamics of learning and the impact of the adaptation of the other agents; then we present some works on the modelling of the other agents involved in the learning; finally we focus on the Adaptive Multi-Agent Theory that models the local interactions of the agents involved in the process in order to avoid conflicts.

Credit assignment focuses on how the reward is perceived by the learners. The easiest way is to split it equally among the learners. This is called the global reward, when one learner receives a reward, all the learners receive the same reward. This poses the problem that some agents may receive a reward that they do not deserve or, more generally, the agents do not receive rewards proportionally to their actions in the learning process [Wolpert and Tumer, 2001]. On the contrary, the local reward focuses on rewarding only the agents that perform well in the learning without consideration for the other agents. This can provoke the apparition of greedy behaviours among the agents. Balch [Balch, 1997] experiments suggest that a local reward increases the homogeneity of the learners while a global reward is best suited when the learners' specializations are needed. Between these two extremes, some works propose to reward the agents according to their true participation in the learning process. This is useful when scaling to a very large number of agents [Wolpert and Tumer, 2001]. Tangamchit et al. [Tangamchit et al., 2002] propose a solution where the reward is not applied to the agents whenever an action is done but instead the agents receive it according to a sequence of behaviours. Credit assignment is simple to put into action, but the decision to reward an agent besides another, and how, can have a real impact on the learning process. Moreover, some choices may lead to some sort of team learning where only one agent does all the work. The objective while designing credit assignment in concurrent learning can be sum up by "some social scientist words: how to reach the individual interest of none but the collective interest of all" [Lichbach, 1996].

Dynamics of learning aims to take into account the environment changes provoked by the learning of the other agents. These dynamics are difficult to observe and the evolutionary game theory tools are the most common to analyse the dynamics of concurrent learners. Using these perspectives, the aim of the agents should be to reach

the Nash equilibrium, meaning that each agent considers its reward optimal according to the other agents (if it tries to change and modify its reward, this can only decrease its own reward) [Gmytrasiewicz, 1992].

Team-mate modelling models the other agents to understand their behaviours and takes them into account during the learning process. For example, based on a Bayesian learning method [Boutilier, 1996], the agents are able to learn the behaviours of their neighbours and they will try to cooperate with them. Another approach [Suryadi and Gmytrasiewicz, 1999] leads to model a set of behaviours the agents are likely to follow and choose which one is the best suited in the current situation. The finality is the same and each agent tries to cooperate with the others during the learning phase. Works have been conducted in order to avoid infinite loops (an agent models the behaviour of another agent that models the behaviour of the first agent) [Mundhe and Sen, 2000] or on how the communication can improve the team-mate modelling [Ohko et al., 1997].

The Adaptive Multi-Agent Theory aims to model the local interactions between the agents in order to avoid situations that would prevent them to reach their objectives. Then, the global solution is emerging from the local processes [Capera et al., 2003]. Using this solution, each agent is given a criticality which represents its own constraints according to its knowledge. The higher is this criticality, the more the agent tries to change itself in order to lower the constraint level on itself. But the agents also take into account the other agents. Indeed, each agent is given a set of Non Cooperative Situations which are situations designed according to the domain and the context of the system [Bonjean et al., 2009] and known to prevent the global objective. For example, if two robots have to ship boxes from one spot to another, one non cooperative situation would be the two robots taking the same box. The agents in an adaptive multi-agent system are then taking actions in order to decrease their criticality and, at the same time, they identify and avoid the non cooperative situations that would decrease the criticality of their neighbours.

2.2.2.3 Example of MAS-Based Learning

The reinforcement learning techniques seems well suited for MAS-based learning. Indeed, the agents are able to evaluate an action according to the result of this action and their objectives. They may also take into account the consequences on the environment, on their neighbourhood,... In the literature, several works are using Q-Learning and TD-Learning based algorithms to perform learning in a multi-agent system.

But the agents can be a tool of the learning more than learner by themselves. For example, some works propose ant-based agent to perform clustering of data. A generic algorithm for it would be similar to the algorithm 2.

One main feature of ant-based agent algorithms is the spatial representation. Therefore, each agent using this algorithm is initialised with the knowledge of several locations. These

Algorithm 2.2: Clustering algorithm for ant-like agents.

```
initialisation of locations
while Exist data to sort do
    seize data
    compare with locations
    move data to most relevant place
end
```

locations are places where certain types of data belong. Then, the agents are able to retrieve pieces of data and analyse it in order to determine where they would fit the best. In the end, the agents have moved all pieces of data to their most relevant location. Thus, the representation consists of several stacks of data grouped by similarity. This similarity is computed using various methods, like distance computation.

This algorithm is a simple one used to illustrate this kind of works. In more advanced works, the agents are able to determine the cluster locations by themselves, they are able to adapt their fitness functions –to compare data–, they can cooperate,... In such systems, the agents do not integrate learning techniques but their behaviours, skills and characteristics are designed in a way that the whole multi-agent system is able to perform learning tasks.

2.2.2.4 MAS-Based Learning

MAS-Based Learning is interesting to address problem involving **complexity and dynamics**. Moreover, the decentralised aspect of such techniques ensures a relatively **low cost in term of resources** while they are **quick enough to perform a learning**. However, current techniques are based on **the global representation** of the system, requiring a **good knowledge to model it**. When a surveillance system aims at monitoring behaviours, the main issue is precisely the lack of this knowledge: unknown behaviours, abstract definition of alerts,... Therefore, a MAS-based learning methods should be **able to model the system whatever it is** and even if all the information on it is not known.

2.3 Surveillance Systems

According to the common definition, surveillance is the monitoring of the behaviours and information of entities. The aim of the surveillance can be management, protection or other justifications. There are a lot of surveillance means, from the camera recording to the geographical location devices via biometrics surveillance and data mining techniques [Valera and Velastin, 2005]. In this thesis, we address the surveillance systems along two axes. First, we are interested in the surveillance systems aiming at the analysis of the observed behaviours in order to identify threats and anomalies. Indeed, a lot of systems make available the "pictures" of what they observe but not all are proposing an analysis and

let the users identify the threats. Second, we focus on computer-aided surveillance systems. By these terms, we mean systems that are able to perform the said analysis and learn to identify new threats, illicit and abnormal behaviours or even anomalies. A lot of systems aiming at the identification of anomalies provide tools but are still relying on the users to identify the final anomaly. However, some systems are able to identify the anomalies by themselves.

Following these two axes, we are able to narrow down the interest of this thesis when it comes to surveillance systems. Indeed, the application field is the system I2C, a maritime surveillance systems aiming at the identification of threats and abnormal behaviours by the analysis of data from a wide range of sensors deployed over the monitored area. This analysis is made using computer algorithms and the users are able to make a validation, or invalidation, of the detected anomalies, in order for the system to learn.

This section is thus divided in three parts. First, we present a general overview of surveillance systems, focusing on anomaly detection systems. Second we present different techniques of identifications of dangerous events in surveillance systems, and how they are learnt. Finally, we present several examples of surveillance systems in the domain of maritime surveillance.

2.3.1 Surveillance Systems Overview

First of all, when speaking of surveillance systems, a lot of works in the literature deal with the medical domain [Thompson et al., 2006]. In this particular domain, we have to distinguish systems designed to analyse data and to prevent epidemics and systems that are able to analyse symptoms and to identify a disease. On one hand, the systems aiming at the prevention of an epidemic rely on the highlight of several proven cases of a given disease. Then, using statistical methods about the people (circulation, relations,...) and according to the knowledge on the said disease (propagation rate, lifespan,...), these systems are able to propose estimations on the spreading of the disease and the impact on the population [Miller, 1994]. On the other hand, the systems focusing on one particular patient are fed with its symptoms. Then they are able to compare these symptoms with their knowledge database in order to identify the most plausible disease according to the input. These systems are the machine counterpart of the differential diagnosis [Thompson et al., 2006]. This method is used by doctors to identify a disease: they use their knowledge to eliminate diseases one by one until they find the one that involves all the symptoms that the patient presents.

However in this thesis we focus on systems able to identify anomalies in the behaviours of entities. Although the techniques are similar, there is a difference with the systems exposed above aiming at the identification and the systems we want to focus on aiming at the prevention.

In Valera and Velastin [Valera and Velastin, 2005], the authors identify four processes

when it comes to surveillance: the detection, the tracking, the behaviour analysis and the retrieval.

2.3.1.1 Detection and Tracking

It is common to speak of third generation surveillance systems when dealing with systems that have to cope with a large number of sensors and resource deployed over a given area. The aim of such systems is to provide the users a good understanding of what they observe and eventually focus their attention on specific zones of interest. One of the main investigations in this field concerns the data fusion of the information coming from several sources [Hall and Llinas, 1997]. The other focus is the identification of anomalies and it is debated in section 2.3.2.

Most of the techniques we present here are already applied to crowd surveillance but the involved principles can easily be extended to generic entities surveillance. The majority of current surveillance systems use visual detection based on sensors such as camera or satellites. The devices are often associated to a tracking module able to detect, recognize and track an object through the camera objectives or through the satellites or radar representations.

This have been used in traffic surveillance [Lou et al., 2003; Remagnino et al., 1997] where the operators can use the system in order to track a vehicle in a parking lot, but it has also been applied to highway traffic, in maritime harbour [Pozzobon et al., 1999] or railway surveillance [Ronetti and Dambra, 2000]. There are also some projects in maritime surveillance where the aim is to provide an accurate representation of the monitored area using several sources.

The common point of these systems is the distribution of the detection devices and the fusion of the data in order to track an object from one end to the other end in the monitored area. Several works on data fusion and integration can be found in [Battistello et al., 2011; Ristic et al., 2008]. The authors of [Remagnino et al., 1997] highlight the need of an interpretation of the behaviour of the detected objects in order to build an automated surveillance systems. Without it, human interventions would still be needed. This is one of the reasons why the current systems come with an analysis module [Ivanov et al., 1999; Remagnino et al., 1997].

2.3.1.2 Behaviour analysis

In the surveillance context, whatever is the domain (maritime, crowd control, healthcare [Li et al., 2011], sociological [Gupta and Hossain, 2011] or even in computer network security [Shi et al., 2011]) we can categorize the behaviour analysis in two distinct parts.

On one hand, there are systems that model the authorized behaviours, using neural

networks for example, and which trigger an alert for abnormal behaviours whenever a behaviour is not detected and not described in the model [Fok and Lingard, 2011; Celik et al., 2011a]. One major drawback is that the unknown behaviours are labelled as dangerous ones because they are not represented in the knowledge database. However an unknown behaviour is not necessarily an illegal one [DCNS, 2010].

On the other hand, the second category of systems tackles the anomaly detection from the other end. Such systems model the unauthorized behaviours using the legislation of the domain of application. Therefore, the abnormal behaviours are detected according to the observed events in the monitored area. These systems use several techniques from multi-agent based one to expert systems via neural networks. [Tan, 2005b; Jakob et al., 2010; Gupta et al., 2009; Nilsson et al., 2008].

Expert systems are coming from the artificial intelligence field and intervene in several forms in the behaviour analysis. Their operation is based on two principles: a knowledge database and an inference engine. The database is composed of events and rules representing the whole knowledge on the domain to monitor. The inference engine indicates a way to manipulate the events and the rules according to what we have to observe. Thus, if the aim is to detect abnormal behaviours, the inference engine might represent the modelled authorized behaviours and trigger an alert when a detected behaviour is not in the knowledge database. Some works are designing learning expert systems that model real-time behaviours and report abnormal ones, meaning unusual ones. The main drawback of such systems is the low adaptability of these systems to new behaviours. Indeed, the knowledge database can not be exhaustive and all unknown behaviours, or new ones, are not necessarily anomalies to detect.

Multi-agent based detection systems come in two ways. On one hand, there are systems where the agents are autonomous entities that are able to run through the different parts of the system. They assess and evaluate the components they are visiting and are able to report anomalies [Dasgupta, 1999]. On the other hand, there are systems where the agents are representing the entities in the monitored area. This kind of detection is simulation-based and the observation on the entities are compared to the agents behaviours in order to trigger relevant alerts [Tan, 2005a]. The use of multi-agent technology allow these systems to be more flexible but the drawback is that they remain unable to adapt themselves to new behaviours.

In the surveillance context, there are also systems based on chronicle recognition [Dousson and Ghallab, 1994; Cram, 2010] or intention recognition [Cohen et al., 2008]. The chronicle recognition is used to detect one-time events related one to another by temporal constraints happening in a given context [Dousson and Ghallab, 1994]. These events and constraints are often represented using graphs that are then used to define the knowledge database for the system. This is useful for a system where the events are known beforehand and can be represented. When the system to monitor is highly dynamic, events can appear or disappear at any time and a surveillance systems must adapt itself in

order to comply with the dynamism.

With the detection and the tracking of an object, there is often a behaviour analysis module that is able to detect anomalies and help the user to focus on these. We now focus on the different techniques of learning and understanding of scenarios.

2.3.2 Anomaly Detection in Surveillance Systems

The question is how to detect anomalies, or abnormal behaviours, on the built representation of the monitored area. When asked to the users of surveillance systems, they answer by their experience, their knowledge and their feelings. Back in 1983, Hogg [Hogg, 1983] proposed a system able to detect a walking man according to spatial constraints. This is an hand-crafted knowledge built in this system and there is no flexibility to adapt to new behaviours. For a fully automated system, such behaviours have to be learnt by the system rather than built-in.

First, an abnormal behaviour is a behaviour deviating from what is considered as normal. The detection of such behaviours is often seen as a classification problem [Gonzalez et al., 2002]. Indeed, the observations made on the supervised area have to be separated, classified into two groups, the normal and the abnormal ones. Therefore, it is common to use machine learning methods to tackle this problem.

However, it is difficult to apply common classification algorithms to real surveillance systems. Indeed, a training phase for these algorithms would require normal and abnormal behaviours examples but, most of the time, only normal ones are available in a sufficient amount to have an efficient training. Besides, all the abnormal behaviours are not known and the system might encounter one sooner or later. This means that the set of abnormal behaviours is virtually infinite, as well as the set of normal behaviours. The difficulty to consider the unknown is a major drawback of classification algorithms.

Regarding anomaly detection, a lot of works in the literature propose to model the normal behaviours these methods. Examples are the works of Denning [Denning, 1987], Kraiman et al. [Kraiman et al., 2002] or Jakubek and Strasser [Jakubek and Strasser, 2002]. But these systems are difficult to apply. Cai and Franco [Cai and Franco, 2009] point out that on of the main difficulty is the representation of the knowledge and of the input data used by the classification algorithms.

Besides the specific classifications algorithms, the abnormal behaviours detection techniques are also using methods and algorithm coming from the machine learning field. Thus, Ghosh et al. [Ghosh et al., 1998] use neural network to detect intrusion in computer networks while Lee et al. [Lee et al., 2000] are using a rule-based engine. In another domain, Bolton and Hand [Bolton et al., 2001] use clustering to identify credit card fraud and Janakiram et al. [Janakiram et al., 2006] use Bayesian networks to identify anomalies in sensor networks.

There are four families of anomaly detection techniques that can be identified [Chandola et al., 2009a], the classification-based, the nearest neighbour-based, the clustering-based and the statistical-based. The following subsections visit these techniques.

2.3.2.1 Classification-Based Techniques

Classification [Duda et al., 2012] is a way to learn models from a set of training data and then use the learnt model to associate input data to one of the identified classes. Classification-based anomaly detection works the same way, meaning that after a learning phase, the classifier is able to label input data as normal or abnormal. That works if a distinction is possible between normal and abnormal classes. Classification techniques can be multi-classes [Barbara et al., 2001], meaning that several classes are labelled as normal and the classifier is able to determine if an instance belong to one of these or to none, in which case it is considered abnormal. There is also the one-class classification [Roth, 2006] techniques where all the training data are used to determine one normal class and an instance that would not belong to it would be considered as an anomaly. Several classification-based techniques are available given a system.

Neural networks are the most known technique. The first step is to provide the neural network with training data to learn the normal classes. Then, when fed with the input data, the neural network processes it and the input is classified as normal if not rejected, as abnormal otherwise. Several works exist onto this base [He et al., 1997; Manevitz and Yousef, 2000; Moya et al., 1993; Kojima and Ito, 1999].

Another well known classification techniques are the Bayesian networks. Bayesian networks are able to provide data with a probability to be part of normal classes or anomalous classes after a training phase [Diehl et al., 2002]. This basic view of Bayesian networks has been improved to take into account the dependencies between variables in the considered systems [Das and Schneider, 2007; Janakiram et al., 2006].

Rule-based techniques are a way to learn the rules that model the normal behaviour, then when an instance does not comply with these rules, its behaviour is classified as an anomaly. Again, a training phase is needed in order to learn the rules and then the algorithm is able to process the input data and determine their normality, or abnormality [Mahoney and Chan, 2002; Kato et al., 2001].

Classification-based techniques are able to implement powerful learning algorithms and are fast in the processing of input data since the comparison data are precomputed during the learning phase. However these techniques rely on accurate label for the training data and the specification of the detected anomalies can be hard to obtain as the training data are more often focused on normal behaviours.

2.3.2.2 Statistical-Based Techniques

Statistical anomaly detection are based on the following principle: "An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed" [Anscombe, 1960]. Therefore, a statistical model for normal behaviour is built. Then, for data input, inference is applied in order to determine if it belongs to this model or not. If the instance of the data have a low probability to be generated by the defined model, it is considered as an abnormal one.

Statistical techniques can use a confidence score besides the statistical one and use it to refine the global score and the detection. One main interest of statistical technique is that the labelling of the training data is not an obligation. About the drawbacks, the main one is that these techniques suppose that the data is generated following a particular distribution, which is not always true; as a corollary, the choice of the best distribution to represent the data is not simple [Motulsky, 1995].

2.3.2.3 Nearest Neighbour-Based Techniques

These techniques are based on the idea that normal behaviours occur in the vicinity of each others while anomalies occur far from this neighbourhood. Thus this requires a measure allowing to represent the distance between two instances of data [Boriah et al., 2008].

One technique is to compute the k^{th} nearest neighbours of an instance. This can be defined as follows: *The anomaly score of a data instance is defined as its distance to its k^{th} nearest neighbour in a given data set*. A score is then associated to each instance of input data according to its relative distance to the core normal data and if this distance exceeds a threshold, the instance is considered as an anomaly [Eskin et al., 2002; Ramaswamy et al., 2000].

Another technique consist in taking into account the relative density instead of the neighbours by themselves. The system is able to compute the density of the neighbourhood of an instance and class it as normal (high density) or abnormal (low density) [Zaïane and Lee, 2002].

The neighbours techniques are purely data driven and do not need a priori knowledge of the data; they have better results in terms of missed anomalies since a high neighbourhood in the training data set is very rare; and these techniques are easy to use, providing a distance measure is defined for the considered data. On the drawbacks, these techniques require a distance measure and if it is not well designed, the system must not be able to give relevant results; it can be costly in term of resources to compute all the distance between each pair of instances during the learning phase; and the distance metric is the key point of the neighbour-based techniques and this can be a challenge for data with a high complexity or interdependencies.

2.3.2.4 Clustering-Based Techniques

Clustering-based techniques are close to neighbours-based techniques in the way that they postulate that similar data instances can be grouped into clusters [Jain and Dubes, 1988; Ertöz et al., 2003; Basu et al., 2004]. Therefore, anomalies do not belong to any clusters and can be identified that way. The difference consists in the metric. Where neighbours-based techniques use the pure distance between to instances, the cluster-based techniques take into account the cluster whom the instance belongs to.

Clustering algorithms can handle complex set of data, so it is possible to handle more complex data than with neighbour-based techniques; the test phase is faster because the number of clusters is small and constant over time (you compare the instance of data to a cluster and not to each other instance). Yet, clustering algorithms are the key point of these techniques and have a direct impact on the performance while detecting anomalies; besides, if the anomalies are forming a cluster, these techniques are rendered useless.

2.3.2.5 MAS-Based Anomaly Detection

As exposed, several techniques from machine learning domain can be used for anomaly detection. In machine learning, multi-agent systems have been proven useful to address complexity and dynamic issues. For this reason, MAS are more and more used in surveillance systems since the area and entities to monitor are more and more numerous and they adopt complex behaviours.

In this section, we detail the work of [Tan, 2005a] as an example in MAS-based anomaly detection. This work seems of interest to us since it is applied to maritime surveillance and it uses multi-agent system.

The aim is to track intent into harbour areas. To achieve this goal, the author propose a multi-agent system composed of four kinds of agents:

Data Agent: The data agents are the agent representation of the real world, they carry information for the multi-agent system from the various data sources. For example, a data agent might represent a vessel track with its identification, its flag,...

Cognitive Agent: These agents represent what the ships are not allowed to do. They are also called violation agents. For example, the *Speed Violation Agent* is used to illustrates the violation of speed limit in a given area. This is an agent representation of rules.

Violation and Cognitive Blending Agent: these agents are able to relate a data agent to a cognitive agent. For example, when a vessel has an excessive speed, the corresponding agent is associated to the Speed Violation Agent. One ship can be associated to several violation agents.

Intent Agent: Using the information available, local knowledge and the related violation

agents, the Intent Agents are able to analyse the behaviour of a vessel according and identify it as Friendly, Neutral, Potentially Hostile or Unknown.

Therefore, the multi-agent system presented in this work is able to represent both the vessels in the monitored area and the rules that they are not allowed to violate. Then, cognitive agents are used to integrate all this information and determine the suspicious level of the concerned ships. Ultimately, the Intent Agents are able to weight each piece of information in order to determine what are the most important in the current situation. It finally leads the identification of its behavioural nature.

This work is closely related to classification techniques. Indeed, it represents rules from the maritime legislation and it is able to sort the ship behaviours into normal and abnormal ones according to these rules.

One major drawback of this work consists in the design of the violation agent associations. Indeed, using violation agents, it is possible to build complex behaviours representations and then use it to detect such behaviours in the real world. However, it is a complex tasks and the it needs specific experts according to the considered domain: ship movement protocols and surface warfare threat assessment are used in this work. Here, the strength of the MAS approach is in the classification process and the complex behaviour detection.

2.3.2.6 Abnormal Behaviours Detection

Chandola et al. [Chandola et al., 2009a] state several drawbacks in the most common used anomaly detection system. They are for the majority issues that can be relate to the machine learning corresponding machine learning algorithms. First, when the data set is complex and use a high number of dimension to represent the training data, **the algorithms have difficulty to make a difference between normal and abnormal data**. There are techniques addressing these dimensionality issues, projecting the data to lower dimensions. But the resulting performance are dependent on the distinction between normality and anomalies in the projected space.

Second, as stated before, **classification based techniques require labels for both normal and abnormal behaviours** while, most of the times, only the normal ones are available. Besides, when labels for anomalies are available, the balance between the two sets of labels is rarely reached and the learning phase is impacted because too few anomalous examples are available. To tackle these issues, semi-supervised learning are efficient since they only use normal labels, but require a sensitive enough measure to detect single anomalies. They are indeed best effective when the number of anomalies is growing.

Third, the majority of anomaly detection techniques relies on **the possibility of a good distinction between the normal behaviours and the abnormal ones**. When the border between the two groups is fuzzy, the detection algorithms are less efficient and are not able

to provide good results. The same issue exists when an anomaly is coming from combined normal behaviours.

Fourth, there are **considerations on the application of these systems**. The classification based techniques require a training phase but then are fast when running in real time. However, they are not flexible and require a new training when applied to another domain or when the context of application changes. On the other hand, the nearest neighbours-based techniques have no training and the detection phase can be quite time consuming, which is not viable in real-time and use. However, these last kind of techniques are more suitable to dynamic context, if the time is not a key feature.

Finally, the anomaly detection techniques make the assumption that the anomalies are rare against the normal behaviours in the context where they are applied. This is often true but there are cases where a huge amount of abnormal behaviours can overflow the normal ones. One simple example is the spreading of a worm in a closed computer network. If not detected in time, the majority of computers can be infected and the number of abnormal behaviours might be larger than the number of normal ones. Sun et al. [Sun et al., 2007] and Soule et al. [Soule et al., 2005] use semi-supervised techniques to address this point.

The efficiency of anomaly detection techniques is highly dependant on the domain where it is applied. Indeed, given the available input data, the desired output and the context of application, as well as its constraints and requirements, one technique is proven to be more useful than the others. Besides, this is also dependent on the notion of anomaly. An anomaly in computer network is not the same as an anomaly in maritime surveillance. The works on abnormal behaviours and anomalies detection lack an unified technique that can be applied in several domains, without too much tuning before it is operative and effective. Therefore, surveillance systems lack a **generic and efficient** method of anomaly detection.

2.3.3 Maritime Surveillance Systems

In the recent years, the maritime activities have grown rapidly. It now provides a support for numerous and various traffics, for example arms dealing and drug smuggling, or illegal activities like illegal fishing. Besides, human operators involved in maritime surveillance use systems able to supervise more and more large areas thanks to various types of sensors deployed on the coastlines and on mobile platforms. As a consequence, these operators need help to identify criminal activities in order to deal with the threat they represent. These activities are embedded among the behaviours of the thousands of ships evolving in the supervised area. Thus, efficient automated surveillance systems is a key point in the maritime domain [Henocque and Lafon, 2011].

Significant technical progress have been made in wide maritime area coverage by different sets of sensors [Rasheed et al., 2011], in heterogeneous data processing and fusion [Battistello et al., 2011] and in detection of abnormal behaviours methodologies [Wang et al., 2011] that could be usefully integrated together to build up

a new generation of maritime surveillance systems for efficient security applications in high density traffics areas [Aliakbarpour et al., 2011].

2.3.3.1 Area Coverage and Data Fusion

Nowadays, thanks to the technological advances, it is possible to use more and more powerful sensors and radars to monitor wider and wider areas. From high frequency radars to remote camera for direct visualisation, a large panel of tools are nowadays available for the operators and the people involved in surveillance. It is not different in the context of maritime surveillance.

However, these equipments are specialized. For example, visual recognition system are able to work within a limited range and are influenced by the weather and natural conditions. The coastal radars are able to identify a target tenth of miles away but are very sensitive to noise and their use can be dangerous for planes or boats in their cone of wave diffusion. Finally, each kind of system has been designed in order to be a complement to the others.

That is why the data fusion is a major field of study in maritime surveillance domain. Data fusion is *the process of integration of multiple data and knowledge representing the same real-world object into a consistent, accurate, and useful representation*. The application in maritime surveillance is to regroup all pieces of information available about an object, mostly a ship, and to provide their visualisation for the operators [Vespe et al., 2008].

The majority of works in this domain focus on target tracking using multiple visualisation means [Guerriero et al., 2008; Di Lallo et al., 2006]. These works are based on the combination of several kinds of sensors (Automatic Identification Systems, video cameras, High Frequency radars,...). This has been implemented in several projects, for example Stradivarius[1] or AMASS[2].

But one central feature that is increasingly studied these last years is the anomaly detection in maritime surveillance.

2.3.3.2 Anomaly Detection in Maritime Surveillance

Being able to identify and track a target from one end to the other of its trip is not sufficient when it comes to threat assessment and the identification of anomalies. Anomalies are what is not considered as normal according to the domain and the context, whatever the reason is. For example, in maritime surveillance, an anomaly could be the disappearance of a ship from the radar echoes. According to Kazemi et al. [Kazemi et al., 2013], we can divide the anomaly detection techniques into two groups: the data driven and the knowledge driven ones.

[1] http://www.pole-mer-bretagne.com/stradivarius.php
[2] www.amass-project.eu

The data driven techniques use all the available data in order to identify the anomalies. This data is coming from the sensors and materials used for the detection of tracks (see section 2.3.3.1) but also from other sources like databases maintained by the concerned authorities [Andler et al., 2009] or from open data available via different sources [Kazemi et al., 2013]. Then, algorithms are applied to this data in order to identify the anomalies. The most common is the use of a rule-engine like method where the data is processed through a set of rules and an anomaly is detected when a rule is violated.

On the other hand, the knowledge driven techniques are set up in order to detect anomalies according to the relation between the tracked objects. Several works are using such techniques [Roy, 2008; Tan, 2005b; Auslander et al., 2012]. The works of Roy is a rule-based engine reasoning on data about the situational facts on the ships and detecting anomalies when against the rules. The works of Tan use a multi-agent system in order to represent normal behaviours in harbour areas and an anomaly is detected when a ship does not follow a known behaviour, meaning a behaviour recorded by the multi-agent system.

One of the main interest in the knowledge driven approach is the abilities to use it in order to prevent the threats. Indeed, data driven approaches are based on data about the ships in the monitored area, this means that it has to be available to perform the detection. In other words, this means that facts are needed before the related threat can be identified. In another domain, that the same thing as seeing the gun before knowing that the robber is here to rob the bank.

With knowledge-driven techniques, you could use the knowledge on the target in order to identify and prevent the threat. This has been used by Jakob et al. [Jakob et al., 2010] in a multi-agent simulation aiming at the identification of the most secure way to build ship convoy in order to avoid maritime piracy. Other similar works are [Vaněk et al., 2012; Komenda et al., 2013]. The common point of these work is the use of simulation in order to prevent risky situations and build the appropriate approach in order to avoid it.

One common drawback of these techniques is the use of expert knowledge to build the simulations or to build the rules used to detect anomalies. However, it is hard to avoid the involvement of the experts for two reasons: first, the maritime domain is a complex one and several reasons and information can explain a situation, which can be an anomaly or not, depending on only one piece of data that can be overwhelmed by a lot of others; second, the operators and experts involved in the maritime surveillance prefer to rely on their experience and their own knowledge to identify an anomaly. This is a common syndrome when one is faced with the machine. Some studies show that this could be tackled if the operators involved in the maritime surveillance are involved in the learning process of the machine and are afterwards able to send feedbacks to correct the anomaly detection algorithms [Claisse et al., 2010; Brax et al., 2013b]. Ultimately, the focus in maritime surveillance systems is on the **detection of abnormal behaviours from several entities with complex behaviours and interaction in wide areas**.

2.4 Conclusion & Discussion

In this state of the art, we expose the different families of machine learning generally used in computer science. The three main approaches are the supervised learning, the unsupervised learning and the reinforcement learning. These techniques are widely used and present several advantages and drawbacks.

First, *supervised learning* need training data set provided by experts in order to assimilate correct outputs for the learning system. When running, the inputs are compared to the knowledge of the learner and the most relevant output is given. This is useful and efficient when large sets of training data can be gathered and used. Also, such systems provide accurate results if they are well trained and if the inputs are well fitted to what we want to observe. This means that the syntax of the inputs have to be correct or the learner would not be able to understand it. However, **the main drawback of supervised learning is that it is time and resource consuming**. Indeed, to be efficient, several training sets have to be used and this takes time. More, when analysing an input, the learner has to compare it with all its knowledge. Another disadvantage is that supervised learning has **difficulty to handle dynamics of the system it learns**. Each time a new knowledge appears on the system, a training phase has to be done again. And when facing unknown inputs, the system is not able to process it efficiently.

Second, the *unsupervised learning* techniques do not require the intervention of a user to provide outputs. Data are gathered and the learner is able to sort it out according to various parameters. Such systems are designed to analyse large pools of data to identify clusters and hidden structures. These are efficient systems to clear data and provide some information before a more precise analysis. Like supervised learning, **the unsupervised learning is time and resource consuming** for similar reasons. Besides the resource cost, unsupervised learning is well suited to identify group of data but is **unable to label them or to use them**. Experts are needed to identify these groups and validate them. More, the identification of the clusters is **highly dependent on the methods used to compare data** and small variations can bring huge differences on the outputs.

Finally, reinforcement learning brings a new approach to learning techniques. It is performed by entities that are able to perceive the results of their actions, to evaluate them and propose the best one. Thus, this is a learning using feedback as rewards or punishments according to the objective of the learner. Multi-Agent Systems are well suited for reinforcement learning since each agent can be a learner. However, in reinforcement learning, several issues have to be addressed. First, the **data representation for the agents has to be designed according** to their characteristics and their skills. If the agents are not able to apprehend the data, they will not be able to process them through the learning abilities. Then, another main drawback in reinforcement learning is the **definition of the function used to determine reward or punishment**. This function depends on the context and the domain, but also on the data to analyse.

The second part of this state of the art exposes a surveillance system overview. A surveillance system is a system designed to track and monitor entities, their activities and their behaviours. There are two main aspects to take into account: the *detection and the tracking of entities* and the *behaviour analysis*.

Detection and tracking of entities is performed by several arrays of sensors and detection devices, like radars, video cameras,... Data is then processed and fused in order to build a representation understandable by the users of the system, the surveillance operators. These operators are then in charge of the second aspect, the behaviour analysis. According to what they observe and their experience, they are able to identify situation to handle: from threat to answer to rescue.

With the increasing number of entities to monitor and the increasing size of the monitored area, the need for an automatic behaviour analysis arises. This allows the analysis of the behaviours and the alert triggering when an abnormal one is identified. Therefore, several systems use learning techniques in order to perform it. For example, supervised learning is used to compare observed behaviours with database of known behaviours and raise alerts on unknown ones. Another example is a multi-agent system representing the legislation rules and the entities. When two of these agents are grouped together, the system identify an abnormal behaviour.

As the automated behaviour analysis uses several techniques coming from machine learning, several similar drawbacks are to be noted. The **cost in term of resource can be high** depending on the number of entities and the size of knowledge databases. More, these systems are efficient when the domain is well known and modelled, but the main interest of surveillance systems is to **monitor complex and dynamics environment**. Finally, it is **difficult to design a generic behaviour analysis systems** since the domain, the context, the entities and the behaviours are different from one system to another. Somehow, the **experience and the knowledge of the operators** carry the differences between several system and even within one between the alert and the normality. It also impacts the learning process, needed to train the system. Thus, designing such systems is a complex and long task.

In light of this chapter, we identify several lacks automated behaviour analysis, both for the alert triggering and the learning to perform it. An efficient alert triggering system has to use a **generic representation of entities and their behaviours** in order to be used in several systems. Also, learning process should **take into account the knowledge of the operators**. Ultimately, for more efficiency, such a system should be low resource cost and able to learn whilst used.

3 A Multi-Agent Generic Model for Surveillance Systems

« Creativity and new ideas are solutions of all problems. »

Alfred E. Van Vogt

Contents

The chapter in English starts page 53.

Résumé général du chapitre

Le contexte théorique et applicatif de ce travail a été présenté dans le premier chapitre. Le fonctionnement général d'un systéme de surveillance y est exposé ainsi que les principaux enjeux de ce domaine. Plus spécifiquement, dans le cadre de la surveillance maritime, il faut ***permettre une couverture totale et permanente des frontiéres maritimes****, approvisionner le systéme avec des informations en provenance de différents capteurs, fournir une* ***détection automatique des comportements anormaux*** *et* ***comprendre les menaces*** *qu'ils peuvent représenter et enfin* ***offrir une interface adaptée aux utilisateurs****. Nous nous sommes concentrés que la détection automatique des comportement anormaux.*

La seconde partie de ce chapitre développe les concepts utilisés lors de la mise en œuvre de notre modéle. Les notions ***d'agents et de systémes multi-agents*** *expliqués et plus particulièrement* ***la théorie des AMAS*** *(Adaptive Multi-Agent Systems). Cette théorie est basée sur la notion de coopération entre les agents qui composent le systéme. Cette coopération se fait de maniére locale et autorise l'émergence de la fonction globale á atteindre. Cette théorie est utilisée le cadre de ce travail car elle est particulièrement adaptée dans le cas oú le système a représenter est* ***complexe, dynamique, ouvert et non entièrement connu****.*

Dans ce chapitre, la contribution principale de ce travail est présentée. Elle consiste en la définition du ***modèle agent MAS4AT*** *se basant sur la théorie des AMAS. L'objectif de MAS4AT est de s'attaquer á deux aspects principaux d'un systéme de surveillance : la levée* ***d'alertes pertinentes*** *et* ***l'apprentissage des évènements****. Certains aspects de la théorie des AMAS ont été utilisés lors de la conception des agents, á savoir* ***le caractére local des agents*** *et* ***la résolution de certains situations de non coopérations****.*

Nous avons identifié le manque d'un système générique et réutilisable qui puisse fonctionner en temps réel dans le domaine de la surveillance. Il apparaît que lors de la définition d'un tel système, la levée d'alerte est primordial. En effet, la surveillance concerne l'observation d'entités dans une zone définie et des alertes sont lancées quand une situation suspecte est identifiée. Une situation peut être définie comme étant une séquence d'événements. Dans MAS4AT, nous proposons d'attribuer une valeur á chaque événement indiquant son niveau de suspicion. Ainsi, la somme des événements donne un niveau équivalent pour une situation donnée. Lorsque ce niveau dépasse un seuil d'alerte, la situation est jugée anormale et l'entité associée passe en alerte, ce qui signifie que son comportement est suspect.

Nous sommes donc capables, dans MAS4AT, de représenter une situation par une fonction mathématique représentant la séquence des évènements ayant amené à la-dite situation. Mais les valeurs des évènements ne sont pas connus a priori. En effet, lorsque cela est fait par les opérateurs de surveillance, ils évaluent inconsciemment les évènements qu'ils observent et sont capable de dire si ils sont importants ou non à un instant donné. Pour cette raison, mais aussi de par le nombre important

et inconnu d'évènements possibles, nous avons doté MAS4AT de mécanismes d'apprentissage. Ainsi, lorsque le systéme lève une alerte, l'opérateur a la possibilité de corriger le système et de lui indiquer une erreur. Les agents peuvent alors s'auto-ajuster pour répondre á ce feedback et proposer des alertes plus cohérentes par la suite.

Aprés une bréve introduction, la deuxième section de ce chapitre présente les deux problématique associées à la détection de comportement anormaux dans un système de surveillance : ***la levée d'alerte et l'apprentissage****. La troisième section détaille le système á base d'agents MAS4AT selon ces deux aspects. Enfin, le chapitre se conclue par un rappel du modéle et de ses atouts.*

3.1 Introduction

The current work falls into the domain of anomaly detection techniques. The state of the art exposed the different existing techniques in anomaly detection and spotted their qualities and drawbacks. It appears that these techniques are very useful and efficient as long as they meet the following requirements:

- The domain and the context are well known and the designers of the system are able to model and represent it in a understandable way for the system.
- The users are able to provide a training data set in order to be learnt by the system. These data have to be as complete as possible for a good detection in the running phase. Also, time is a key point for this training.
- The complexity and the scale of the system have a direct impact onto its efficiency, but also onto the resource cost.

As a consequence, it appears that for each new anomaly detection system, the whole process of design has to be done again. Even the learning techniques used in such systems are dependant on the domain and the context for the best efficiency.

This chapter is the main contribution of this thesis and aims at filling the lack of a reusable system in several surveillance contexts. Section 3.2 exposes the problematic we tackle. This problematic is twice: first we address the alert triggering process and expose the stakes and requirements of such techniques; second we introduce the learning in the anomaly detection.

Section 3.3 exposes the design of a *Multi-Agent System for Alert Triggering*, called *MAS4AT*. This section follows the same steps as the previous one: First we develop the multi-agent system that is able to detect anomalies and trigger alerts. Second we extend the system in order to introduce the ability of learning.

The use of a multi-agent system has been discussed in the chapter 1. In a few words, the agent-based techniques seem to be the most suitable to answer the complexity of nowadays' applications and are able to solve a wide range of problems. In surveillance systems, the

objective is to trigger alerts as a consequence of the detection of abnormal behaviours. This is a global objective coming from several local entities. Indeed, studying the global system is not efficient to spot one abnormal behaviours amongst hundreds of others. However, studying each entity separately is a better strategy. Therefore, the Adaptive Multi-Agent System theory provides a way to handle this by focusing on the behaviours of the entities and their local interactions.

Finally, the conclusion (section 3.4) exposes the full model of agents used in this work and discusses their use.

3.2 Surveillance Systems Problems

In surveillance domain, there is two major features which are addressed once the methods and techniques are designed. First, there are questions about the alert triggering provided. Indeed, this is the logical consequence of the anomaly detection. The general question is when and how to trigger an alert? Second, it concerns the learning in a surveillance system. We have seen that learning is a plus in a surveillance system, either before or while running. Several techniques may be used but none are generic or fast enough. By "generic", we mean that these techniques are specific to a domain and are not applicable to other ones. The two following subsections expose the problematic related to these aspects of surveillance systems.

3.2.1 Alert Triggering Problematic

Surveillance systems are designed to monitor the behaviour of entities in a given area. One of the main interest in such systems is the detection and identification of abnormal behaviours which can lead to threats or dangers.

Several systems exist that are able to track entities and identify such behaviours. In the literature, it is common to qualify it of anomaly detection. There are a lot of techniques for it, according to a context and a domain. The anomaly detection leads to the alert triggering.

Currently, if surveillance systems are widely present in our society, they are usually designed for a specific purpose. Each time a new one is needed, the whole design and implementation process has to be done from the beginning again.

There are several reasons for this. First, the context of the system influences the tracking of the monitored entities. It is not possible to track a plane in the same way than a person in a crowd. Aggregation and tracking techniques have arisen to address these issues and are not investigated in this thesis.

Second, the main problem is how to detect an abnormal behaviour? Most of the time, surveillance systems rely on the experience of the operators. One of the reasons is that it is difficult to represent such a behaviour, or to identify the events leading to it. Finally, when

a potential threat situation happens, only the experience of the user gives the explanation for it. Even if the system is able to provide information on the course of events leading to it, only the analysis by an expert is able to validate it.

If we take the most common example,i.e. the surveillance of a crowd, an automatic system can't perform a good anomaly detection for one simple reason: human behaviour is so unpredictable that it is not possible to identify and model all the possible ones. And even in this case, one behaviour can have several reasons to appear and only a few are real threats.

It appears that when a situation arises to be analysed, the experts recount the events that led to it. Only then they are able to propose a diagnosis and identify the situation as an anomaly or a normal one. In order to build a generic representation for a situation, two parameters have been identified:

- The **importance of the events** that are involved in a situation. This importance is defined by the context and the knowledge on the domain. For example, in maritime surveillance, a *stop event* is less important that a *fire report event*. This parameter can be represented as a numerical value.
- The **weight of the events**. Indeed, according to a situation, each event involved in it might not have the same weight than the others. It depends on the context and the time between the event and the current situation. For example, an event that happened five days ago might be less weighted than an event that occurred one hour ago.

These two parameters seems sufficient to represent a situation as long as it is possible to decompose it in several events.

Thus, we propose the function 3.1 as a representation of a given situation which is the combination of several events. This representation allows to associate a numerical value, r^t for each situation: this is the score of the situation. This score is the result of the sum of several events, a_i, each one weighted, w_i^t, by its importance in the current situation. This importance can be the number of appearance of the event or anything else that is considered as useful as long as it is representable by a numerical value. It has to be noted that a situation is temporally located and the next time step –second, hour, month,...– can see it changed beyond recognition.

$$r^t = \sum_{i=1}^{n} a_i \times w_i^t \tag{3.1}$$

This representation is close to the human one. When an expert assesses a situation, he gives a score to the course of events, weighted by his observation and the information he has. Then, given the final score, he can identify the eventual anomaly. We advocate that this representation is the most generic one for a situation.

The values of the weights come from the observation on field and can be easily quantifiable. Then the main issue concerns the value associated to each event. The question

is how to value an event when it is dependent on the context, the knowledge, the target,...? Moreover, the system to represent is dynamic and open. To answer it, MAS-based learning methods seems to be necessary.

3.2.2 Learning Problematic

In the previous section, we have introduced a representation of a situation (see function 3.1) using numerical values. It is a sum of weighted coefficients. However, the values of the coefficients are hard to quantify. This is why a learning process seems to be useful in a surveillance system. In fact, lots of surveillance systems integrate techniques to learn new situations and to be able to identify them afterwards.

However, this learning is, most of the time, dependent on the context of the system. Besides, it is based on known representation of the entities and/or the events taken into account. For a generic system based on the numerical representation we have introduced, the values of the events and data are unknown and not possible to give, or with high difficulty.

One solution is to ask for the user of a system to quantify each situation when it appears. With time, the system would be able to give a score for all the possible events. But the cost in human resource would be far too high, providing there are no errors. And what would be the confidence of a system that has to be initialised from scratch? Is it worth the time spent to configure it?

For this reason, a system able to learn the values of the different events with a minimal human intervention is required. So far, it would not be wise or advisable to leave the operator apart. Indeed, he is an expert of the domain and the final user of the system, which means that the system should be compliant with him. Here, compliance means that the identifications performed by the system should be reliable, meaning accurate according to the human operators.

So, the problem is to design a system able to learn the unknown values of a function without the need for an expert to value them one by one. In this work, we propose a solution to tackle this problem and we design a system able to learn these values using feedbacks. Indeed, rather than giving each value of each event, the operator is only asked to correct the system.

In a row, the system is able to detect abnormal behaviours and raise an alert toward an operator. Then, the operator is able to correct the system when an alert is wrongly triggered or when it is missing. Alongside the numerical representation, we think that this method is well suited for being as accurate as possible and as close as possible as the operator knowledge and experience.

3.3 Surveillance and Learning, a Generic Model

We are interested into two aspects of surveillance systems, the alert triggering and the learning process. Given the identified problematic, we propose a multi-agent system to address these concerns. In a first step, we focus on the alert triggering and propose an agent model for this specific part of the system. In a second step, we extend the model in order to add the abilities required for the learning part.

3.3.1 A Model for Alert Triggering

3.3.1.1 Architecture of a Multi-Agent System for Alert Triggering

Surveillance is the act of monitoring entities in a area covered by a pool of sensors. The main work of this thesis concerns the automation of the anomaly or abnormal behaviour detection. In order to achieve this goal, the system must be able to represent the real-world in an understandable and usable way. Two specific points have been identified.

The first point of interest are the monitored entities. In a multi-agent system, an entity is usually represented by an agent. The assumption is made that at least one of the other components of the surveillance system provides an efficient way to detect and track the wanted entities. Using the agent representation, the multi-agent system is then able to propose an agent-based view of the real world. The design of the agent is described in the section 3.3.1.2.

The second point of interest in surveillance systems are the data that can be gathered on the entities. This data can be of two main kinds: on one hand the data that are general, such as the identification of the entity, its color, its name,.... This data is aggregated by several knowledge sources and observations –direct or not– on field. On the other hand, there is the punctual data that are detected at a given time: stop of an entity, disappearance or appearance,...This data is the most interesting since it is a direct visualisation of the behaviour of the entity.

Once it is possible to describe the events that can happen on an entity, there is the need to give an importance to these events. The importance of an event is its value according to the domain and the user. For example, in the maritime domain, if a tanker stops in open sea, it is generally more important than if it is a fishing ship, for which a stop can be considered as normal. Following the works of Mano et al. [Mano et al., 2010], we have restrained the importance of an event to three parameters.

The initial importance: When something happens, it has a first impact that can be assessed. In surveillance systems, there is the same impact when an event happens on an entity. The operator is able to assess the event and rank it among the others. For example, in building surveillance, a fire is far more important, at first sight, than an unclosed door. This parameter is the numerical representation of this impact.

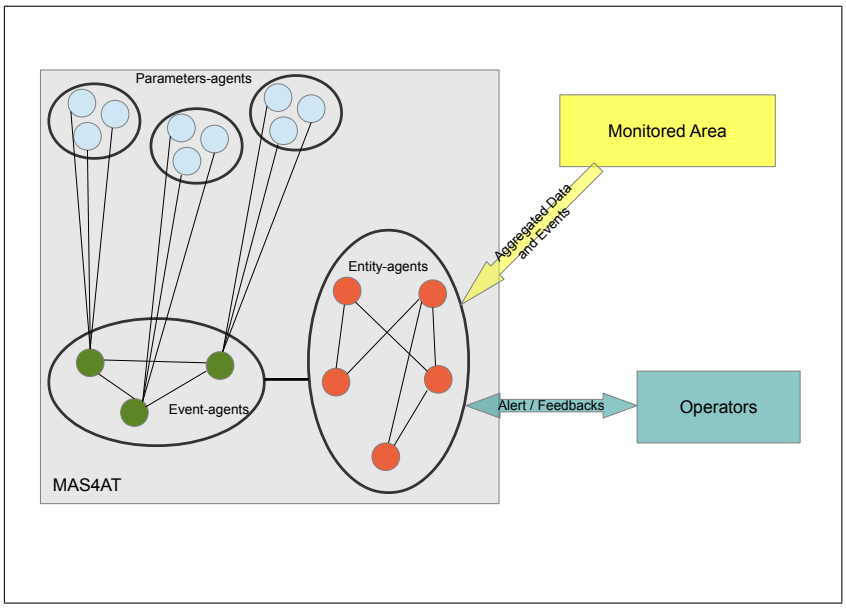

Figure 3.1: Architecture of MAS4AT.

The increasing importance: The increasing importance of an event is an additional parameter to the initial importance. This is the representation of the growing importance of the event while it lasts. Indeed, the operator can assess a minimal initial importance for an event but keeps an eye on it and decides to investigate it furthermore if it lasts too long. This parameter is the numerical representation of this importance and is used each time step that the event lasts.

The decreasing importance: An event can disappear on an entity, meaning that this is the end of the event. For example, the ship starts again. But this disappearance does not mean that the operator will do as if it never existed. On the contrary, the operator keeps track of all the events that happened for a certain time window. These events are then used to assess a situation when it arises. Of course, keeping a track of all the events is not possible nor wanted and the most former events are gradually forgotten. This parameter is the numerical representation of this forgetting factor.

Thus, for each event, we add three parameters in order to provide a more accurate description of the event.

In a general surveillance system, we have identified three structures that are required for the surveillance activity: the entity to monitor, the event that can happen on the entity and the parameters used to characterize the event. In multi-agent systems, these three concepts are represented by agents. Figure 3.1 shows the global architecture of the system.

MAS4AT is so composed of three kinds of agents: the entity-agents that are a representation of their real-world counterparts, the event-agents that are the representation of the events that can happen on an entity, and the parameter-agents used to refine the event

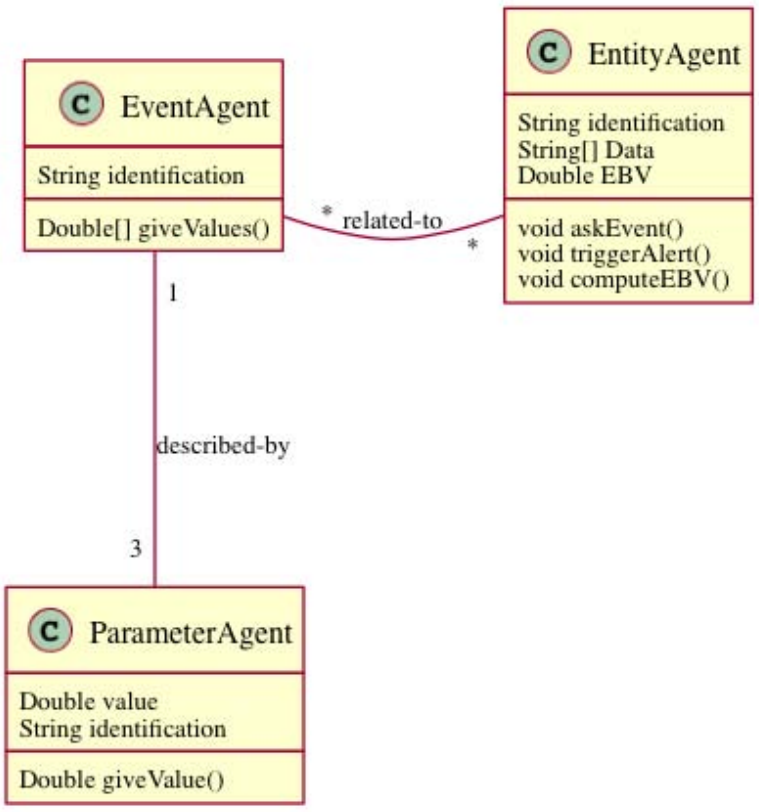

Figure 3.2: UML Model for Alert Triggering (a)

agents.

The figure also shows the interactions of MAS4AT with the other components of the surveillance system. First, there is the component that provides aggregated data from the sensors and the knowledge databases. This component sends information on the identified entities as well as the eventual detected events on it. Second, MAS4AT communicates with the operator involved in the surveillance systems. Indeed, the aim of MAS4AT is to trigger alerts when an anomaly or an abnormal behaviour is detected. Besides, the operator is able to send a feedback about the sent alert. This is more investigated in section 3.3.2. The interactions between the various agents are investigated in section 3.3.1.2.

3.3.1.2 Description of the Agents

The skills and characteristics of the agents of MAS4AT are shown in the UML class model of figure 3.2. This model is focused on the alert triggering.

First, we focus on the entity-agents. They are the representation of the entities monitored in the real-world, meaning that for each entity, a corresponding agent is created in the multi-

agent system. This agent is given three characteristics.

***Identification*:** This is the unique identification of the agent. This can be related to the unique identification of the corresponding entity for efficiency.

***Data*:** This is a structure containing all the knowledge on the entity that can be useful for the alert triggering process, as well as the information that could be used by the operator. The main purpose of this characteristic is to pass information from the system to the operator.

***EBV*:** EBV stands for *Entity Behaviour Value*. This a numerical representation of the behaviour of the entity. The more this value is high, the more the entity has a suspicious behaviour. The computation of this value is explained below.

In addition to this set of characteristics, the entity-agents are given a set of skills allowing them to trigger alert when needed and according to their characteristics.

***askEvent*:** When an event is detected on an entity by the surveillance system, it is echoed on the corresponding entity-agent using this skill. This allows the entity-agent to take into account new events and use them for the alert triggering process.

***triggerAlert*:** This is the main purpose of the entity-agent, the ability to trigger an alert toward the operator. An alert is triggered when the value EBV exceeds a threshold defined by the operator.

***computeEBV*:** This skill enables the entity-agent the ability to compute its EBV. This value is computed according to the function 3.2.

$$EBV = \sum_{i=1}^{n} (Init(e_i) \times nb(Init, e_i) + (Incr(e_i) \times nb(Incr, e_i) - (Decr(e_i) \times nb(Decr, e_i) \quad (3.2)$$

This function cumulates the importance of the different events and data, e_i, happening on an entity according to the values of their parameters and taking into account how long these events last in time. This function can be used to provide a graphical representation of the behaviour of an entity-agent as a curve and it also gives the corresponding inequality considering the threshold fixed by human operators. More details on EBV can be found in section 3.3.1.3.

To sum up, the entity-agent is a virtual representation of the real-world entities in the monitored area. It is able to receive and take into account the detected events and data on the entity in order to ask the value of the events and compute the Entity Behaviour Value. Finally, if this value is above a given threshold, an alert is triggered.

Second, we focus on the event-agent. Concerning the alert triggering alone, these agents are rather simple. They possess one characteristic for their *identification* and one skill,

giveValues, allowing the agent to retrieve the values of their own parameter init, incr and decr and transmit it. The event-agent is a representation of the event happening on the entities. For each event, an agent is created or used if already existing. For all the entities, there is a pool of unique event-agents. Indeed, there is no need to create one set of events for each entity as their signification would be the same.

One key feature of the use of a multi-agent system is that the number and nature of events is not a required knowledge beforehand. Each time an event is detected, if it is not known by the system, a new agent is created and can be used by the entity-agent.

Finally, the last class proposes a model for the parameter-agents. For the same reason as for the event-agents, these are quite simple. They consist of an identification unique within the event and giving their meaning: the initial parameter, the increase one or the decreasing one. The second characteristic is the value of the parameter. This value is given at the creation of the agent and can evolve during the learning, see section 3.3.2. Finally, the skill *giveValue* allows to retrieve the value of the parameter in order to be used by the caller.

This model provides the description of three agents designed to trigger alerts in a surveillance system. The entity-agent is able to compute the behaviour of the entity they represent in a given situation. To achieve it, each entity-agent is related to a set of event-agents: the agents representing the events involved in the current situation. The entity-agent must know the values of the parameters of the events happening on it, this is given by the corresponding characteristic in the parameter-agent. Thus, each event-agent is associated to three parameter-agents describing its internal characteristics. The values of the parameters are arbitrary initialised at the first start of the multi-agent system. The learning process described in section 3.3.2 allows the parameter-agents to modify their value according to operator feedbacks. Finally, the entity-agent is able to decide if an alert has to be triggered or not.

3.3.1.3 Computation and Use of EBV

The Entity Behaviour Value of an entity is designed to represent the behaviour of a value using a graphical curve, as illustrated in figure 3.3. Figure 3.3 shows that for each step, an entity-agent can compute its EBV and can deduce the corresponding inequality compared to the threshold given by the operator. When the computed value is over the threshold, the entity behaviour is considered suspicious and an alert has to be raised by the corresponding agent.

For example, at the fifth time step, the behaviour of the entity can equals:

$$EBV_{S5} = 1 \times Init_{e1} + 5 \times Incr_{e1} < threshold \tag{3.3}$$

This means that for the given entity and the situation S, the entity suffered one time the event *e1* and that it lasts for five time steps. The result is under the threshold so the entity-agent does not trigger an alert.

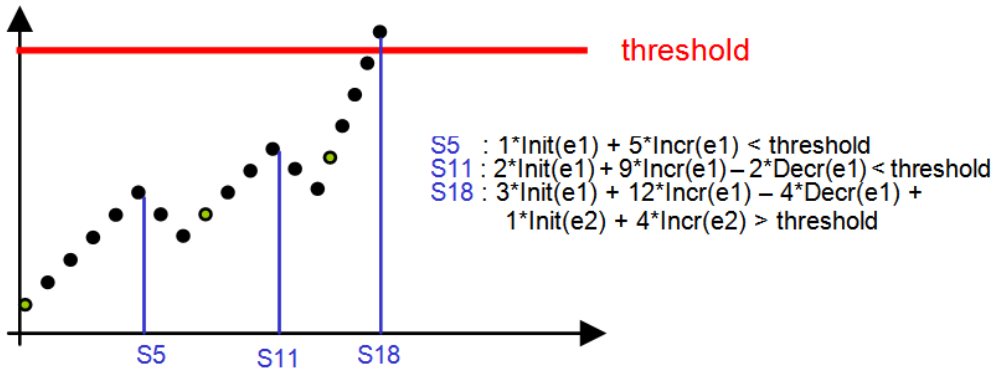

Figure 3.3: Graphical Representation of Entity Behaviour (example).

Six time steps after this situation, we focus on the situation S11, represented by the following equation:

$$EBV_{S11} = 2 \times Init_{e1} + 9 \times Incr_{e1} - 2 \times Decr_{e1} < threshold \tag{3.4}$$

This is an extent of the previous situations since the event *e1* lasts two more time steps and disappear for two time steps. Again, there is no need to trigger an alert for the entity-agent.

Finally, the situation S18 is given by the function:

$$EBV_{S18} = 3 \times Init_{e1} + 12 \times Incr_{e1} - 4 \times Decr_{e1} + 1 \times Init_{e2} + 4 \times Incr_{e2} < threshold \tag{3.5}$$

At time step 14, the entity suffers the event *e1* and the event *e2* at the same time. The equation translates it. This time, the resulting EBV is superior to the threshold, so the related entity-agent triggers an alert toward the operator.

When factorized, the inequality 3.5 can be written under the form of the inequality 3.6. This inequality is a direct application of the function 3.1 used to describe a situation. The added variable is the representation of the alert threshold while the sign of the inequality represent the current state of the situation: alert or not alert.

$$\sum_{i=1}^{n} a_i \times w_i^t \gtrless threshold \tag{3.6}$$

This means that for each situation in a surveillance context, as long as a valuation is possible, it is possible to represent it under the form of an inequality and numerical values exploitable by agents or algorithms. Then the comparison of the EBV with the threshold allows the agent to decide if an alert has to be sent or not. This provides and ensures a simple design of the agents and their behaviours.

3.3.1.4 Collective Behaviour of the Agents

One of the main advantages of the use of a multi-agent system is the ability to give the agents the tools for interactions and fill them with behavioural skills. In term of surveillance systems, it could be useful to transcribe interactions between the entities in the monitored area. Indeed, various cases might be difficult to identify since they are the result of several individual behaviours, these behaviours being considered as normal when considered one by one.

Obviously, it is unrealistic to try to represent all the possible collective behaviours for the reasons exposed in chapter 2. To deal with these issues, we introduce the concept of stigmergy [Grassé, 1959b] in our system. Stigmergy is a mechanism providing indirect communication between agents by leaving marks in their environment. The most common example is the pheromone tracks left by the ants. In regard of stigmergy, an ant has two main abilities: one, it can left a pheromone track in its environment; and second, it can perceive a pheromone track in its environment. Using these two skills, the ants are then able to communicate with each other in order to provide information: danger ahead, location of food,... As the multi-agent system techniques are based on the observation of natural processes, it was not long since this mechanism was used [Ranjbar-Sahraei et al., 2012; Serugendo et al., 2011].

As a consequence, the entity-agent have been modified in order to provide skills to use stigmergy-like abilities. Figure 3.4 describes the new UML model.

Firs, the entity-agents need tools to process a stigmergy-like communication. To achieve this, we provide them with the skills *markEnvironment* and *perceiveEnvironment*. The first one allows the entity-agent to leave a mark in its environment in order to give information. Practically, this mark is left whenever the entity-agent is subject to an event and it is associated to an evaporation factor meaning that it is temporary only. Besides, the main interest of this method is that the mark is only local. Then, when an entity-agent is in the vicinity of a mark, it can perceive it with the appropriate skill and it takes it into account like any other events. To sum up, a mark is an event created by an entity-agent and this event is local and temporary.

Second, the entity-agents are given the ability to ask the EBV of the other agents in their neighbourhood. The neighbourhood is composed of the agents in the vicinity but also the agents that have created the perceived marks. The *askEBV* skill is used to directly ask for the EBV of another agent, as an information. The *giveEBV* skill is used to answer such a request. This is useful for an entity-agent to know the EBV related to a mark. Indeed, a mark left by an entity with a high EBV is more important than a mark left by an entity with a very low one.

Finally, the entity-agents have the ability to spread an alert to other entity-agents using the skill *spreadAlert*. This propagation is made following two conditions: first, the entity-agent using this skill has triggered an alert; second, the alert triggered is because of a mark

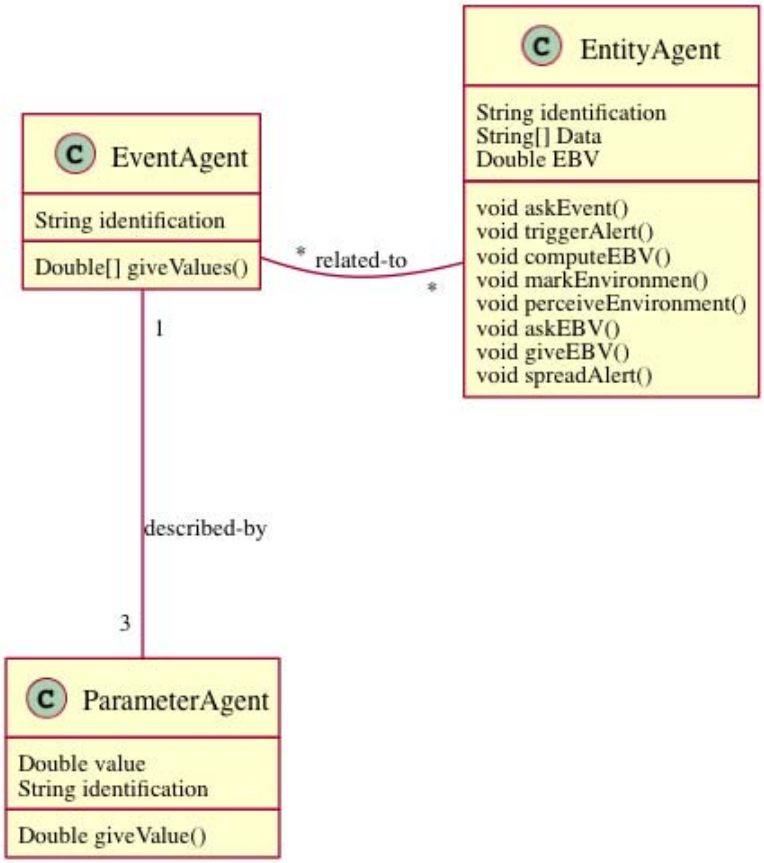

Figure 3.4: UML Model for Alert Triggering (b).

left by another entity-agent. This means that if an agent perceives a mark and it provokes an alert, the agent will think that the creator of the mark is an accomplice and so it perhaps has to trigger an alert as well.

This mechanism is useful to detect abnormal behaviours that are the combination of several normal behaviours. Here an example form the maritime surveillance domain. A cargo ship leaves from a harbour known to be the seat of suspicious activities. This only is not a sufficient reason to trigger an alert. This cargo ship follows an usual route but she stops in open sea for a moment, arguing an engine failure. Again, this is not enough for an alert, but the corresponding entity-agent leaves a mark in the environment to enhance the information of this stop. Finally the ship starts again and pursues its journey with no other events. Then, three hours after the stop of the first ship, a speedboat coming from another suspicious harbours stops in open sea, in the vicinity of the cargo ship's stop. The related entity-agent perceives the mark left and use it to ask the EBV of the cargo-ship. Once the answer received, the entity-agent is able to compute the EBV of the speedboat and it is enough to trigger an alert. In the end, the entity-agent related to the speedboat propagates

Agent	Characteristics	Description
Entity-agent	identification	A unique identification allowing to relate the agent and the real entity.
	data	A bunch of information on the entity.
	EBV	The numerical representation of the suspicious level of the entity behaviour (see function 3.2.
Event-agent	identification	A unique identification to relate the agent and the real event.
Parameter-agent	identification	The identification of the role of the parameter-agent within the event-agent: initial, increase, decrease.
	value	The value of the parameter according to the agent.

Table 3.1: Summary of the agents and their skills for alert triggering.

the alert to the cargo ship and the related agent is able to trigger an alert as well.

These two ships are drug dealers: the cargo transported drugs and left it in a shoal area known by the speedboat. Then, several hours after, the speedboat arrived and retrieved the freight. Independently, these two behaviours are not suspicious and only the combination of the two led to an alert on the two ships. Such behaviours are difficult to identify with the usual anomaly detection techniques (rule engine, statistical,...). However, the multi-agent system proposed in this thesis is able to cope with it.

The tables 3.1 and 3.2 provide a summary of the agents designed for the alert triggering process. This table shows that the agent are design to be as simple as possible while sticking with the real world system.

3.3.2 A Model for Learning

The previous section exposes the model used to design a multi-agent system able to trigger alerts when applied to surveillance domain. In this model, the agents rely on the values of three parameters (*initial*, *increase* and *decrease*) in order to compute the value representing the suspicious level of a behaviour. At the start of the system, these parameters are initialised with arbitrary values. These parameters are the representation of the events' importances according to the operators involved in the surveillance process. However, it is difficult for them to give consistent values to these parameters. Indeed, the consideration of a specific event comes from the experience and the knowledge of the operator. Besides, two operators might not take into account one event in the same way.

Thus, it is important that the system MAS4AT can learn the values of the parameters of events according to the feedbacks (see figure 3.1) of the operators involved in the surveillance.

Agent	Skills	Description
Entity-agent	askEvent	Allows the agent to take into account the events on its real world counterpart.
	triggerAlert	Allows the entity-agent to trigger an alert on itself.
	computeEBV	Allows the entity-agent to compute its Entity Behaviour Value.
	markEvt	Create a new event, local and temporary.
	perceiveEvt	Allow the entity-agent to perceive an event left in its environment.
	askEBV	Ask the EBV of the entity-agents in its neighbourhood.
	giveEBV	Give its EBV to its neighbours.
	spreadAlert	Propagate an alert to other entity-agents.
Event-agent	giveValues	Allows the event-agent to send the value of its parameter to the caller entity-agent.
Parameter-agent	giveValue	Allows the parameter-agent to send its value to the parent event-agent.

Table 3.2: Summary of the agents and their characteristics for alert triggering.

This learning is based on reinforcement learning techniques, since it is well suited for multi-agent systems. Besides, we aim at using feedbacks from the operators in order for the MAS to learn in real time. When an alert is raised by MAS4AT, the operator will be given the ability to send a feedback to the system. We only focus on negative feedbacks, meaning that the operators is only able to correct the multi-agent system. The positive feedbacks are implicit when the operator provides no feedback.

The feedbacks can be of two kinds: On one hand the operator can invalidate an alert triggered by the multi-agent system. This occurs when the alert is wrong because the multi-agent system gives too high values to the parameters of the events involved. On the other hand, the operator can spot an alert that has not been triggered by the system. This is the contrary of the previous case, this occurs when the values given to the events are too low.

Thus, according to the feedbacks and the concerned situations, the parameter-agents are able to tune their own value in order to comply with the feedback. The situations are described by inequalities (see section 3.3.1.3) and an alert is related to the sign of the inequality. Thus, when a feedback is received, the global EBV must be decreased if it is a wrong alert and it must be increased if an alert has not been spotted. This modification of the global EBV is made by the self-tuning of the parameter-agents. Finally, feedback by feedback, the inequalities restrain the search space and lead the agents towards the parameter values to find.

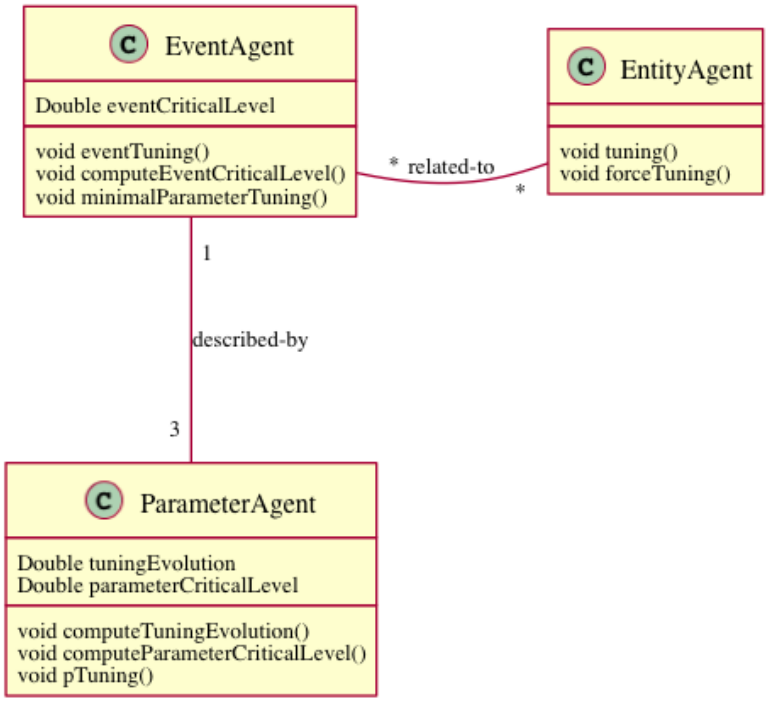

Figure 3.5: UML Model for Learning.

3.3.2.1 Description of the Agents

Based on the three kinds of agents described in section 3.3.1, we propose a model for learning agents. The learning is distributed in the agents: the entity-agents receive the feedback from the operators while the tuning of the parameter value is performed by the parameter-agents and the event-agents. The figure 3.5 shows the characteristics and skills of the agents to support the learning. The global UML model is summarized in section 3.4.

The abilities of the agents are of two kinds: the **-Tuning* are abilities used to ask the tuning of the parameter-agents according to a feedback and the skills *compute-** used in a decision process. This process authorizes the agents to decide if they will tune their parameter or not.

The main idea for the tuning is as follows. When a parameter-agent receives a feedback, it tries to tune itself in order to comply with it. To achieve it, the parameter-agents considers that the global EBV of the entity-agent is superior or inferior to the threshold and it should otherwise. Then, the parameter-agent has to tune itself in order to participate the modification of the global value, in the direction indicated by the agent.

So, this tuning is a simple action that increases or decreases the value of the parameter. This tuning is performed using an Adaptive Value Tracker (AVT) [Lemouzy et al., 2011]. The

AVT is a multi-agent system that is able to track a value according to feedbacks it receives. More specifically, the feedbacks are considered as temporal constraints that must be fulfilled and are represented using agents. Therefore, each constraint agent created must cooperate with the other in order for all of them to be satisfied and cooperatively reach the searched value. Indeed, the constraints expressed by the agents are the values indicated wrong by the feedbacks and each agent is satisfied when it proposes a value outside of the 'forbidden ones'. When all the agents are satisfied, the constraint agents are able to reduce the search space to the value we want to reach.

The AVT starts from an initial value. Each time a feedback is received, the AVT increases or decreases its value to comply with the feedback and according to a delta. The delta is variable: the more the AVT receives the same feedbacks (*i.e.* asking to search the value in the same direction), the more the delta value is great. When a different feedback is received, this value is reduced. Following the feedbacks, the AVT is able to build an implicit representation of the constraints on the search space and reduce it to the value to find.

The decision process is useful because whenever an agent is asked to tune itself, this is due to a global feedback. With its own local knowledge, the agent might decide that it is not able to tune itself, or that its tuning might be useless. The decision for the tuning of the agent is described in section 3.3.2.2.

The skills for the tuning process are basically a demand for tuning of the values. However, following their decision, the agents might decide not to comply and no tuning is done. If so, the skills *forceTuning* and *minimalParameterTuning* are used to force the tuning of at least one parameter. The algorithms using these abilities and performing the learning are described in section 3.3.2.3.

3.3.2.2 Agents Tuning Decision Process

When the operator involved in the surveillance sends a feedback to the system, he sends a global one. Indeed, if he has to look for the parameters that are responsible for the error of the system, it is a lost of time and he will not find this tool efficient nor useful. This is why the feedback is on an entity and not on a parameter.

When a feedback is received by an entity-agent, it concerns a given situation. This situation is the result of several events represented by the event-agents and the parameter-agents. But in this situation, not all the events might be responsible for the global error. More, as an event-agent represents a unique event that can happen on several entities and within several situations, it can not be wise to tune its parameter since it will disturb the other situations: a normal situation could become an abnormal one, and vice versa.

In order to address this problem, we have introduced a decision process in the parameter-agents. Using the skill *computeTuningEvolution* and the characteristic *tuningEvolution*, a parameter-agent is able to decide by itself if it tunes its value or not. This decision is made according to the current feedback and the past tunings performed by the

agent.

In our system, we introduce a mechanism similar to the AVT in the decision process of the parameter-agent. Each time it receives a feedback, the agent compares it with its past actions. The more the agent is asked to search the parameter value in one direction, the more it is confident in its actions. If the agent is asked to search the parameter value in the other direction, it will be reluctant, at first, to change its searching direction. Several requests will be needed before the agent changes its tuning.

More precisely, *tuningEvolution* is a numerical value representing the global direction of tuning of a parameter-agent. This real value is between -1 and 1: The more the value approaches 1, the more the parameter-agent has tuned its value by increasing it; on the contrary, the more the value approaches -1, the more the parameter-agent has tuned its value by decreasing it. The computation of *tuningEvolution* is given by the formula 3.7.

$$\begin{aligned} tuningEvolution = operatorFeedback \times systemDynamic \times agentImportance \\ + (1 - systemDynamic) \times tuningEvolution \end{aligned} \quad (3.7)$$

In this function, the *operatorFeedback* is a numerical value equals to 1, when the global value has to be increased, or -1, when the global value has to be decreased. This variable is only an indicator of the feedback from the operator. It is taken into account in order to compare it with the evolution of the past tunings.

The variable *agentImportance* represents the importance of the parameter in the current situation. It is computed by the ratio of *the weight multiplied by the value of the parameter* to *the cumulated weight of the other parameters, multiplied by their values as well*, involved in the situation (see function 3.8).

$$agentImportance_i^t = \frac{a_i \times w_i^t}{\sum_{j=1}^{n} a_j \times w_j^t} \quad (3.8)$$

This also provides a way to handle the nature of the parameter-agent. Indeed, the *initial parameter* of an agent is less and less determining while the event lasts. On the contrary, the *increase parameter* is more and more decisive. Therefore, when an event appears, its *initial importance* prevails on the other parameters and its modification has a greater impact since it has the greater weight. After several time steps, its the *increasing importance* that is predominant. Finally, once the event has disappeared, the *decreasing importance* should be more taken into account as its weight is growing.

To sum up, when an agent is highly involved in a given situation, its tuning might have a greater impact than if it is little involved. Thus this variable ensures that the more the agent influences the situation, the more it impacts the computation of *tuningEvolution* and so it has a greater change to tune itself.

The variable *systemDynamic* represents the dynamics of the system. Indeed, the more the system represented by the agents is dynamic and evolving, the more the past state of the

system is useless to take into account, and vice versa. *systemDynamic* is a real value between 0 and 1. When equals to 0, this means that the system is not dynamic at all and there is no need to consider the current feedback. On the contrary, when equals to 1, the system is in constant evolution and the past states are not useful for the parameter tuning because, as the system is in constant change, they are not true any more.

There are two benefits in this function. First it ensures that the agent stays constant in its tuning, meaning that when searching the parameter value, it will not change the searching direction for each contrary feedbacks and it waits for several requests asking to do so. This is also a way to address wrong feedbacks.

Second, it ensures that the agent will not disturb past situations. Indeed, if an agent is asked to increase its parameter value to comply with a feedback, it is because the current value is not right. The decision process prevents the agent to loop back to this value on the next feedback and thus to forget the past action. These past actions are taken into account during a certain time, and the most former are less considered than the newer, thanks to the *systemDynamic* variable.

This function is used by the parameter-agents to decide if they perform a tuning following a feedback. The algorithms in section 3.3.2.3 describe the whole learning process.

3.3.2.3 Algorithms for Learning

During the running of the surveillance system, the operator is able to send feedbacks to the multi-agent system. These feedbacks are used to signify a missing alert or an erroneous one. When an entity-agent receives a feedback, it performs a tuning according to the algorithm 3.1.

Algorithm 3.1: Entity-agent tuning algorithm.

foreach *event-agent used by the considered entity-agent* **do**
  Call eventTuning()
end
EBV ← computeEBV()
if *EBV == previousEBV* **then**
  Call ForceTuning()
end

The design of the feedback is such that it concerns an entity at a given time. Thus, the entity-agent is able to perceive the feedback and to relate it to the current situation. As the situation is a combination of several events, the entity-agent then broadcast the feedback to the concerned event-agents. These agents have then to perform a tuning following the algorithm 3.2.

After the tuning of the event-agents, the entity-agent checks if there is a difference between the current Entity Behaviour Value (EBV) –computed after the tuning– and the

previous one –computed before the tuning. If there is no difference, the agent assumes that no tuning has been performed and asked for the tuning of the less constraint event-agent (see algorithm 3.4).

This is one aspect of the AMAS Theory. Indeed, each agent has the objective to reach a satisfaction that helps the system, without degrading its own satisfaction. Therefore, the agents are able to cooperate and identify the less constraint between themselves. Here, the cooperation allow to improve the whole system without degrading too much the agents and their beliefs.

Forcing the tuning of at least one event-agent allows the system to react each time there is a feedback. This has two objectives: first, it prevents the agents to do nothing despite the feedbacks and second, it if the agents are not tuning their value, the operator might find it useless to send feedbacks to a system that does not take it into account.

Algorithm 3.2: Event-agent tuning algorithm.

```
foreach parameter-agent of the considered event-agent do
  | Call parameterTuning()
end
```

Similarly to the entity-agent, when an event-agent receive a feedback, it broadcasts it to its parameter-agents, which will use the algorithm 3.3. When receiving a feedback, the parameter-agent computes its own *tuningEvolution* (see 3.3.2.2). If the new value of *tuningEvolution* is compliant with the feedback, the parameter-agent performs a tuning of its value.

The compliance of *tuningEvolution* with the feedback is a simple check of the signs of the two variables. If it is the same, it means that the agent agrees to tune its value in the direction asked by the feedback. If different, it means that the agent prefers to tune its parameter in the reverse direction than the feedback, in which case, the agent does not perform the tuning in order to not disturb the system. This mechanism follows another aspect of the AMAS Theory. Indeed, the parameter-agents are able to use the *tuningEvolution* variable to check if they are in conflict with other agents: themselves in past situations. In case of a detected conflict, they decide to not perform the tuning in order to keep the satisfaction they have. More than with other agents, they are cooperative with themselves.

Algorithm 3.3: Parameter-agent tuning algorithm.

```
tuningEvolution ← computeTuningEvolution()
if compliance(tuningEvolution, operatorFeedback) then
  | Call pTuning()
end
```

In the case of there is no tuning performed by none of all the parameter-agents involved in a situation, the entity-agent asks the concerned event-agents for a forced tuning. This tuning is made according algorithm 3.4. When none of the agents decide to tune themselves, it is because they have decided that the required tuning is not compliant with their

knowledge, meaning their usual actions. Thus, it is irrelevant to ask a forced tuning of all the agents, because it would imply that all the agents must go against their belief, rendering this belief useless, and so for the decision process.

Algorithm 3.4: Entity-agent force tuning algorithm.

```
foreach event-agent of the considered ship-agent do
    Call computeEventCriticalLevel()
end
Determine less constraint event-agents
foreach less constraint event-agents do
    Call parameterMinimalTuning()
end
```

Instead, each event-agent computes it critical level. The critical level of an agent is a numerical value representing the constraints on an agent. The higher is this value, the more constraint is the agent. The function 3.9 is used to compute this value.

$$a_{CL} = |S_e| \text{ avec } S_a \text{ l'ensemble des situations tel que } a \in S \tag{3.9}$$

This function is a simple count of all the situation where the event a is involved. Indeed, an event-agent used by several entity-agents to compute their EBV –*i.e.* is used in several situations– is more constraint than an agent used by only one entity-agent. Once the less constraint agents are identified, they are designated to perform a tuning no matter their decision. In order to achieve it, they force the tuning of at least one of their parameter according to the algorithm 3.5.

Algorithm 3.5: Parameter-agent force tuning algorithm.

```
foreach parameter-agent do
    Call computeParameterCriticalLevel()
end
Determine less constraint parameter-agents
foreach less constraint parameter-agents do
    Call pTuning()
end
```

This algorithm is similar to the algorithm used to force the tuning of an event: the parameters of the concerned agent compute their critical level and the less constraint is designated to tune its value no matter its *tuningEvolution* value. The critical level of a parameter has the same nature as the critical level of an event: a numerical value expressing the constraint on the agent. However, its computation is different, as illustrated by the function 3.10.

$$p_{CL} = gapTo0^2 - gapToBound^2,\ \{gapTo0, gapToBound\} \in [0,1] \tag{3.10}$$

In this function, $gapTo0$ represents the distance between *tuningEvolution* and 0. The more this value is small, the more the agent is close to change its tuning direction, either because

it has received a lot of feedbacks asking to do so, or because it has not yet perform enough tuning in a given direction. The variable *gapToBound* is indicates the distance between the current value of the parameter and the boundary of the search space, according to the feedback –the inferior boundary if asked to decrease the value and vice versa. The more the value of the parameter is close to the boundary, the more the agent is constraint in its tuning because it reduces the residual search space. These two values are normalized between 0 and 1.

The result of this function is a numerical value between -1 and 1 expressing the constraint on the concerned parameter-agents. The smaller is this value, the lesser the agent is constraint. Once the less constraint parameter-agents are identified, the tuning of the parameter value is performed by the related Adaptive Value Tracker and according to the feedback.

The algorithms presented in this section are based on a reinforcement learning approach in order to tune the weight, the importance, of the events happening on the entities in a monitored area. The operators involved in the surveillance process are able to send a feedback on a situation on an entity. Following this feedback, the concerned agents modify their value in order to comply with it. This is a learning process that aims to reaching accurate values, meaning values that do not provoke feedbacks from the operators.

3.4 Conclusion

In this chapter, the generic model *Multi-Agent System for Alert Triggering (MAS4AT)* designed to address the alert triggering in a surveillance system has been detailed.

By analysing the entities and the information involved in the surveillance process, three kinds of agents have been designed –see figure 3.6– in order to achieve two roles.

First, the main purpose of MAS4AT is the abnormal behaviour detection, thus the system is able to trigger alerts. We adopt a numerical approach to model the behaviours of the entities in the monitored area. For each entity, a entity-agent is created and able to compute an Entity Behaviour Value (EBV) indicating the suspicion level on the entity. This EBV is computed according to the event happening on the entity, represented by its *initial* importance, its *increasing* importance while it lasts and its *decreasing* one once it has disappeared.

This numerical approach allows to give a graphical representation, using a mathematical curve, of the behaviour of an entity. It is then possible to define a threshold which provokes the triggering of alerts when exceeded. This threshold is defined by the operators.

This method **avoids to define beforehand all the authorized behaviours** for the entities, nor the forbidden ones. Indeed, it is not possible to enumerate all these behaviours in an efficient way. Instead, we use the numerical approach to cumulate the suspicion related to entities and to trigger alerts when this suspicion level is too high.

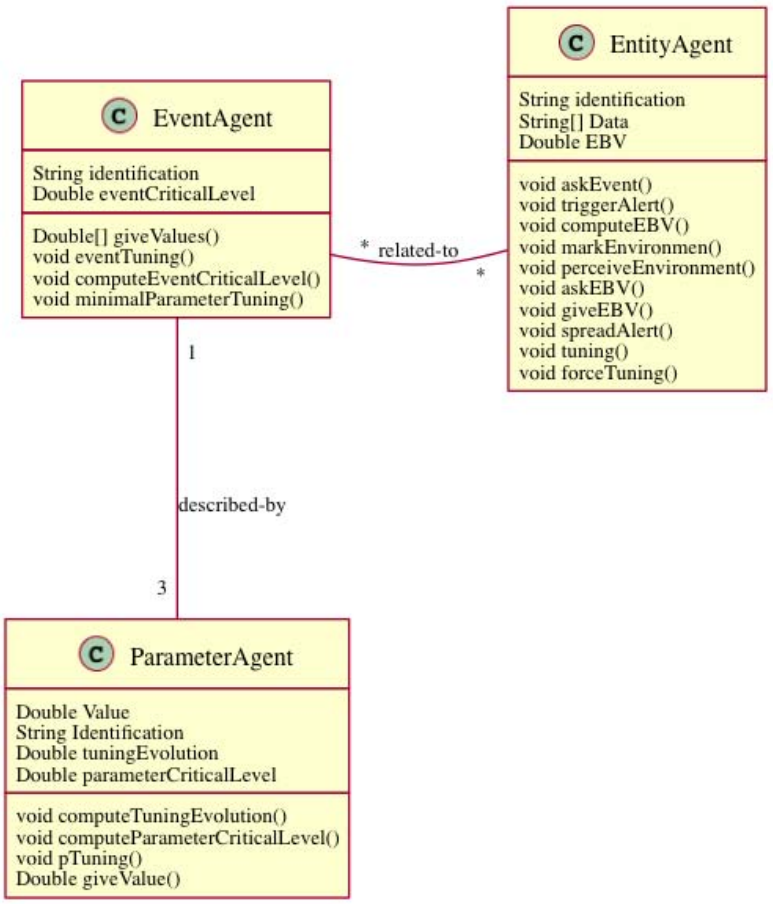

Figure 3.6: Global UML Agent Model.

The collective behaviours of the entity-agents also allows **to take into account abnormal behaviours that are the combination of several normal behaviours**. MAS4AT is able to construct an implicit representation of all the supervised entities behaviours using numerical functions. This ensures a **simple processing** by the system and a low resource cost.

Besides, **MAS4AT** is able to **perform a learning** in order to provide more accurate abnormal behaviour detections. Using feedbacks from operators and their abilities to tune their values, the agents of MAS4AT are able to do a reinforcement-based learning, tuning the values of the importance of the events that could happen on entities. The mechanisms used are inspired by the AMAS Theory in the way that they have local beliefs and they possess cooperation skills. These aptitudes allow the agents to comply with two situations related to the non cooperative situation in the theory. The first one is the uselessness of the agent when they do not perform tuning, the second one is the conflict of the agent with itself. When one of these situations is detected, the agents are able to make actions in order to solve them

(force the tuning and decide to not tune itself).

Finally, MAS4AT exhibits several qualities. On one hand, it is an open system and the monitored entities are taken into account as they appear, the same goes for the events and each new one is represented on the fly as an event-agent. In a row, MAS4AT is able to adapt itself to the monitored area by creating entity-agents and event-agents related to what is observed, when it is observed. On the other hand, the system MAS4AT consists of simple agents that ensuring a low cost in term of material resources, such as memory, and the representation of a situation using a simple mathematical function ensures a relatively low computation time.

From a theoretical point of view, **we thought this model generic enough to be used in several surveillance contexts**. By generic, we mean that the this model could be used in other domains that maritime surveillance. The only prerequisite is that the representation of a situation must be made as a combination of events. Besides, each event should be representable as a numerical value in order to compute a numerical representation of the entity behaviours. We think such domains are aerial surveillance, traffic controls, network intrusion detection,... Indeed, in these kinds of systems, a situation might be represented as a sequence of events and a suspicious level could be associated to each situation. With the suspicious level computed as a mathematical function, the system MAS4AT is able to propose alerts and learn the parameter values according to feedbacks (from an operator or from the system itself). In chapter 4, we investigate the use of this model in a maritime surveillance system.

4 A MAS4AT Application: A Maritime Surveillance System

« Just because you believe it doesn't make it so. »

Orson S. Card

Contents

The chapter in English starts page 79.

Résumé général du chapitre

Dans les chapitre précédents, nous avons identifier les manquements des systèmes de surveillance actuels, à savoir le manque de généricité et de fonctionnement en temps réel avec un faible coÃ»t en ressources. Dans le chapitre 3, nous proposons un modèle agent, MAS4AT, pour adresser ces aspects et s'attaquer à deux problèmes associés à la surveillance : l'identification de situations anormales et l'apprentissage des valeurs des évènements constituant une situation.

Le système MAS4AT se veut générique et adaptable dans de nombreux système de surveillance, pour autant que la représentation d'une situation puisse s'exprimer sous la forme d'une fonction mathématique et que des mécanismes de retours soient disponibles pour l'apprentissage par renforcement. La projet I2C associé à cette thèse offre un cadre d'application pour tester le système en condition réelle.

Le projet I2C est un projet européen qui vise à fournir une solution tout en un dans le domaine de la surveillance maritime. Avec un panel de capteurs et de senseurs déployés dans toutes les zones à surveiller et des bases de données contenant toutes les informations disponibles sur les navires, I2C est capable de fournir une représentation en temps réel des zones maritimes concernées. Ensuite, un moteur de règles contenant la législation maritime effectue un tri sur les évènements qui arrivent aux navires et un module d'analyse comportementale permet d'isoler les situations qui sont suspectes et donc dignes d'intérêt. Les opérateurs de surveillance maritime peuvent ensuite prendre les mesures nécessaires pour répondre à la menace ou corriger le système en lui indiquant une erreur.

Au sein de I2C, le système MAS4AT est en charge de l'évaluation des situations et de la levée d'alertes. Pour des raisons de spécifications et de maintenance, quelques adaptations ont été faites : une division en deux systèmes multi-agents et des changements d'appellation. OpMAS concerne la représentation des navires grâce à des agent-navires tandis que PaMAS représente les évènements des situations grâce à des agent-paramètres et des agent-evenements. Ce sont les seuls changements effectifs par rapport au modèle initial. Le fonctionnement et les comportements des agents reste identique à celui présenté dans le chapitre précédent.

L'intégration dans I2C étant en cours de réalisation, des données réelles ne sont pas encore disponibles. Pour pallier à ce manque et permettre de tester MAS4AT, nous avons aussi conçu et développé une plateforme de simulation, SIM4AT. Cette plateforme est capable de générer des scénarios aléatoires qui seront suivis par des entités. Un simulateur de moteur de règle est capable de lire les scénarios et d'envoyer les évènements aux agents concernés. De plus, un autre composant est capable de simuler la connaissance des opérateurs et d'évaluer les alertes envoyées par les agents. Ce composant est ensuite capable d'envoyer des retours pour signifier aux agent-navires que l'alerte est erronée ou manquante, provoquant l'ajustement des agents.

Dans ce chapitre, nous présentons donc le projet I2C et le système qu'il vise à développer, ainsi que le rôle de MAS4AT en son sein. Nous y détaillons aussi les adaptations mineures nécessaires

pour l'intégration. Enfin, nous présentons la plateforme de simulation SIM4AT capable de modéliser et remplacer les différents composants de I2C interagissant avec les agents, afin de permettre des tests sans attendre l'intégration finale.

4.1 Introduction

The Multi-Agent System for Alert Triggering (MAS4AT) model presented in chapter 3 has been designed to be generic and applicable to any surveillance systems, as long as it is possible to identify events, entities and to use a numerical representation of behaviours. In order to validate the model, we have chosen an application in a specific range of surveillance: the maritime domain.

In the recent years, the maritime activity has grown rapidly and open sea is now the location of several illegal activities, from illegal fishing to drugs and arms dealing. However, maritime surveillance does not involve only the identification of threats, but also the monitoring for rescue and assistance.

Whatever the reason, countries with high coastal areas are in demand of efficient maritime surveillance systems. Thus, our generic system MAS4AT has been applied into the European Project I2C[1][Morel and Claisse, 2010a; Morel and Broussolle, 2011]. This chapter details the integration of MAS4AT in I2C and is organized as follows.

Section 4.2 presents the project I2C and its characteristics. I2C is an ambitious project aiming at the aggregation of several tools useful in the maritime surveillance. A description of the role of the multi-agent system is given.

Section 4.3 provides the description of the implementation of MAS4AT in I2C. MAS4AT for I2C is slightly different from the generic model, due to some constraints of the project. This section describes and explains the variation between the model presented in chapter 3 and its application in this context.

Finally, section 4.4 describes the platform SIMulation for Alert Triggering (SIM4AT). This tool has been designed to simulate various components of the project I2C in order to test MAS4AT. The section provides a model of SIM4AT as well as guidelines for its use.

4.2 Maritime Surveillance: The Project I2C

The project I2C aims at integrating data processing as well as exploitation capabilities in order to detect all types of ships –medium and large– and to identify the threats at sea in order to make a quick report toward the authorities and plan for efficient and relevant actions.

[1]Integrated system for interoperable sensors & Information sources for common abnormal vessel behaviour detection & Collaborative identification of threat

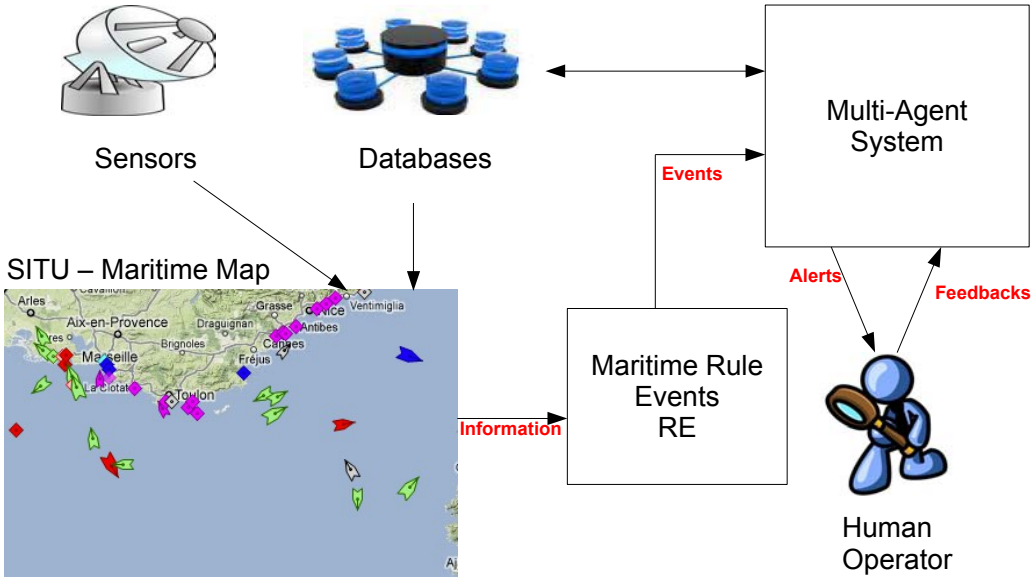

Figure 4.1: Architecture of I2C.

The first objective of the project is the deployment of sensors and radars on several platforms: mobile ones like planes or helicopters for video cameras or immobile ones for High Frequency Surface Wave Radars. All these captors send data to a centralized architecture of I2C. Thus, the first *I* of I2C stands for *Integrated system for interoperable sensors*.

Second, I2C retrieves all information coming from different sources in order to aggregate data on the different ships in the monitored area. This data is available on a graphical interface to be used by the users. Besides, using this data and the information, the system is able to detect abnormal behaviours of ships. That is the *Information sources for common abnormal vessel behaviour detection* part of I2C.

Finally, when abnormal behaviours are detected and alerts triggered, the system aims at proposing a way to identify and understand the underlying threat. This is the *Collaborative identification of threat*.

In a row, the *I2C project proposes a new generation of innovative sea border surveillance end to end systems integrating key existing and in development capacities to track all vessel movements and activities to early identify and report on EUROSUR threats (clandestine immigration, law enforcement, illegal fishing, terrorism...) associated with detected abnormal vessel behaviours*.

4.2.1 Architecture of I2C

Figure 4.1 presents the global architecture of the system I2C. One of the main features of I2C is to propose data and information coming from different sources. Indeed, several sensors are deployed on coastal platforms as well as on mobile devices: Long range High-Frequency (HF) radars, High-Frequency Surface-Wave (HFSW) radars, video

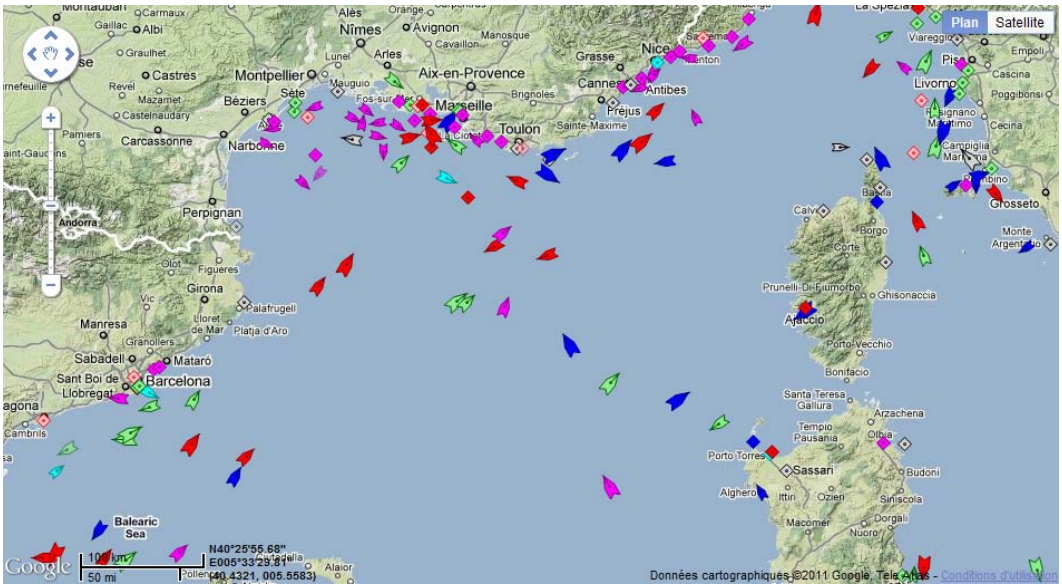

Figure 4.2: I2C Common Operational Traffic Picture.

cameras, satellite picture,...These devices come with integrated automatic tracking and recognition tools. In addition, information on the different ships can come from other sources like captaincy manifests, web pictures, Automatic Identification System (AIS)[2] data, ...Therefore, the system I2C also integrates this data into a knowledge database to be used by the users and the various components of the system.

Once all data is available, the system proposes a visualisation of the monitored area using a cartographic representation, it is called the *Common Operational Traffic Picture (COTP)*. Figure 4.2 illustrates this representation. Using a touch screen interface, the users, the operators involved in the surveillance process are able to display the information on a desired vessel. They are also able to designate an interest on a given vessel and focus on it in order to analyse its behaviour. Thus, the main objective of this component is the fusion of the information coming from the various sources and their display on a graphical interface for the users. There are two kinds of user: the operators monitoring the system and triggering alerts when a ship has a suspicious behaviours; the experts that receive the alerts and are able to identify the related threat or danger, they give a semantic definition of the alert.

Using the COTP, the users are able to identify a ship and see the available information on it: AIS identification, name, color, flag, speed, orientation, ...These information are sent by the system to the component Behaviour Analysis (BEAN). BEAN is composed of a rule engine and a multi-agent system as shown on figure 4.3.

The rule engine is the entry point of the data. Its purpose is to detect specific anomalies and events happening on the vessels in the monitored area. The rules are designed to detect events that are known to be relevant in the abnormal behaviours cases: unusual events like

[2]AIS is a radio positioning and vessel identification system in maritime traffic.

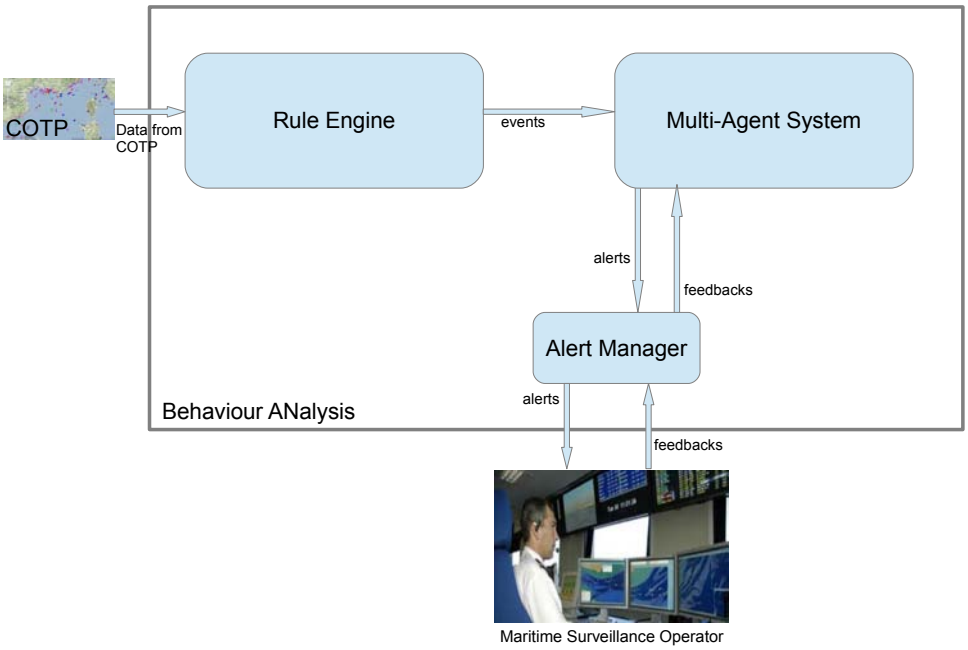

Figure 4.3: Architecture of the BEAN Component.

a stop in open sea for engine failure but also events that are against the maritime legislation, implanted in the rule engine. Rules 4.1 and 4.2 illustrate rules of the rule engine.

$$\textbf{if } speed > 15 \textbf{ and } area = "Harbour" \textbf{ then } Detect(speedAnomaly) \tag{4.1}$$

$$\textbf{if } shipType = "cargo" \textbf{ and } speed = 0 \textbf{ then } Detect(cargoStopInOpenSea) \tag{4.2}$$

The first rule allows the detection of a ship with a speed above the speed limit in an harbour area, the second one allows to detect the stop of a cargo ship in open sea without apparent reasons.

The detected events are then sent to the multi-agent part of the BEAN component. The MAS role is to combine the events and the anomalies in order to compute a suspicious level for each monitored vessel –see section 4.2.2. The rule engine and the multi-agent system have complementary roles. Indeed, the rule engine performs a first pruning of the data of COTP in order to identify relevant events from irrelevant ones. Then, the multi-agent system use these events and combine them in order to compute the suspicious level of a ship.

The aim of the whole Behaviour Analysis is to receive information and analyse all the events that are happening on the ships in the monitored area. This analysis allows BEAN

to associate a suspicious level to each ship and the component is thus able to detect the anomalies in the ship behaviours. Finally alerts are sent to the operators when it is found to be necessary.

Ultimately, the operators can compare the alerts triggered by BEAN with what they analyse according to their knowledge and experience. If they agree with the automatic alert, they transmit it to the concerned authorities in order to identify and answer the situation (thread to tackle, need for assistance,...). On the contrary, if the operators do not agree with the alert from BEAN, they can send a feedback aiming at the correction of the system. A feedback is also sent when the operators spot an alert that has not been triggered by BEAN.

I2C is a European Project aiming at the surveillance of maritime areas to identify and answer threats and issues. This project integrates acquisition tools as well as anomaly detection mechanisms. In the BEAN component, we now focus on the role of the multi-agent system and how the different parts of MAS4AT can be used for it.

4.2.2 The Role of the Multi-Agent System

In the component Behaviour Analysis (BEAN), the rule engine is combined to the self-adaptive multi-agent system. Indeed, the events and anomalies detected by the rule engine are sent to the MAS in order to be processed by the agents. Then, the MAS decides to send an alert to the operators, if an abnormal behaviour is detected. The role of the multi-agent system is triple.

First it **evaluates and represents the behaviours of the ships**. Indeed, the MAS should integrate agents to represent each ship in the monitored area. Then, according to the events sent by the rule engine, the agents should be able to represent the behaviours of the ship in the current situation. This behaviour is defined by a series of events that happened on the ship. In MAS4AT, there are entity-agents that can represent the ships. They are also able to compute a value, called *Entity Behaviour Value (EBV)*, that is a numerical representation of a behaviour at a given time and it is based on a combination of events. As this value is given by a function –see 3.2– the agents are also able to give a graphical representation of the behaviour value over time. As the more the EBV is high, the more the entity behaviour is suspicious, this curve represents the evolution of the suspicion level of the entity during time and according to the events from the rule engine.

Second, the agents representing the ships should be able to decide to **trigger an alert** towards the operators when its behaviours is considered too suspicious. In MAS4AT, the entity-agents are able to compare their EBV to a fixed threshold. This threshold is a representation of the need for an alert when a behaviour is too suspicious. Indeed, when the EBV is superior to the threshold, the corresponding entity-agent is able to trigger an alert onto the related entity. More specifically, each time an event is received by the agent, its values are used to combine it with the previous events. When the combination reach a critical state, an alert is triggered.

Third, the BEAN component has to **learn the values of the different possible events**. These values are a numerical representation of the importance of the event. It has been exposed in chapters 2 and 3 that it is not realistic to ask the human operators to give a value to each events because it is their own experience and knowledge that build it. MAS4AT integrates agents to represent the events and their values as well as learning mechanisms to learn the event values. The learning is based on the feedbacks from the operators correcting a wrong or a missing alert.

The multi-agent system in the BEAN component of I2C should be able to represent the entities and the events happening on them. These events have to be valued and combined by the agents and according to the data from the rule engine in order to raise alerts when the entity behaviour is considered too suspicious by the agents. The model of MAS4AT described in chapter 3 is well suited to perform these tasks.

4.3 Adaptation of MAS4AT to I2C Project

The model MAS4AT described in chapter 3 has been designed to be generic and adaptive to any surveillance systems as long as prerequisites are fulfilled: entity representation and numerical events representation. In this section, we describe the adaptation of MAS4AT for the BEhaviour ANalysis (BEAN) component of the project I2C. In the end, we explain how the changes made are not impacting the generic model or its running.

In I2C, there is a distinction between the supervised ships, and what can happen on them. The operators are able to intervene on the representation of the ships, in order to add or modify an information, to trigger or remove an alert, an so on. On the contrary, the rules and the subsequent events are internal to the component Behaviour Analysis and can only be modified by experts. In order to cope with this need, the agents in MAS4AT have been separated into two different multi-agent systems: the *Operative Multi-Agent System (OpMAS)* and the *Parameter Multi-Agent System (PaMAS)*.

OpMAS is composed by the entity-agents and has the role of the ship representation and the computation of their behaviours while PaMAS represents the events and is able to learn their values according to the feedbacks from the operator. The new architecture of MAS4AT is shown in figure 4.4. This separation does not change the communication nor the relationships between the agents. It only allows a better optimisation of each part.

4.3.1 The Operative Multi-Agent System (OpMAS)

OpMAS is the multi-agent system within MAS4AT for I2C that is in charge of the representation of the ships in the monitored area and the alert triggering. Thus, OpMAS is composed of entity-agents which are here called ship-agents to be closer to the supervised entities. The appellation Entity Behaviour Value has been modified alike in Ship Behaviour Value. The changes of names are the only difference between the agents in OpMAS and the

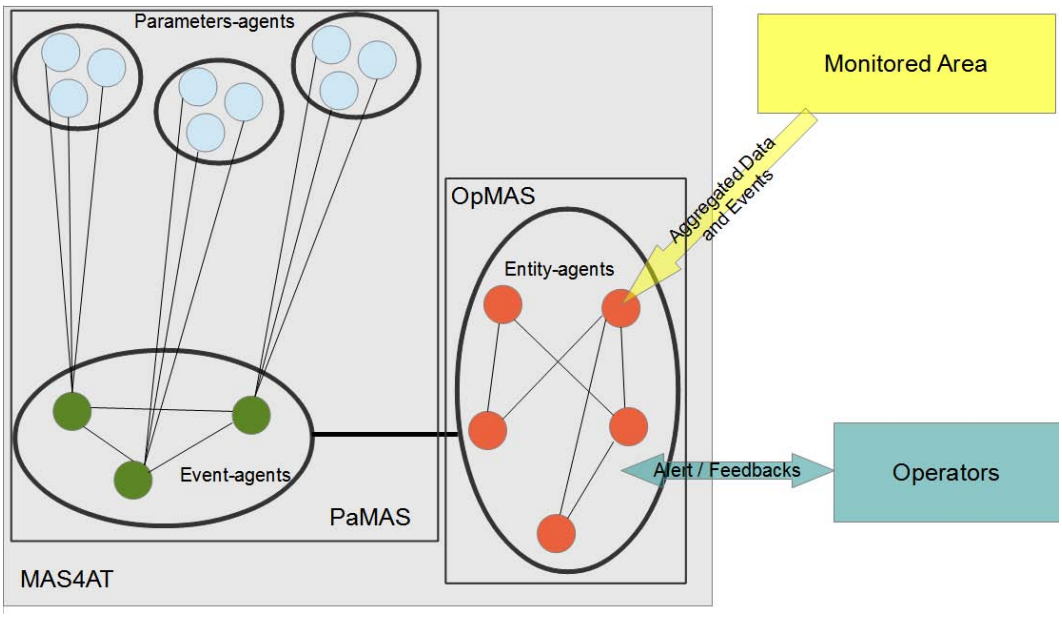

Figure 4.4: Architecture of MAS4AT in the system I2C.

agents in the generic MAS4AT.

OpMAS also integrates a software component, called the *Agent manager*, whose role is to act as an interface between the operators and the system MAS4AT. On the one hand, the agent manager receives the alerts from a given ship-agent and sends it to the operator of I2C. On the other hand, the agent manager receives the feedbacks from the operator and the events from the rule engine and transmits them to the concerned ship-agent.

The operation of the ship-agents is the same than the entity-agents. Each ship agent is able to compute a numerical value, its *Ship Behaviour Value* according to the function 4.3.

$$SBV = \sum_{i=1}^{n} (Init(e_i) \times nb(Init, e_i) + (Incr(e_i) \times nb(Incr, e_i) - (Decr(e_i) \times nb(Decr, e_i) \quad (4.3)$$

For each event received for the ship-agent, the SBV is computed according the weight of the three parameters Init, Incr and Decr and the values given by the related event-agents e_i.

In a row, the Ship Behaviour Value represents the behaviour of a ship as the combination of several events of various importance, given by the value of the parameters. The higher is the result, the more suspicious is the behaviour of the ship. The figure 4.5 shows an example of the combination of different events happening on a given ship-agent.

Finally, for each time step, the ship-agents update the weights of the events involved in the computation of their SBV and trigger alerts when it is necessary. The time step unit in I2C is the duration of the interval between two updates of the Common Operational Traffic Picture.

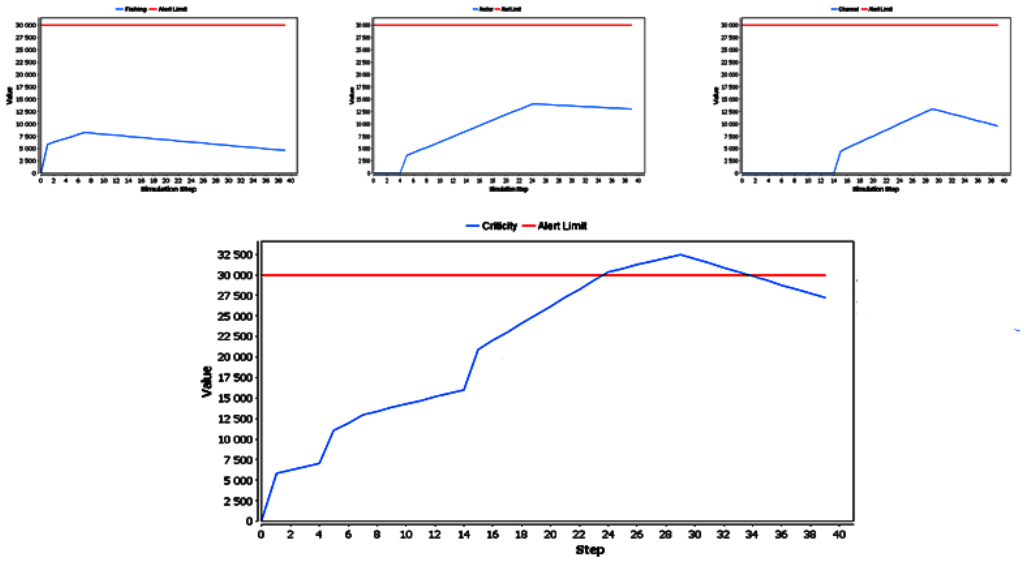

Figure 4.5: Event Value Combination.

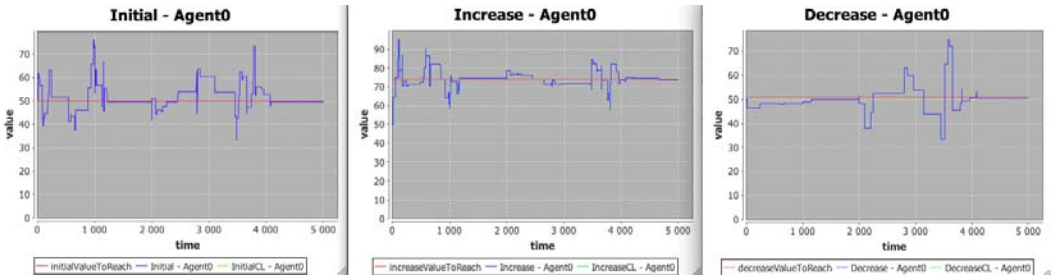

Figure 4.6: MAS4AT Learning Example for I2C.

4.3.2 The Parameter Multi-Agent System (PaMAS)

PaMAS is the multi-agent system within MAS4AT for I2C which is in charge of the representation of the events. According to the model in section 3.3.2, PaMAS contains event-agents composed of three parameter-agents each.

Their role is to value the events that are happening on the ship and send these values to the ship-agents when asked to do so. They are also able to perform a self-tuning in order to learn the values of the parameters according to the operator feedbacks. The figure 4.6 illustrates the result of the learning process.

The only difference between the generic model and I2C is that the events represented by the event-agents comes from the rule engine which performs a preprocessing of what happened in the monitored environment. It means that the rule engine can send anomalies to MAS4AT. These anomalies are taken into account as events by the ship-agents and are thus represented as event-agents. This can lead to events that are automatically trigger an

alert, for example on a ship that is blacklisted by the authorities because known to practice illegal fishing.

Ultimately, each ship-agent is able to give an accurate behaviour value and to represent each situation by an inequality compared with an alert threshold. Except for the refactoring of the agent names and the separation into two different multi-agent systems, OpMAS and PaMAS, the system MAS4AT used in I2C corresponds to the generic model presented in chapter 3.

4.4 A Simulation Platform for MAS4AT

Within the project I2C, there are several partners developing several tools and components for the final system. The design, the deployment and the installation of the sensors and radars take time, as well as for the data integration and fusion techniques and the the behaviour analysis engine. When these components have to be tested, some are dependent of the others: data fusion needs the tracks captured by the sensors and the detection engine needs the data coming from the aggregation.

That is where time is a constraint: all the components are designed by the different partners at the same time and the testing has to be done independently because none of the needed other components are set up and operational.

Therefore, we have proposed a simulation platform called *Simulators for Alert Triggering* (SIM4AT) that aims at simulating several components of the project I2C as information sources. This tool is to be used to evaluate MAS4AT. This tool has been designed in regard to the behaviour analysis engine, and more specifically the anomaly detection within it. This tool is made to be generic and reusable in other contexts.

The platform SIM4AT has been designed to be easily pluggable with detection systems. In the same way, we have designed it to be easy to use and exploit. The user provides the number of wanted entities, the number of possible types of the entities as well as the number of possible events that can happen on an entity. Then, the platform processes the data and provides one scenario for each entity. Finally, these scenarios are used to test the detection system while visualisation allows to observe the behaviour of MAS4AT.

4.4.1 Principles of SIM4AT

SIM4AT is a simulation platform aiming at the replacement of both the rule engine and the operator works in the project I2C. For this, it also requires to simulate the data sources for the rule engine. SIMAT is designed to test anomaly detection of a surveillance system, so the definitions of the different components and structures are kept as simple as needed in regard to this.

Thus first step is to design the information that feeds the rule engine in order to

Time	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Sc1	–	–	E1	–	eE1	–	–	–	–	–	–	–	–	–
Sc2	E1,E2	–	eE1	–	–	eE2	–	E3	–	–	eE3	–	E4	eE4
Sc3	E1,E2	E3	–	eE2	–	–	E2	–	eE1,eE3	E4	–	eE2	–	eE4

Table 4.1: Scenario Examples (a).

implement it in SIM4AT. The platform aims at simulating the behaviours of entities in order to be analysed by MAS4AT. To achieve it, we have modelled scenarios to feed the rule engine simulator with. A scenario is a representation of an entity behaviour: it is a series of events occurring from time to time. This definition provides scenarios with simple structure but able to represent complex features. Moreover, it does not take into account the meaning of the events, neither the significance of a whole scenario.

Indeed, according to the chapter 3, MAS4AT only needs the identification of the events and of the entities in order to work. The significance behind the identifications are left for the operators when they analyse a situation. Therefore, the structure of our scenarios only uses these two parameters. The table 4.1 shows three different scenarios, one by line, *Sc01*, *Sc02* and *Sc03*.

The first scenario, *Sc01* is a very simple one where only one event, *E1*, happens at time step 03, before it disappears, *eE1* with *e* for *end*, at time step 05. During the rest of the time, nothing else happens.

The second scenario, *Sc02*, is a little more complex because several events takes place into it, and sometimes at the same time, like the events *E1* and *E2* at the first step. Finally, the third scenario sees several appearances and disappearances of several events overlapping. For example, the event *E1* appears at time step 01 and disappears at time step 09 while the event *E2* appears at step 07 and disappears at step 12.

In SIM4AT, the events and the scenarios are randomly created by the appropriate component, the scenarios generator. The duration of one time step is left to the user appreciation according to the domain. The events are also randomly generated, according to their definition in section 3.3.1.

It can be useful to feed the system with known scenarios in order to test specific cases in a specific context, or to serve as proof of concept. Regarding this need, SIM4AT can take user-generated scenarios using the same structure. The only need is for the user to identify the different events that are in its scenario.

For example, the table 4.2 shows two scenarios that are an adaptation of direct observation in maritime surveillance. In these scenarios:

- The event *E1* represents a *stop in open sea*.
- The event *E2* represents an *engine failure*.

Time	01	02	03	04	05	06	07	08	09	10
Ship1	–	–	E1,2	–	–	eE2	eE1	–	–	–
Ship2	E1,3	–	eE3	–	–	–	E4	–	–	–

Table 4.2: Scenario Examples (b).

- The event *E3* represents a *proximity with an unknown ship*.
- The event *E4* represents the *disappearance of the radar track of a ship*.
- The interval between two time step is of 30 minutes.

Concerning the ship *ship1*, the scenario indicates a stop in open sea, apparently for engine failure. When the engine failure is fixed, the ship can start again and continue its journey. This also shows the data fusion of the real system: the radar coverage indicates that the ship has stopped and the information from the captaincy indicates the engine failure.

The second scenario is a little more complex, involving at least two ships: the one that run the scenario and an unknown one illustrated by the event *E3* at time 01. Indeed, the ship *ship2* stops in open sea and the radar coverage indicates a proximity with an unknown ship. After an hour, the unknown ship starts again and rolls away from *ship2*, which is represented by the end of the event *E3* at time 03. Finally, two hours later, at time 07, the ship *ship2* disappears from the radar. This scenario represents the scuttling of a ship after her occupants have been taken hostage.

Once the scenarios are created, either by a random generation or given by one user, they are processed through the rule engine simulator. Its running is simple: it reads the scenarios step by step and sends every encountered events and end of events to the detection system, MAS4AT in our case.

MAS4AT then processes the eventually received events and updates its representation of each entity. It means that for each entity, the corresponding function allowing to compute the Entity Behaviour Value (EBV) is updated –see chapter 3. When an EBV exceeds the alert threshold, MAS4AT then sends an alert to the operator simulator.

Parallel to the rule engine and MAS4AT, the operator simulator is also able to read the scenarios of each entity and to compute their EBV. The operator simulator knows the real values of the events created and so the EBV it computes is also the "real one", meaning the one towards which the parameter-agents should converge to be correct.

Finally the operator simulator can compare them with what is coming from MAS4AT. If an alert is sent from MAS4AT, the operator simulator takes a look into the EBV of the related entity. If this value does not indicate an alert, a feedback is sent back to MAS4AT. On the contrary, when no alert is sent and the value computed by the operator simulator indicates one, a feedback is sent to MAS4AT.

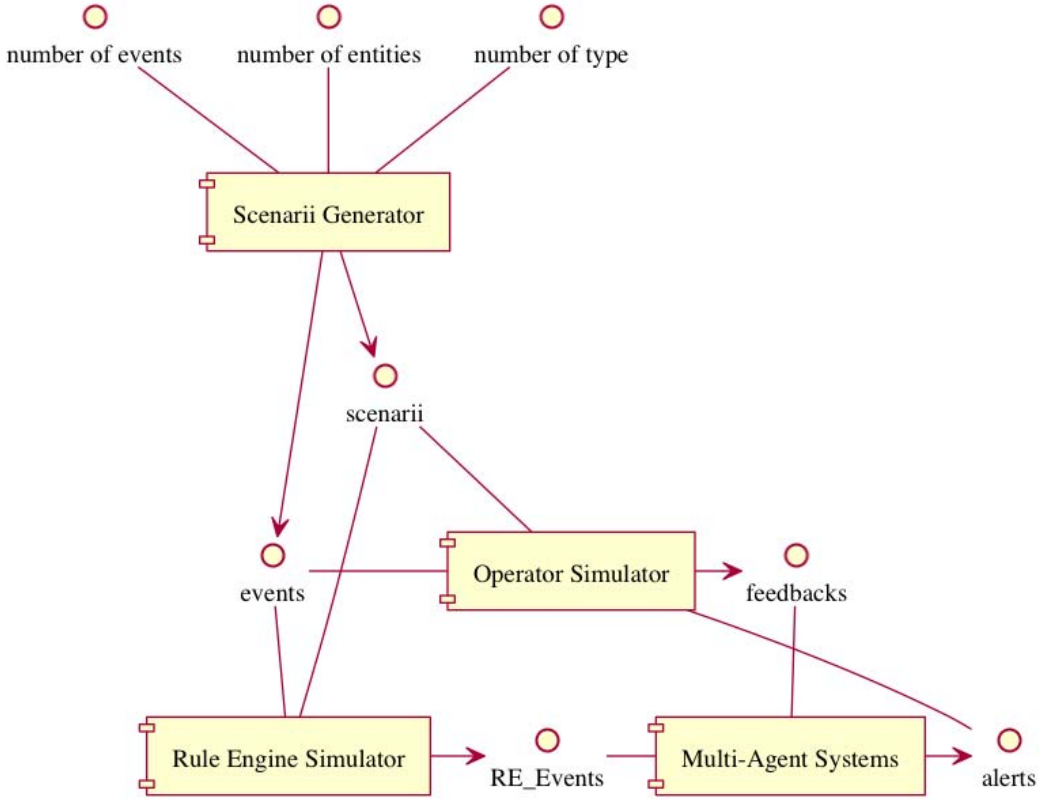

Figure 4.7: Architecture of SIM4AT.

4.4.2 Architecture of SIM4AT

SIM4AT is composed of several components that are interconnected, each one having a specific role within the platform. Figure 4.7 shows the different components and how they are connected to each other as well as to the muti-agent system. The current section explains the role of each part of SIM4AT.

There are three main components in SIM4AT: the *Scenario Generator*, the *Operator Simulator*, the *Rule Engine Simulator*. Besides these components, there are the *Anomaly Detection System* and a *Visualisation Component*. The first one is a component able to detect abnormal behaviours in the generated scenarios. In this thesis, this component is MAS4AT. The *Visualisation Component* allows the user of SIM4AT to observe the behaviours of the *Anomaly Detection System*. This can be used to tune this component according to the test results. When tested with MAS4AT, it allows to observe the convergence speed of each parameter-agent as well as the behaviours of each entities in real-time. These results are presented in chapter 5.

4.4.2.1 Scenarios Generator

In order to test surveillance systems, several means are available. One of them is the use of test scenarios. Indeed, the surveillance systems are designed to identify anomalies in behaviours and trigger alerts. To test such a system, it is common to design several known scenarios and run them through the systems. As already explained, a scenario is the course of actions and events on an entity during its journey in an area.

As they are designed for test purposes, the scenarios represent normal behaviours as well as abnormal ones. Then, the detection system runs them and the aim is to detect all the abnormal behaviours.

The component Scenarios Generator has been designed for this purpose. The user feeds the system with the maximum number of entities, the maximum number of events that can happen and the number of types of the entities. This can be useful when several kinds of entities are in the monitored area. Indeed, if the detection mechanisms are the same, the events happening might not be taken the same way for each kind.

First, the generator creates random events. According to what has been said in chapter 3, the events created in SIM4AT are composed of an identification and three values corresponding to the three parameters. Then the generator creates the entities. The entity in SIM4AT is more simple than the one modelled in chapter 3. Indeed, there is only a need for the identification of the entity and the scenario that is generated and associated to it.

Then the Scenarios Generator is able to provide scenarios randomly generated for each entity. The scenarios are sequences of events and ends of events randomly related to an entity. The generation is random but follows three simple rules in order to avoid rogue or useless scenarios :

- An event has 20% chance to start and 40% chance to end. We consider an event to be a particular moment in the lifetime of the entity, so it has a higher chance to disappear than to appear.
- When an event start, it must end before the end of the scenario. This prevents the generation of scenarios where there are only starts of events and no ends at all.
- It is not possible to have more than three times the same event in a row. This prevents the generation of scenarios where one event keeps happening, and only it, on an entity.

The generated scenarios are designed to be uncorrelated to any significance. This means that the events are identified by a unique identification, but no meaning is associated. It is the same for the entities. This way, the system is generic enough to be used in several domains. It is also possible to feed the system with already designed scenarios in order to test the detection for specific behaviours.

Finally, the Scenarios Generator also provides the number of Entities, the number of Events and the randomly created entities towards the Rule Engine Simulator and the

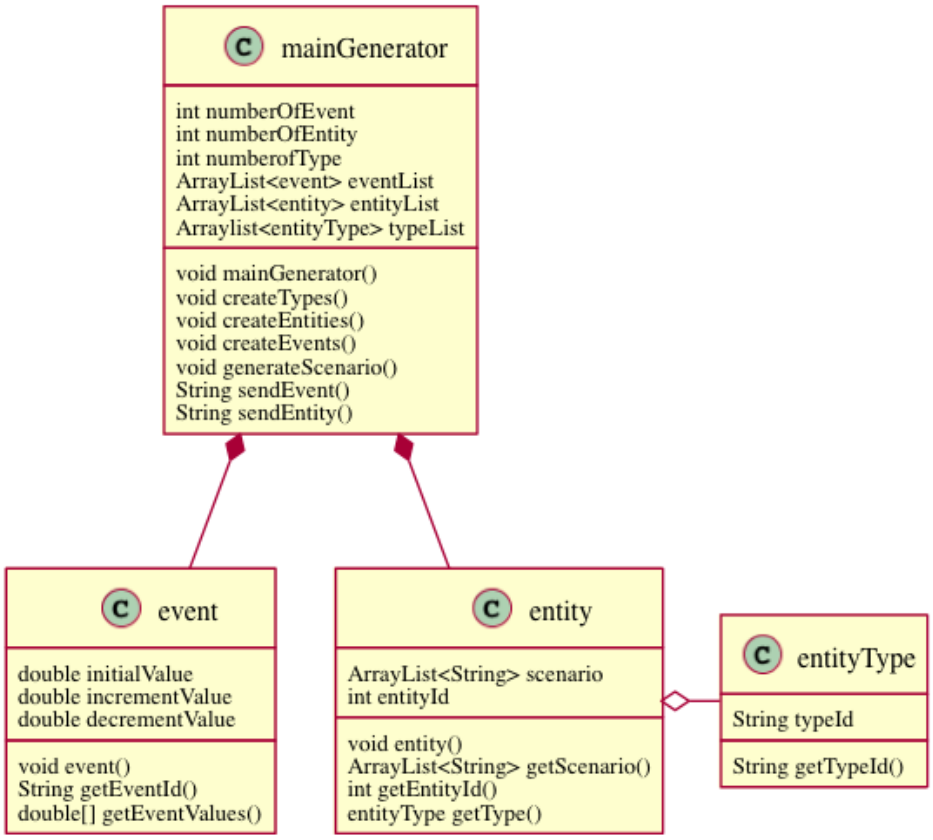

Figure 4.8: UML Model for Scenarios Generator.

Operator Simulator. These components use this information to run as described in the next subsections.

The UML model for the corresponding component is described on figure 4.8. It shows that the *mainGenerator* class is composed of a set of *events* and a set of *entities* holding the type, created by the generation process.

4.4.2.2 Operator Simulator

One of the other component of a surveillance system is the operator involved in the surveillance process. Its role is to monitor the entity and identify their abnormal behaviours in order to send assistance or to answer a threat.

In the project I2C the operator is also able to send feedback to the anomaly detection system by correcting it. It means that the operator is able to spot an alert missed by MAS4AT, but he is also able to indicate that an alert sent by MAS4AT is actually not an alert.

The component Operator Simulator of SIM4AT has the same role and provides these feedbacks. Once the entities the events and the scenarios have been created, the Scenarios Generator transmits it to the Operator Simulator. When testing a surveillance system, the

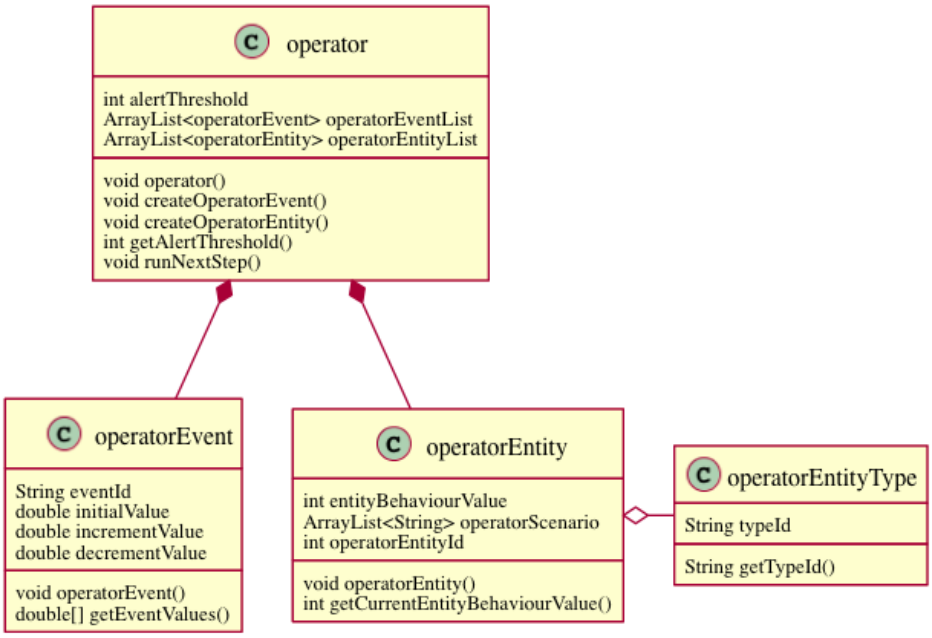

Figure 4.9: UML Model for Operator Simulator.

operator knows the scenarios that are tested in order to evaluate the detection. Besides, he is able to assess and evaluates the events that are used in the scenarios.

These features have been implemented in the Operator Simulator. Thus, it is able to run the scenarios parallel to the detection engine, using the real values of the events. Therefore, when the multi-agent component sends an alert to the operator simulator, this last is able to compare it with its own knowledge and invalidate the alert if needed. On the contrary, if the multi-agent system does not raise an alert and the operator simulator finds one on an entity, it can send the corresponding feedback.

The UML model for the component Operator Simulator is given on figure 4.9. This model shows that this component possesses its own knowledge on the entities and the events, allowing it to be independent in its comparison with the detection system results. This has been designed this way to be as close as possible to the real operators who possess their own knowledge and confront it to the surveillance system when assessing a situation.

4.4.2.3 Rule Engine Simulator

In a surveillance system like I2C, there is a rule engine designed to analyse the data aggregated on each entities of the monitored area. The rule engine sends events and anomalies according to its rules and towards the Multi-Agent System MAS4AT in order to take them into account.

In SIM4AT, a Rule Engine Simulator has been designed to take this role. It receives the scenarios generated by the *Scenarios Generator* and reads them, step by step, for each

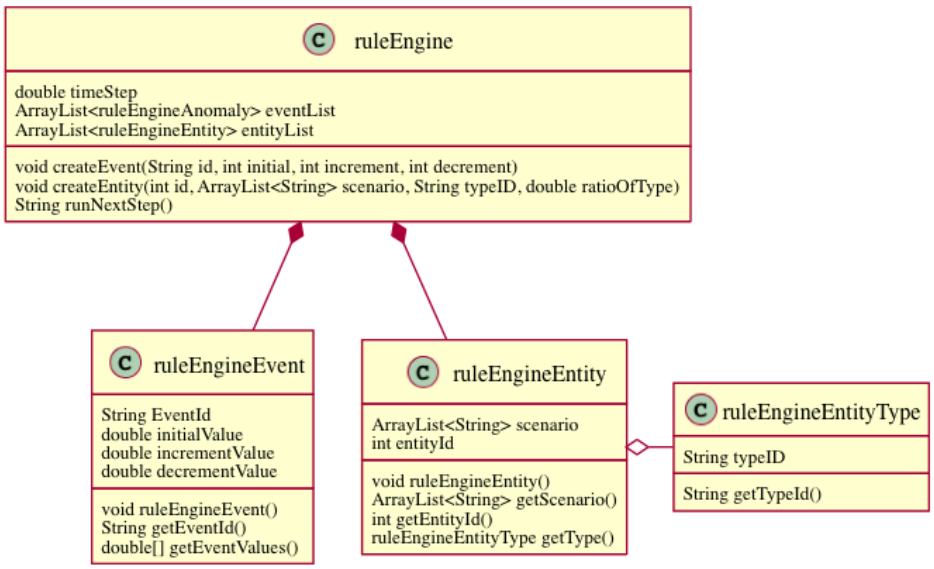

Figure 4.10: UML Model for Rule Engine Simulator.

entity. Whenever an event is encountered, it is sent to MAS4AT as well as the corresponding entity. This component only simulates a rule engine because it reads scenarios in order to send events. The role and operation stays the same: send events to MAS4AT. Figure 4.10 describes the UML model of the Rule Engine Simulator.

4.4.2.4 Anomaly Detection System

The connections between the SIM4AT and MAS4AT is direct: the events coming from the Rule Engine Simulator can be consumed by the component OpMAS (Operative Multi-Agent System) of MAS4AT. The corresponding ship-agent is able to retrieve the values of the event from PaMAS (Parameter Multi-Agent System) and compute its ship behaviour value. When exceeding a threshold, the ship-agent then sends an alert towards the Operator Simulator. Finally, this last can send a feedback to MAS4AT if needed.

However, it can be useful to test other detection systems. This way, SIM4AT can be connected with another component that would replace MAS4AT. Of course, the anomaly detection system must meet the same requirements and needs:

- The events must be decomposed into three parameters and a value must be given to each parameter.
- The detection system must have a way to represent both the entities and the events.
- The entities represented in the detection system must provide a way to compute a numerical value indicating the suspicion level.

- It is not a requirement per se, but the detection system should be able to provide alerts toward the Operator Simulator and it should be able to receive feedback as well.

If a detection system fulfils these requirements, it is easily pluggable to SIM4AT. Then SIM4AT can be used to test the anomaly detection as described in section 4.4.3.

4.4.3 SIM4AT Operation Process

Figure 4.11 described the running process of the platform SIM4AT plugged with MAS4AT. After the initialisation phase, SIM4AT performs the simulation of the environment of a surveillance system, from the reading of data to the detection of abnormal behaviours. The *Rule Engine Simulator* reads the scenarios for each entity and eventually sends a pair of values to MAS4AT. This pair is composed of the identification of an event as well as the identification of an entity. This is done when the related scenarios indicates that an event happens.

If the event or the entity is not known by MAS4AT, it creates the corresponding agents –entity-agent and / or event-agent and parameter-agent. Then, it computes the *Entity Behaviour Value* (see chapter 3 of the corresponding entity. When the value exceeds the alert threshold defined in MAS4AT, it is sent to the *Operator Simulator*. Parallel to the MAS4AT process, the *Operator Simulator* uses its knowledge on the events and the entities to compute its own entity behaviour value. This value is then compared to the value sent by MAS4AT.

The comparison is equivalent to check if these values are on the same side of the threshold. If it is the case, this means that the *Simulator Operator* and MAS4AT agree to say that the concerned entity triggers an alert, or not. On the contrary, if these two values are not on the same side of the threshold, this means that the two components disagree and the alert state of the entity. As for the real system, we consider that the *Operator Simulator* has the true knowledge on the events and the entity behaviours. This means that it is MAS4AT that is wrong and it has to self-correct.

Therefore, if the two values are different, the *Simulator Operator* sends a feedback to MAS4AT. It induces the tuning process of the parameter-agents in MAS4AT. Finally, the Rule Engine Simulator proceeds to the next entity or to the next step in the entities scenarios. This ends when all the scenarios have been read. During all the process, the user is able to observe the results and the values of the parameter-agents through the *Visualisation Tool*. Figure 4.12 shows the interface of the Visualisation Tool. This allows to observe the convergence of the parameters towards the values created by the generator and validating the Anomaly Detection System. The user is also able to observe the Entity Behaviour Values of each entity, both computed by MAS4AT and the operator simulator.

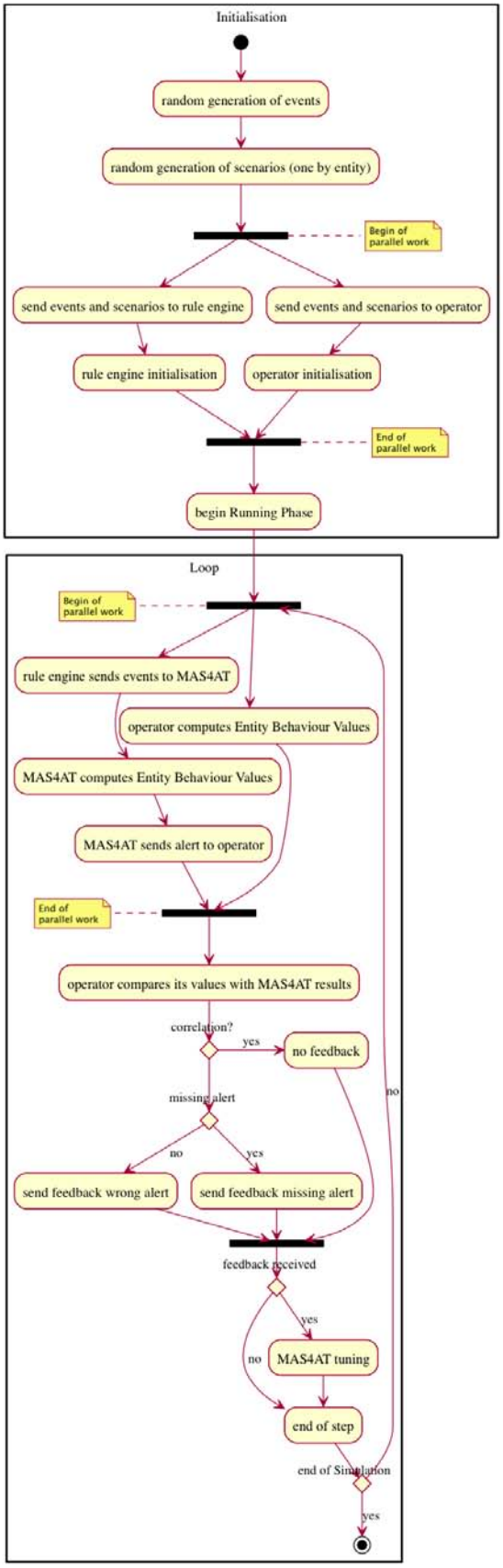

Figure 4.11: Running Order of SIM4AT.

Figure 4.12: Graphical Interface of SIM4AT.

4.5 Conclusion

In this chapter, we present the I2C project, the European funded project aiming at the integration and interoperability of several tools in order to provide a maritime surveillance system. This project is the context of my PhD thesis, it integrates arrays of sensors and radars supplying an intelligent situational picture of the monitored area and the ships evolving in it. The events detected on the ships are then sent to the Behaviour Analysis (BEAN) component in order to be processed.

This chapter also describes the application of MAS4AT in this component. We described the differences with the generic model presented in chapter 3. In order to reach a representation closer to the reality, we decomposed MAS4AT into two sub multi-agent systems, the Operative Multi-Agent System (OpMAS) which represents the monitored vessels and computes their behaviour values, and the Parameter Multi-Agent System (PaMAS) representing the events that can happen and manage their values. These values are learnt by MAS4AT and more specifically by PaMAS, according to operator feedbacks.

Therefore, in I2C, the MAS4AT component is able to use data from the sensors processed through the intelligent picture and a rule engine into BEAN in order to trigger alert when the suspicion level on a vessel is too high. This leads to the identification of abnormal behaviours and the concerned authorities are then able to take actions.

The integration of our agent model in I2C allows us to validate its genericness. Indeed, the interfaces between MAS4AT and the other components are kept minimal but they are nonetheless sufficient for OpMAS and PaMAS to fulfil their roles: the alert triggering and

the learning. Moreover, this integration in an industrial project also validates the concept and the operation of our multi-agent system.

Finally, this chapter presents the platform *Simulators for Alert Triggering*. We have designed SIM4AT as a set of tools designed and implemented in order to test MAS4AT in real conditions without the inputs from the real components. Indeed, input data are created by simulators and transmitted to the multi-agent system as would be data from real devices. SIM4AT is also able to handle the outputs of MAS4AT and compare it with its knowledge in order to send accurate feedbacks for the learning. This platform is useful to test our model and, since it is generic, it is usable in order to test other alert triggering systems as long as they use an event and entity behaviour representation similar to the one in I2C.

Ultimately, the next logical step is to test our multi-agent system to validate its efficiency regarding two objectives: the learning of the parameter values of the events and the alert triggering. The first objective is generalized by the learning of a unknown mathematical function. The second objective is the accurate alert triggering of MAS4AT, meaning it raises the same alerts as the operators. This is reached by an efficient learnings.

5 Experimentation, Evaluation and Validation of MAS4AT

« A single test which proves some piece of theory wrong is more valuable than a hundred tests showing that idea might be true. »

Arthur C. Clarke

Contents

The chapter in English starts page 101.

Résumé général du chapitre

Lors des précédents chapitres, nous avons définis le système multi-agent MAS4AT pour l'apprentissage des paramètres d'une fonction et la levée d'alerte dans un système de surveillance. Nous avons ensuite détaillé le contexte d'application de ce travail, le projet I2C et les adaptations nécessaires dans MAS4AT. L'accès aux données réelles n'étant pas possible au moment de ce manuscrit, nous avons aussi mis en place une plateforme de simulation pour nous permettre de tester le système en condition.

Il existe plusieurs critères pour tester et évaluer un système : l'adéquation de la solution, l'utilisation des ressources matérielles, le temps d'exécution, la généricité,/dots Dans le cas de MAS4AT, les agents doivent être capable de trouver les valeurs des évènements qu'ils représentent dans un temps limité et avec une utilisation des ressources qui reste faible en raison du grand nombre de navires et d'évènements. Dans ce chapitre, nous nous concentrons donc sur la convergence des agents vers les valeurs à trouver.

Dans un premier temps, nous observons la convergence des agent-paramètres dans un cas simple de manière à valider les mécanismes d'apprentissage et d'ajustement. Cette expérimentation est faite dans le cas d'un seul évènement puis de trente-trois pour valider le processus avec plusieurs agents. Nous pouvons observer que les valeurs des agents convergent vers les valeurs à trouver, le tout avec un nombre limité de retours utilisateurs. Ici, c'est le nombre de retour qui est important et non le temps de convergence car il peut se passer plusieurs heures, voire plusieurs jours, sans qu'une situation anormale n'apparaisse.

Une fois l'apprentissage validée, nous avons menées plusieurs expérimentations pour valider le comportement des agents quand ils sont confrontés à des entités qui suivent des scénarios comportementaux. Encore une fois, nous pouvons observer la convergence des agents vers les valeurs à trouver. De plus, les courbes de comportements estimées par les agents sont proches des courbes "réelles" et les alertes des agents sont cohérentes avec les alertes réelles. Nous pouvons aussi voir que le nombre de retours des utilisateurs diminue avec le temps et qu'ils sont de plus en plus espacés. Cela traduit le fait que les agents sont des plus en plus proches des valeurs à trouver et que de moins en moins de corrections sont nécessaires.

Si la convergence des agents est un aspect important pour MAS4AT, dans un système de surveillance, il peut y avoir des erreurs, que ce soit intentionnel ou non. Nous avons donc testé le système avec deux taux d'erreur. Quand il n'y a pas beaucoup d'erreur, le système reste néanmoins capable de converger et de proposer des valeurs correctes et donc des alertes pertinentes. En revanche, plus ce taux d'erreurs augmente, moins le système converge et il devient incapable de lever des alertes. Malgré tout ce fonctionnement est bénéfique au système de surveillance car, si MAS4AT convergeait, les alertes qu'il lèverait seraient certainement erronées. De même, si les opérateurs de surveillance acquièrent une nouvelle connaissance et que les évènements n'ont plus la même

importance, le système doit être capable de prendre en compte les nouvelles valeurs. C'est ce que montre l'expérimentation suivante.

Enfin, MAS4AT est basé sur des agents en accord avec la théorie des AMAS. Nous avons voulu confronter le système avec une méthode de résolution d'équation classique, la méthode de Gauss-Jordan. Nous pouvons remarquer que dans les tests que nous avons menés, la méthode mathématique nécessite un nombre beaucoup moins important de retours des utilisateurs. Cela doit être balancé par le fait que le système multi-agent est conu pour être implanté dans un système dynamique. En fait, le nombre de retours moyen est fixe au cours du temps. Dans le cas de la méthode de Gauss-Jordan, elle doit être exécutée à chaque itération du système et donc, au final, le nombre de feedback est plus important. De plus, cette méthode nécessite une connaissance complète du système, ce qui n'est pas nécessairement le cas dans un système de surveillance.

Ce chapitre détaille et analyse les différentes expérimentations menées en accord avec certains critÃ¨res explicités dans la première ection. La conclusion résume les résultats et ouvre sur des perspectives.

5.1 Introduction

In order to validate the MAS4AT agent model that was developed, we have designed several cases to study, from the learning evaluation to the comparison with another technique through the evaluation of detection in scenarios and the robustness of the multi-agent system. The experimentations have been made using SIM4AT plugged to MAS4AT.

The use of MAS4AT allows to represent the complexity of the problem due to the high number of events provided to MAS4AT. This number is not known beforehand and has to be handled by the agents. The dynamics of the system is also represented as new events can appear at any time.

In this chapter, section 5.2 presents the different indicators we used in order to evaluate MAS4AT. This indicators are both qualitative and quantitative and are well suited to evaluate the result of the system. Then, in section 5.3, we detail the different case studies we have designed to perform the experimentations. For each one, a description of the case study explains the aim of the current evaluation and how it is performed. Then, the results are presented and analysed in respect of the presented criteria.

Then, in section 5.4 robustness of MAS4AT is tested by the injection of several false feedbacks and the modification of the values of the events, the values that have to be learnt by the agents. Finally, section 5.5 compares the results from MAS4AT and the results from a commonly used mathematical solving technique.

In the end, section 4.5 concludes the chapter and discusses the perspectives of MAS4AT. It also underlines the advantages of the agent model we have defined.

5.2 Indicators for the Evaluation of MAS4AT

This section describes the different criteria that are used to evaluate MAS4AT in regard to its use in the I2C system. Indeed, the first objective of MAS4AT is the triggering of relevant alerts, according to its knowledge and its representation of the supervised area and entities. The other objective of our system is its ability to learn the values of the events following the feedbacks and in order to raiser more and more accurate alerts.

We have designed qualitative and quantitative criteria. The qualitative criteria are used to study the compliance of MAS4AT with the I2C system requirements. They are not measurable by themselves. Nonetheless, they are presented in this section since they represent an important feature of the multi-agent system. On the contrary, the quantitative criteria are designed to measure the speed and accuracy of the learning performed by the agents. We focus on the learning and not on the alert triggering because, in MAS4AT, accurate values for the parameters ensure an accurate alert triggering. Besides, in the I2C system, alert triggering is the result of abstract knowledge different for each operator, which is learnt by MAS4AT.

5.2.1 Qualitative criteria

These criteria are subjective ones. Indeed, they are depending on the context and the domain of the system we want to evaluate. Most of them are coming from, or are inspired by the software engineering. The qualitative criteria we have designed have two objectives: first they ensure that MAS4AT is relevant with the other components in the I2C system; second they ensure that MAS4AT is generic and has a low cost in term of resources.

Adequacy of the solution: The system is adequate when useful for its environment. In the case of MAS4AT, it means that the system is able to raise alerts when a suspicious behaviour is detected. The adequacy of the solution can be measured by the distance from the optimal solution. However, this is difficult to evaluate since in the majority of cases, the optimal solution is not known. In the literature, scenarios are generally used for this comparison and the system is validate if it runs on a sufficient numbers of different scenarios. This "sufficient number" is determined by experts on the domain.

Easy-linking and modularity: One of the requirements of the I2C system is the modularity of the various components. This is achieved by the observance of the interfaces defined in the project and specific to each component according to its needs and results. Therefore, the interfaces for the MAS4AT system are limited to the events and the feedbacks as possible inputs, respectively from the rule engine and from the operators, and the alerts for the outputs.

Memory usage: Another requirement is the low memory cost of the component. As the I2C system must handle thousands of entities and hundreds of possible events, their

representation may be costly in term of memory and hardware resources. Thus, the agents have to present a memory usage as low as possible. In MAS4AT, it is reached by the decentralisation implied by the agents and the use of local algorithms.

Maintainability and usability: The I2C system is due to be used by several operators and in several contexts. Once implanted in it, MAS4AT should be easy to understand and maintain by the software engineers that may modify it in order to tune the system according to their specific needs.

Genericness: MAS4AT has been applied to a maritime surveillance system. However, we have designed it be generic and we advocate its usability in other domains. The only requirements are the representation of the entities as agents and their behaviours as numerical values. Then, the system is applicable to other domains like the air surveillance –where the relation with maritime surveillance is direct– or networks surveillance –where the entities might be the nodes of the networks and the events can be system failures on them.

The main problem of these measures is that they are difficult to represent or evaluate. They are achieved by the observance of the requirements of the projects, the specifications and the conventions. In order to assess the efficiency of MAS4AT in term of alert triggering and learning, we have defined several quantitative criteria.

5.2.2 Quantitative criteria

Contrary to the qualitative ones, the quantitative criteria are objective. They represent observable measures and the ones withheld here are designed to be especially used to evaluate the multi-agent component of the system. They can be used to evaluate the performance of the MAS as well as to validate the strength of the MAS while running.

Parameter convergence: The first criteria is the convergence of the parameters towards consistent values. During the tests, these values are known and it is easy to observe and measure the convergence. On the contrary, while running, this observation can be difficult to do as the real values are not known.

Entity convergence: As for the parameters, we can observe the convergence of each Entity Behaviour Value towards the real one, meaning the one computed with the known values of the parameters. Once again, this is easier to observe during the test process.

Global convergence: The global convergence of the system is useful when handling several tenth of entities and events. Indeed, the observation of each parameter independently may be difficult. To answer it, we propose to observe the convergence of the largest gap between the parameter values and the real values. The tinier is this gap, the more the system is close to the real values.

Number of tuning: The number of tuning is useful to represent the confidence of the agents in their solution. On one hand, he more they tune themselves, the less they are confident since they are still trying to reach accurate values. On the other hand, if they are performing few tuning, they will have a greater confidence in their solution since this means that they receive few feedbacks. Thus, the system is able to evaluate the quality of its solution.

Distance between feedback over time: This measure has been introduced to tackle the problem of the unknown values when it comes to the observation of the convergence speed. Since it is not possible, we propose to observe the number of feedbacks sent by the operators as well as their spacing. Indeed, the more there is time between two feedbacks, the more MAS4AT is close to the operator representation of situations; on the contrary, if there are a lot of feedbacks in a short time, this means MAS4AT is often wrong and the agents are far from the "good" values.

These criteria are used in the following test to validate MAS4AT regarding two aspects: the parameter value learning and the alert triggering. The comparison of the results for cases with a few parameters and entities with the results for tenth of parameters and entities also allows us to validate the strength of our multi-agent system when facing scalability.

5.3 Case Studies: Definitions, Results and Analysis

First, the agent-based model MAS4AT presented in chapter 3 is able to learn the values of the parameters of an unknown mathematical function –see 3.3.2. The only knowledge of the system is that the function of the form given by function 5.1. In this function w_i^t is the known value weighting each parameter while a_i is the value of the parameter, the value to learn.

$$r^t = \sum_{i=1}^{n} a_i \times w_i^t \tag{5.1}$$

Second, this function is used to represent the suspicious level of entities –see 3.3.1. Here, the weight of each parameter is given by the environment of the system and is therefore known by the system. The resulting Entity Behaviour Value (EBV) allows the system to trigger alerts towards the operator involved in the maritime surveillance. As the values of the events are not known, the system has to learn them. It uses feedback from the operators correcting MAS4AT. Thus, in the I2C surveillance system, MAS4AT should deal with several entities and it should be able to learn from the operators and raise relevant alerts.

This section presents the tests performed on MAS4AT in order to evaluate and validate these two features of MAS4AT. First, section 5.3.1 presents the evaluation of the learning of the parameter of an unknown function. The objective is to validate the learning principle of our system. Second, section 5.3.2 presents the evaluation of both learning and alert triggering in real condition. MAS4AT is plugged in SIM4AT which provides a simulation

of the I2C environment for our multi-agent system. Then, MAS4AT handles several entities and events and is both able to learn the values of the event parameters and raise alerts.

5.3.1 Validation of the Learning Algorithms

First, MAS4AT is able to learn the parameter values of an unknown function. The only knowledge on the function is that it can be represented using the formula 5.1. It has to be noted that we bypass the entity-agents since they are not involved in the learning process, apart for the feedback transmission.

In chapter 3, we explain that the agents are combined to form an inequality and that the learning is performed according to this inequality. The idea is that the agents are able to tune themselves as their result is compliant with the current inequality. So for each time step, a new inequality is taken into account and the agents eventually have to tune themselves. In these inequalities, only the values of the weights associated to each events are changing, the value to learn stay the same over time.

Therefore, the test algorithm for the learning is the following:

---

generate x event values to learn
Repeat
generate one weight for each parameter of each event
if *an event has no agent counterpart* **then**
| create a new event-agent
end
AV ← result of the function using agent values
RV ← result of the function using real values
if *notComply(AV,RV)* **then**
| Send feedback to the agents
end
Until(Learnt values are equal to values to learn)

The values to learn are generated between 0 and 100. At each time step the result of the function computed with the agent values is compared to the result computed with the real value. When the two values are on opposite side of the inequality, a feedback is sent to the agents. This feedback induces the tuning process of the parameter-agents –see 3.3.2.

This algorithm is used in three cases:

- ▷ First we test the learning process with only one event composed of one parameter.
- ▷ Second, we perform new tests with one event composed of three parameters.
- ▷ Third, we test the learning process with thirty-three events composed of three parameters each.

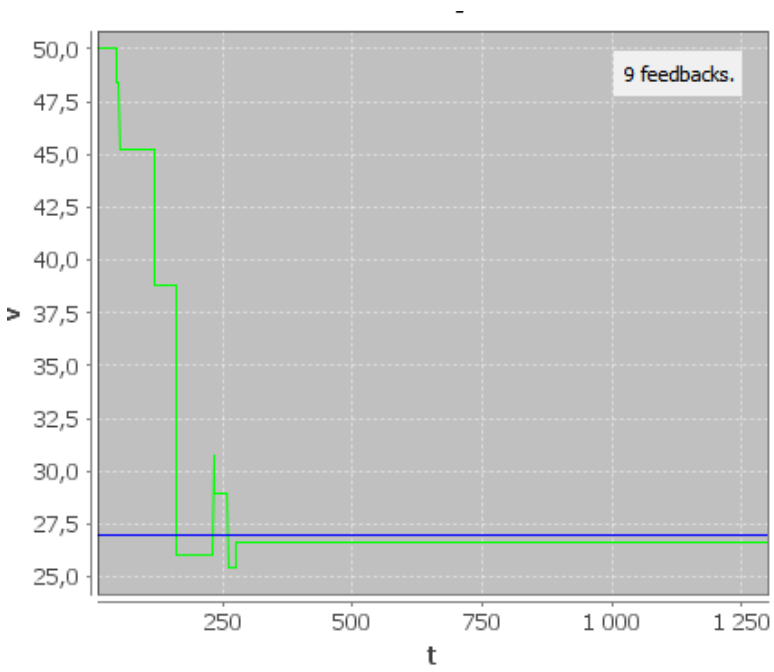

Figure 5.1: Parameter-agent convergence while searching one parameter value.

The first two steps allow us to validate the learning principle both in a generic way and in the I2C context –where one event contains three parameters. The third case allows us to test MAS4AT when facing a huge number of events.

First, it has to be noted that each figure below presents the average results of ten running operations.

Figure 5.1 shows the result of the learning process of MAS4AT for one parameter. Each time it receives the feedback, the parameter-agent performs a tuning of its value using the *Adaptive Value Tracker (AVT)*. We can observe the convergence of the parameter-agent towards the value it has to learn.

On figure 5.1, the green curve represents the mean value learnt by the parameter-agent during this series of 10 tests and the blue curve represents the value to learn, 27.5. We can observe that an average number of 9 feedbacks are needed for the parameter-agent to reach the value to find. The feedbacks are based on the compliance of the parameter-agent with generated anomalies. This explains why there is not a feedback at each time step and why the value is learn after 250 iterations. Indeed, in the meanwhile, the agent possesses a value relevant according to the inequalities, so no feedback is sent.

The results shown on figure 5.1 expose the well functioning of both our method and the AVT. In MAS4AT, each event participating to the representation of a situation is composed of three parameters. Thus, figure 5.2 shows the convergence of three parameter-agents towards their own values to find. When the agents are receiving a feedback, they tune their value according to the algorithms presented in chapter 3. If they reach a dead-end and no tuning is

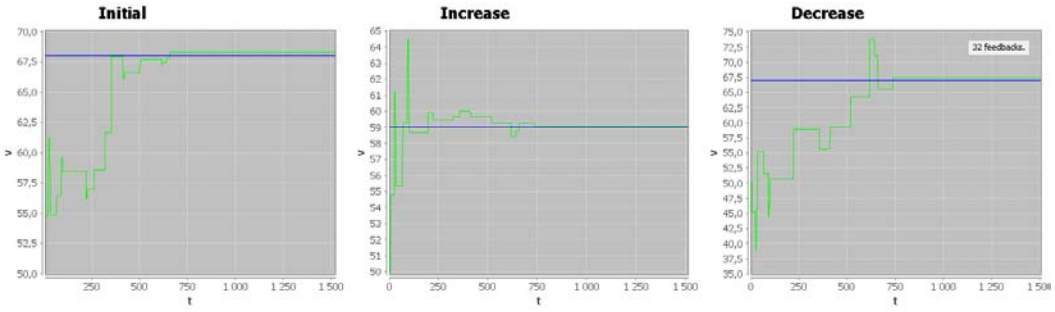

Figure 5.2: Parameter-agent convergences while searching three parameter values.

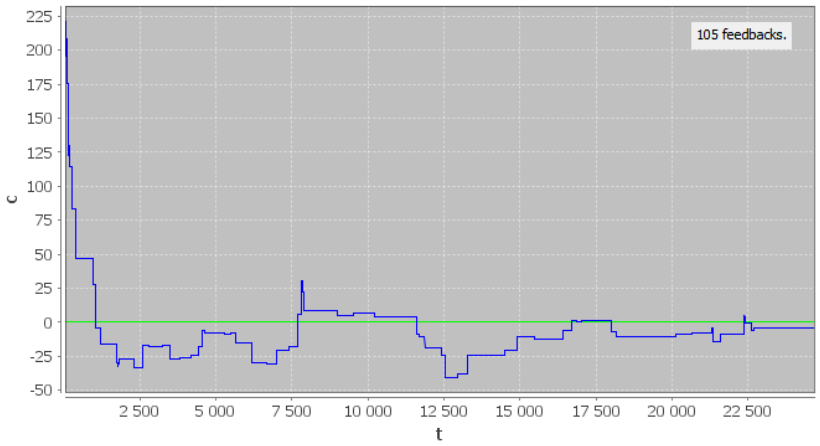

Figure 5.3: System convergence with 99 parameter-agents.

done, they are able to determine which one of them is the less constraint and force its tuning. This prevents the system to be inactive for several steps. On the figure, we can observe that the agents are able to find their values.

On figure 5.2, we can observe the mean convergence for each one of the three parameters during 10 tests. The value to reach for the initial parameter is 67.6, it is 59 for the increase parameter and 67.4 for the decrease parameter. With 32 feedbacks on average, the agents are able to learn the right values using randomly generated inequalities.

These results shows that the decomposition of an event into three parameters has no impact on the learning process, neither on the results. Ultimately, in the real I2C project, MAS4AT might be confronted to several tenth of entities and events. Therefore, we experiment the learning with 33 event-agents, meaning 99 parameter-agents. Figure 5.3 shows the results as the global convergence of the system.

Figure 5.3 illustrates the convergence of the greater error –blue curve– towards 0 with an average number of 105 feedbacks. It illustrates the convergence of the parameter-agents towards their value to learn. Indeed, the more the system convergence is close to 0, the more each parameter-agent is close to its right value. We can also observe some leaps in the

curve, when the greater error is close to 0 before receding farther away. These are due to the learning process. Indeed, the tuning of the agents can provoke tuning of agents close to their values and new feedbacks allow them to reach it again. Examples are at $t \simeq 10000$ and $t \simeq 12000$ or $t \simeq 17500$ and $t \simeq 18000$.

The results we present in this section show that the parameter-agents in MAS4AT are able to converge towards the real values defined for the tests. Even if the values are not exactly the same, they are close enough to prevent the appearance of new feedbacks. These results validate the learning process by self-tuning of the parameter-agents according to feedbacks. Besides, their encapsulation in event-agents does not impact the results.

5.3.2 Validation of the Alert Triggering

In the previous section, we describe the results allowing us to validate our method of learning of the parameters of a function. As MAS4AT is designed to be applied into a surveillance system, we now focus on the alert triggering feature. Using SIM4AT, we build several entities and one scenario for each. These scenario are randomly created at first, their meaning is not studied here. Then each entity-agent is able to compute its *Entity Behaviour Value (EBV)* and raise an alert towards the *Operator Simulator (OS)* when this value exceeds a threshold. During these tests, the threshold value is set to an arbitrary value equals to 800. The OS compares the alerts with its own knowledge and sends a feedback if it disagrees with MAS4AT. When a feedback is received by an entity-agent, it is broadcast to the concerned event-agents, the ones involved in the current situation.

The testing algorithm is the same as the one in the previous section except for the weight of the events. Indeed, they are not randomly generated but are the consequences of the scenarios reading by the rule engine simulator. Indeed, following the reception of events, the entity-agents are able to memorise the number or appearance, of disappearance and the lasting time of each event.

Then, the event-agents and the parameter-agents are able to tune themselves according to the feedbacks they receive. The number of feedback is higher since several entities are using several same events to represent situations from their respective scenario. Second, each situation is different from the other, meaning than when a feedback is received, all the agents are not necessarily concerned. Thus, the agents have to cope with it using their skills and the algorithms described in chapter 3.

We first investigate a simple scenario for a unique entity. The scenario is played once and the parameter-agents are able to tune themselves during its reading. We observe the convergence of the parameter-agents as well as the convergence of the EBV. Second, we investigate the convergence of the EBV of several entities subject to several different scenarios. We also monitor the number of feedbacks and their distribution.

Figure 5.4 presents the convergence of the system when reading one scenario for one entity. We can observe that the parameter-agents are able to track down the "real" values

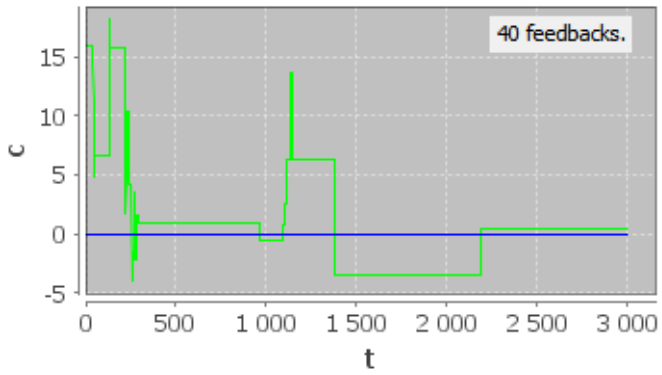

Figure 5.4: System convergence with one scenario and one entity.

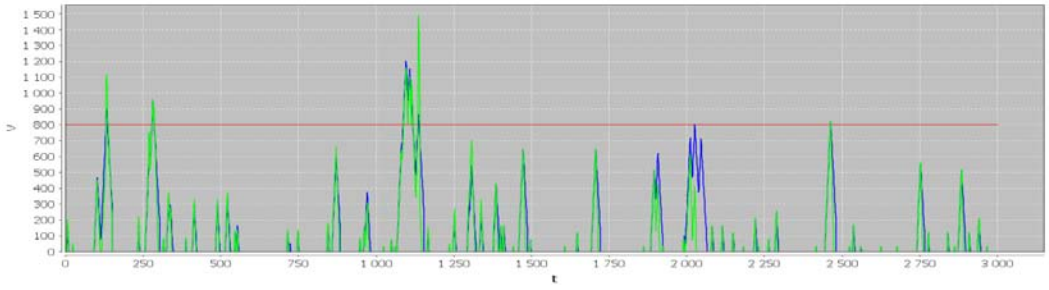

Figure 5.5: Entity Behaviour Computation and Coomparison.

with time. However, this does not show if the resulting alert triggering is correct or not. We have to study the *Entity Behaviour Value (EBV)*.

Figure 5.5 shows the mathematical curve representing the *Entity Behaviour Value (EBV* over time. The blue curve is the EBV computed with the real values while the green curve is the EBV computed using the agent values. The red line symbolises the alert threshold. When the EBV is over the threshold, the related entity is in alert, and vice versa. We observe that the gap between the blue curve and the green one is reducing. Indeed, as the parameter-agents become more accurate in their value, the computation of EBV is also more and more precise. As a result, we can observe that when the green line represents an alert by exceeding the threshold, the agents are able to represent this alert as well.

These two figures allow us to validate both the agent learning when using scenarios and entity-agents, but also the alert triggering. Indeed, it is a direct consequence to the finding of the right parameter values. Another metrics is the correlation between successive feedbacks and the error.

Figure 5.6 shows the time between two successive feedbacks as well as the distance to the real values when the feedback occurs. We observe that when this error approaches 0, meaning the parameters are more and more close to the right values, the time between two successive feedbacks is increasing. This is useful because it shows that less and less

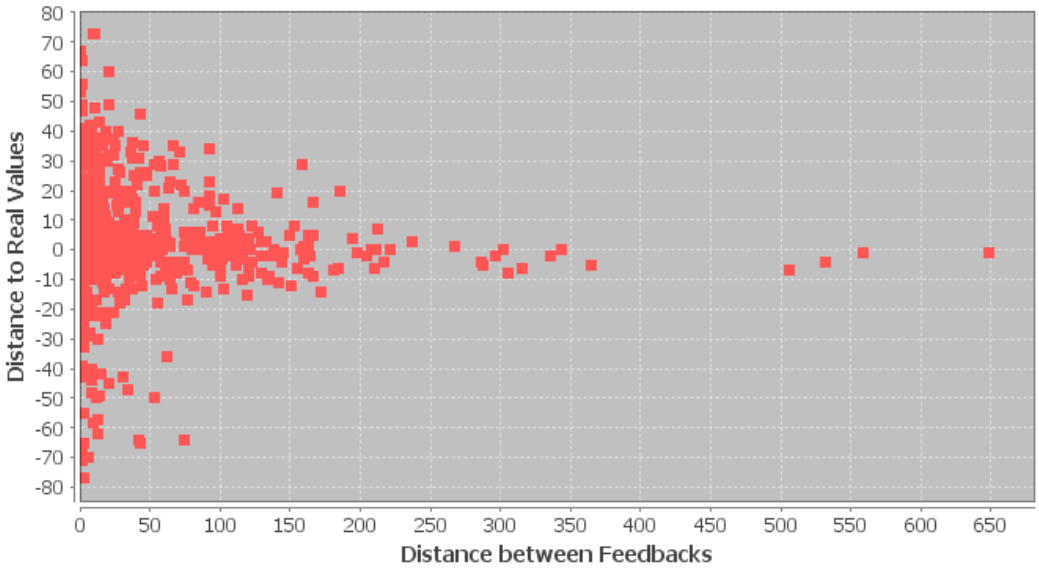

Figure 5.6: Number of Feedbacks during the process.

corrections are needed. This is another measure showing the convergence of the agents, but also the confidence of the system in its solution.

In the real final system, MAS4AT might be confronted to several tenth of entities following various unknown schemes involving several tenth of events. To study the behaviour of MAS4AT when facing such numbers, we propose an experiment with ten entities choosing a scenario amongst twenty possible. Each scenario is built using up to 20 events. The entities run their scenario and choose another one once they are done with the firs one.

The figure 5.7 shows the system convergence. We can observe that the agents are able to reach accurate values as the largest gap is reducing over time. This way, the learning process is validated when confronted to several entities and events.

Besides, the alert triggering is also validated in these conditions. Figures 5.8 and 5.9 show two examples of the computation of an Entity Behaviour Value while performing the current experiment. We can observe that the blue curves –"agent EBV"– are more and more close to the green ones –"real EBV"–, meaning alerts are more and more accurate and relevant.

The results in this section show that MAS4AT is able to handle both the learning and the alert triggering while plunged in a simulated environment. Indeed, the parameter-agents are able to converge towards accurate values while the entity-agents are able to compute coherent EBV and raise relevant alerts. While the system is running, less and less feedbacks are needed meaning that the Operator Simulator more and more agree with MAS4AT.

These results allow us to validate the principle and usability of MAS4AT. Therefore the system is well suited to answer the requirements of the I2C system: MAS4AT is able to learn

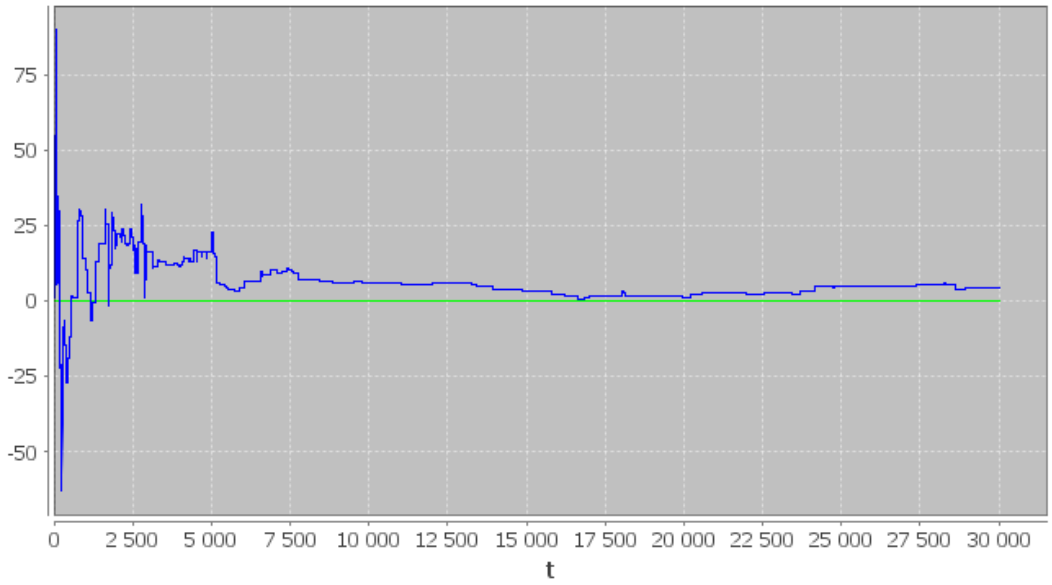

Figure 5.7: System convergence with 10 entities and 20 possible scenarios.

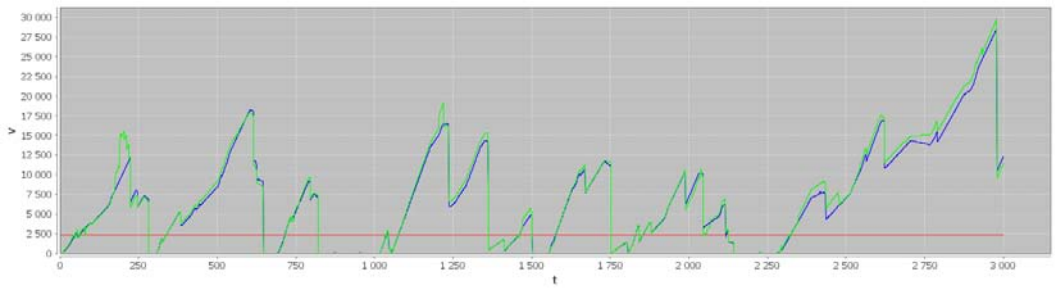

Figure 5.8: Example of the result of the tuning on the EBV (a).

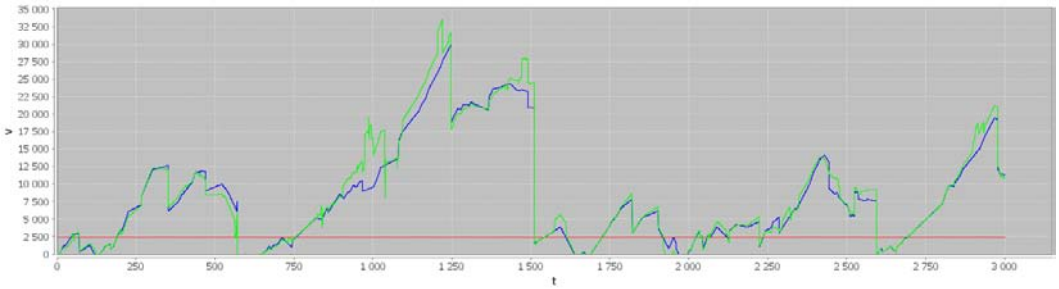

Figure 5.9: Example of the result of the tuning on the EBV (b).

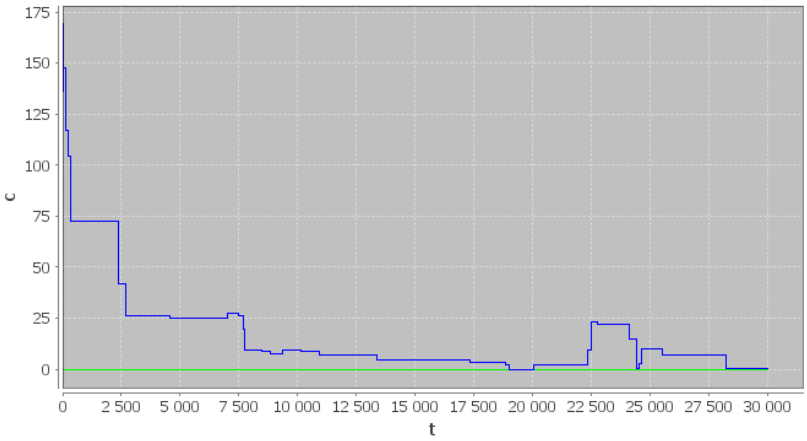

Figure 5.10: Global convergence of the system subject to 5% of false feedbacks.

from the real operators and the triggered alerts will be more and more relevant and useful to them.

5.4 Robustness of MAS4AT

In this section, we test the robustness of MAS4AT to such errors. To reach this goal, we introduce a false feedback mechanism in MAS4AT. This device randomly sends feedbacks towards the multi-agent system with a variable probability. We then exposes the results of the learning and alert triggering performed by the agents regarding both a low rate of error and a high one.

Besides the false feedbacks, MAS4AT might be subject to the operator change. Indeed, several operators are involved in a surveillance system. Thus, the operators might consider one event in different ways: one considers it as banal while the other might consider it important. In MAS4AT, this is represented by the change of the parameter values of this events. Therefore, we have to study the impact of this change on the agents and the results they provide.

We first submit the system to 5% of false feedback, randomly sent to MAS4AT. A false feedback is a feedback indicating that the agents have to tune themselves while there is no actual need for it. For example, when the agents do not trigger an alert, the system could indicate to the agents that one is needed. We observe the impact on the parameter convergence. This experiment is conducted with 10 event-agents, thus 30 parameter-agents. Figure 5.10 shows the impact on the system convergence. We can observe that the resulting curve is similar to the ones in section 5.3.1. Therefore, we can say that there is no visible impact on the system.

We now study the impact of 80% of false feedbacks, see figure 5.11. In this case, the system seems highly disturbed and the agents are unable to find the good values. Whenever

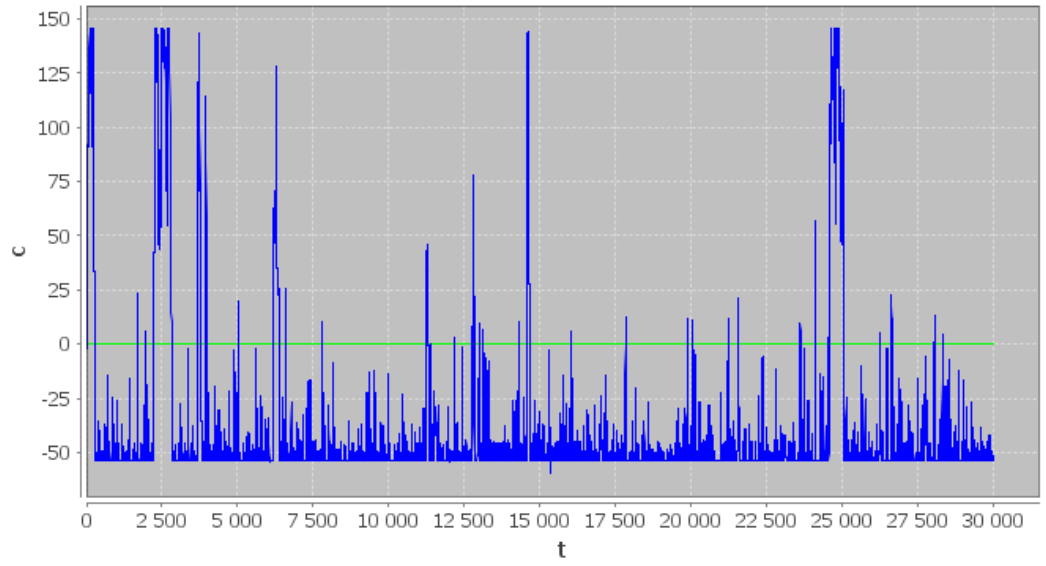

Figure 5.11: Global convergence of the system subject to 80% of false feedbacks.

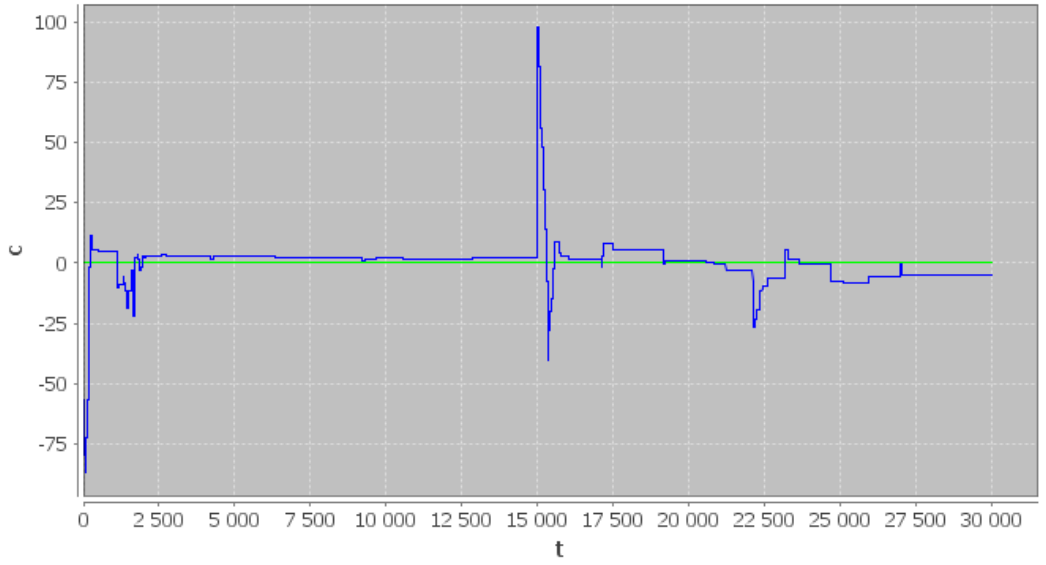

Figure 5.12: Global convergence of the system subject to changes in the values to find.

the agents are stabilizing around relevant values according to the feedbacks they receive, another one arrives contradicting all of what has been learnt before. Ultimately, the agents are not able to propose a solution relevant with the feedbacks since it would be contradictory with their knowledge.

Finally, we study the impact of a change in the values to find. At time 15000, the real values are randomly generated again and the agents are not aware of this. Their behaviours are still the same: self-tuning in reaction to operator feedbacks. We can observe the system

convergence on figure 5.12. The agents are able to converge towards some values. When they are changing, there is a leap in the convergence as the largest error might suddenly increase. But the agents are able to converge again towards the new values. This results show that the agents are able to cope with change. Indeed, at time 15, we observe a great disturbance and the parameter-agent values are far from the values to find. But the agents are able to converge towards the new values in a limited amount of time.

It is important because it can translate a new knowledge in the system or it can mean that the system is used by another operator that has not the same points of interest than the previous one. This way, MAS4AT is able to withstand novelty.

These results in the previous section demonstrate the robustness of MAS4AT when facing false feedbacks, when they are in low quantity. Indeed, the characteristic *tuningEvolution* –see chap 3.3.2.2– of the agents ensures that a unique wrong feedback from time to time is not sufficient to disturb the system. However, when facing a huge amount of false feedbacks, the system might learn wrong values. The computation of *tuningEvolution* is such that several direction indications of the values to find influence the agents in these directions. However, when facing such quantities of error, the system should be analysed in order to identify potential issues or malice.

5.5 Comparison with Gauss-Jordan Method

One of the key feature of MAS4AT is the learning of an unknown mathematical function, more specifically its parameters. To achieve this learning, the agents use equations to represent the function. When they propose a solution for the first equation, they receive a feedback indicating if this solution is above or below the real one. Then the agents tune themselves to comply with the feedback. Finally, they propose a solution for the second equation, still representing the function to learn, and the process is done again.

Ultimately, learning an unknown mathematical function with MAS4AT is equivalent to solving an equation system. In this section, we compare MAS4AT to a mathematical method, the Gauss-Jordan method. First, we detail the method and second, we describe the experimentations and results.

5.5.1 Equation Solving with Gauss-Jordan Elimination

The common definition of the Gauss-Jordan Method is: *In linear algebra, Gaussian elimination (also known as row reduction) is an algorithm for solving systems of linear equations*. It is one of the most common and known methods for the solving of equation systems with a high number of parameters and equations. It is a simple method following the manual

solving. Generally, a equation system is written as:

$$\begin{cases} w_{1,1}a_1 + w_{1,2}a_2 + \cdots + w_{1,n}a_n = b_1 \\ w_{2,1}a_1 + w_{2,2}a_2 + \cdots + w_{2,n}a_n = b_2 \\ \cdots = \\ w_{m,1}a_1 + w_{m,2}a_2 + \cdots + w_{m,n}a_n = b_m \end{cases}$$

In this system, a_i are the unknown parameter to find and $w_{m,n}$ are known coefficients. When reaching a high number of equations, it could be interesting to write it under a matrix form, $Ax = B$ with:

$$A = \begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \dots & \dots & \dots & \dots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{pmatrix}; \quad x = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix}; \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

The first step consists in transforming the matrix A into a triangular one and performing the same operations onto the lines of vector B. For example, for $i > 1$, the transformation

$$L_i = L_i - \frac{w_{i,1}}{w_{1,1}} \times L_1$$

eliminates the unknown variable a_1 in any line that is not L_1. Then the process is done again for each line until reaching a triangular matrix. Finally, it is able to solve the last equation an find the value of a_n. This value is then reported to the line L_{n-1} and so on until reaching the first line. In the end, the values of in vector B are the values of the parameters a_i.

5.5.2 Experimentations

We aim at comparing the results of the Gauss-Jordan method with the results of our agent-based system MAS4AT. Here are the processes followed for each one.

Gauss-Jordan.	MAS4AT.
1. Gauss-Jordan Elimination. 2. Propose results. 3. Receive Feedback. 4. If wrong, exclude solution. 5. Loop to 1 until right results.	1. Read the first equation. 2. Propose results for the current equation. 3. Receive Feedback. 4. Tuning of the agent according to the feedback. 5. Read the next equation. 6. Loop to 2 until right results.

When a solution is reached, we observe the number of feedbacks needed by each method to reach the right values. It is a numerical representation of the convergence of the system

towards the solution. Indeed, the more there are feedbacks, the more the system needs time to find the values. We perform the experimentation on a set of 100 equation systems divided into two groups: one with three unknown variables and the other one with 99 unknown variables. The systems are randomly generated.

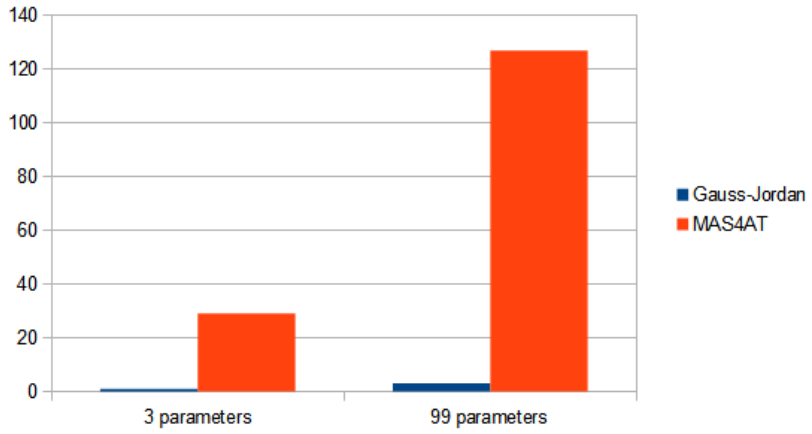

Figure 5.13: Comparison of MAS4AT learning with Gaussian Method.

Figure 5.13 presents the average number of feedbacks needed for each method and for each group. We observe that the Gauss-Jordan method requires none feedbacks when learning the values of three parameters and an average number of 3 feedbacks when confronted to 99 parameters. Indeed, most of the time, this method is able to find the right values of an equation system in a single run. On the contrary, MAS4AT requires an average number of 29 feedbacks –with a deviation of 7– when facing three parameters and 127 feedbacks –with a deviation of 31– for 99 parameters. MAS4AT requires more feedbacks than the Gaussian elimination. It can be explained by the operation of the multi-agent system. Indeed, as it takes into account the equations one by one and proposes one solution at a time, it faces more potential feedbacks. Indeed, mainly in the beginning, the values proposed by MAS4AT have a higher chance to be false since it does not integrate all the information yet –meaning it has not taken into account all the equations. However, each new equation restrains the search space and the number of feedbacks is decreasing over time.

The results give an advantage to the Gauss-Jordan solving method since it is fastest than MAS4AT. However, MAS4AT is designed to be plunged in an **dynamic environment** where all the related equations are not known beforehand but are rather derived from observations. Indeed, the equations representing the situations are progressively constructed by the entity-agents. Our system is then able to propose solutions whenever a new situations arises while the Gaussian method needs to have at least as many equation as the number of parameters in the function to learn.

Even when the system is running since enough time to perform the Gaussian elimination, each new situation modifies the built equation system and the solving has to be done again. On the contrary, MAS4AT integrates the new situations as they arrive. This can be illustrated by the number of mathematical operations performed by the two methods each time a new situation is integrated.

For the Gauss-Jordan method, the number of operation is in the range of $\frac{2}{3} \times n^3$, so it has a complexity of $O(n^3)$, for n parameters. It takes into account the multiplications to transform the matrix into a triangular one and the divisions to eliminate the variables line by line. Additions and subtractions are not taken into account since they have a little cost. As whenever a new situation arises and a new equation is built, the Gauss-Jordan algorithm has to be run again, this method executes $O(n^3)$ operations each time.

On the contrary, when MAS4AT receiving a new situation, MAS4AT performs n multiplications to compute the Entity Behaviour Value –one multiplication for each parameter- and $3 \times n$ multiplications during the tuning decision process –for the computation of *tuningEvolution*. Therefore, the complexity of MAS4AT is $O(n)$ when handling a new situation and the related equation. It is because the multi-agent system reads the equations one by one, as they arrive when the Gauss-Jordan method uses the whole equation system each time.

There is another issue with the Gauss-Jordan approach, which is a consequence of the need of the whole equation system. A surveillance system is an **open environment**, meaning that all the events are not necessarily known beforehand. Therefore, the number of parameters in the function to learn is unknown and might not be constant. In this case, the Gaussian method has to perform a new solving each time a new event is identified in a situation, providing there are enough equations in the system.

When performing the resolution of an equation system in order to learn the values of an unknown mathematical function, the Gauss-Jordan method gives better results than our agent-based model MAS4AT. However, **it requires that all the equations are known, as well as the number of parameters**. But in a surveillance system, the equations represent situations to analyse. Moreover, as a surveillance system is **open and dynamic**, the equations are built on the fly, whenever a new situation arises. In this context, the Gaussian method is not able to efficiently solve the problem since all the new equations and parameters can appear at any time.

However, MAS4AT is able to propose a solution whenever an equation is integrated. It also takes into account new parameters when they appear and it is able to learn the parameter values –i.e. to solve the equations– on the fly. Ultimately, the Gauss-Jordan method is better than MAS4AT when applied to a **close and static** system while MAS4AT performs better in a **open and dynamic** environment.

5.6 Conclusion & Perspectives

In this section, we exposed some qualitative and quantitative criteria that might be used to evaluate the multi-agent system MAS4AT. These criteria could also be classified into subjective and objective ones, respectively. Subjective because they are dependent on the domain and what we need the system to do. On the other hand, there are objective criteria that consist in measures allowing the evaluation of the system while it runs.

Then, we exposed the results of the evaluation of MAS4AT using SIM4AT. The point of these experimentations is to validate the principle as well as its running in real conditions. Starting from the learning of the parameter, the results show that **MAS4AT is able to track the correct value in a finite amount of time**. This validate the learning process. The comparison with a mathematical method shows that a common simple method is more efficient than our self-tuning based learning. However, the strength of a multi-agent system appears when facing a dynamic and open system.

Thus, MAS-based methods are well suited when applied to a surveillance system since they often exhibit these two characteristics amongst others. If the learning is a useful feature in such systems, the central point is the alert triggering. We have obtained positive results when testing MAS4AT in real conditions and for this specific aspect of the system. Indeed, **the parameter-agents are able to find the right values** for the entity-agent to compute accurate Entity Behaviour Values. These values are then used to estimate the suspicious level of an entity and raise an alert if needed.

Moreover, several operators might use a surveillance system and their knowledge and representation of the world is evolving and changing. We provide tests showing that MAS4AT is able to cope with these changes but also with errors and wrong feedbacks. **MAS4AT is able to tune its parameters according to feedbacks and in real time, even if several errors are sent to it**, or if the values to learn have changed.

Finally, we have compared the learning results of MAS4AT to the Gauss-Jordan solving method when the system has to learn the parameter values of an unknown mathematical function. This function is represented by an equation system and the Gaussian method is a classical mathematical tool to solve such a system. If MAS4AT is able to learn the right values, the Gaussian resolution is faster than our multi-agent system. Indeed, our system proceeds step by step when the GJ method is based on matrix computations. However, we identified two issues to the Gaussian solving that prevent it to be applied in a maritime surveillance system:

- The amount of computation to do is greater than MAS4AT. This can be an issue when the resources are scarce and the learning has to be done in real-time.

- The Gaussian method is not able to cope with open dynamic environments since it needs the whole equation system and a known number of parameter to be efficient.

Therefore, our system MAS4AT performs better in the context of a surveillance system. Indeed, it is able to **handle the dynamics and openness** of the system by taking into account each new situation as they arrive, and propose a solution to the related function. It also **handles new events** –and thus new parameters in the function to learn– and integrates them in its solution. Ultimately, MAS4AT has a **low cost in term of resources** and it is able to learn the parameter in **real-time**, i.e. while the system is running.

Conclusion Générale

The conclusion in English starts page 125

Une conclusion...

A L'ORIGINE de ce travail, il y avait les systèmes de surveillance. De nos jours, ils sont de plus en plus présents et de plus en plus complexes, que ce soit les zones ou les entités surveillées ou les possibilités qu'ils offrent. Nous avons identifié un point intéressant dans ce genre de système, la levée d'alerte. Actuellement, les alertes sont levées par des opérateurs humains, mais il y a plusieurs défauts et limites : le nombre croissant d'entités et d'événements possibles sont de plus en plus difficiles à prendre en compte par l'esprit humain. Et quand on pense à automatiser le système, la question du processus d'apprentissage se pose : la valeur et l'importance des événements font partie de la connaissance inconsciente des utilisateurs, de leur expérience, et il est ainsi difficile de les apprendre. De plus, le système n'est pas nécessairement connu dans son ensemble alors que les techniques d'apprentissage existantes ont besoin de jeux de test et d'entraînement précis et portant sur un environnement connu.

Dans cette thèse, nous proposons un système multi-agent générique pour la levée d'alerte. C'est une application de la théorie des AMAS, théorie basée sur la généricité, la simplicité et la distribution des tÃ¢ches. Cela garanti aussi la robustesse et le flexibilité de notre système. Dans cette conclusion, nous exposons dans un premier temps les contributions, à la fois applicatives et scientifiques, de cette thèse avant de proposer plusieurs voies pour améliorer ce travail et les axes de recherches associés.

Contributions applicatives

La contribution principale de notre recherche, dans le cadre applicatif, est la définition d'une architecture d'agents générique pour la levée d'alerte, représentée par le modèle MAS4AT (Multi-Agent System for Alert Triggering). Cette architecture a ensuite été implémenté dans un système réel. En effet, cette thèse s'inscrit dans le cadre du projet européen I2C où notre modèle est appliqué et utilisé.

Premièrement, nous avons définis une situation dans un système de surveillance. Nous sommes partis sur principe qu'une situation à un moment donné et pour une entité donnée est le résultat d'une accumulation de plusieurs événements. Ainsi, les opérateurs impliqués sont capable d'associer une importance à chaque événement identifié et ils l'utilisent ensuite pour déterminer si un situation requiert une intervention ou non. Cette importance peut être donnée par une valeur numérique, ainsi une situation peut être définie par une fonction mathématique qui cumule l'importance de tous les événements impliqués dans la dite situation. Le résultat de cette fonction peut ensuite représenter le niveau de suspicion de la situation et donc de l'entité concernée. Dans MAS4AT, chaque agent-entité est capable de représenter la situation dans laquelle il se trouve en utilisant la fonction 5.2 et donc d'évaluer son propre niveau de suspicion, sa valeur de comportement (EBV).

$$EBV^t = \sum_{i=1}^{n} e_i \times w_i^t \tag{5.2}$$

Deuxièmement, nous avons conçu des agents pour représenter et apprendre la connaissance des opérateurs qui utilisent le système de levée d'alertes. Ainsi, le système est capable d'apprendre de ses erreurs et de lever des alertes de plus en plus pertinentes. En effet, les opérateurs sont capable d'associer une importance à chaque événement mais cette importance est souvent, voire toujours issue de l'expérience et est donc relativement abstraite. Ainsi, il est difficile de demander une valeur numérique pour chaque événement. Le système MAS4AT est capable d'apprendre ces valeurs tandis qu'il fonctionne. L'apprentissage utilise des règles de coopération locales et des comportements d'agents issus de la théorie des AMAS. Dans notre système, l'apprentissage est équivalent à la résolution d'un système d'inéquations. Les résultats que nous avons obtenus valident l'utilité et les avantages de MAS4AT dans un environnement ouvert, dynamique et complexe. Ils soulignent aussi que des comportements simples et locaux peuvent permettre de résoudre des problèmes complexes pour lesquels aucune solution ou aucun modèle générique n'est connu.

En plus de MAS4AT, notre travail propose un modèle d'agent pour tester les processus d'apprentissage d'un système de surveillance. La plateforme de simulation SIM4AT propose des outils pour la création d'événements et de scénarios pour simuler les entrées d'un système réel. De plus, la plateforme est capable de simuler les utilisateurs du système ainsi que les retours qu'ils pourraient envoyer pour l'apprentissage.

Contributions scientifiques

Notre travail a aussi permis d'apporter quelques contributions scientifiques. La première est la définition d'un modèle à base d'agent selon la théorie des AMAS. Cela met en évidence l'adéquation de la théorie quand il s'agit de représenter un système complexe en utilisant des agents locaux. Et l'application à la surveillance maritime valide le principe de base de

ce domaine.

La deuxième contribution concerne la représentation d'une situation. Dans le sens commun, une *situation* est une combinaison de circonstances à un moment donné. En effet, on représente souvent une situation comme étant un ensemble d'événements et d'états, chacun étant identifié, connu et sémantiquement défini. Mais dans la majorité des système, cette connaissance complète n'est pas disponible et ce genre de représentation ne peut pas intégrer le système dans son ensemble. En conséquence, nous proposons de conserver la combinaison d'événements mais de laisser de côté la signification. Chaque événement a un agent pour le représenter et les agents sont capables de se combiner les uns avec les autres pour former une situation.

De plus, chaque agent-événement est capable de fournir une valeur numérique qui traduit sont importance. In fine, dans MAS4AT, nous proposons de représenter une situation en utilisant une valeur numérique qui est le résultat de la combinaison des événements représentés par des agents. L'organisation des agents se fait selon la fonction mathématique qui définis la dite combinaison. Cette méthode permet au système de s'affranchir de l'interprétation des événements et c'est aussi un moyen efficace pour gérer l'apparition de nouveaux événements en temps réel.

Notre troisième contribution scientifique est une conséquence directe de cette représentation à base d'agent et concerne l'apprentissage. La valeur numérique permettant d'évaluer une situation est le résultat d'une fonction mathématique. Dans cette fonction, le poids de chaque événement est connu, mais pas la valeur de l'événement. En effet, cette valeur dépend du contexte du système, de l'utilisateur,...Qui plus est, le nombre d'événements pour une situation donnée n'est pas connu à l'avance. Les systèmes existants se basent sur des données d'entraînement et effectuent des comparaisons entre ces données et les observations faites. L'état de l'art a montré le manque d'un apprentissage dynamique et en temps réel dans ce genre de système. Ainsi, nous proposons de doter nos agents d'un ensemble de capacités et d'attributs pour leur permettre de pallier à ce manque. Cet apprentissage s'inspire des techniques d'apprentissage par renforcement et les agents sont capables de modifier leurs paramètres en fonction des retours qu'ils perçoivent.

En résumé, nous proposons un modèle d'agents avec deux points d'intérêt :

- la **représentation d'une situation** par la combinaison d'événements qui sont représentés et évalués par des agents ;
- une **méthode d'apprentissage par auto-ajustement des agents** de MAS4AT et en accord avec les retours des utilisateurs.

Il faut noter que la théorie des AMAS nous a permis de nous concentrer sur les interactions et les connaissances locales des agents. Cela signifie que nous avons pu résoudre le problème complexe et dynamique qu'est la levée d'alerte en utilisant des mécanismes simples et locaux.

Perspectives applicatives

D'un point de vue applicatif, nos perspectives sont :

- **la validation de MAS4AT** lors de son intégration dans le système I2C permettant de tester l'apprentissage et la levée d'alertes en condition réelle et confrontés à des opérateurs réels ;
- nous pensons notre système suffisamment générique pour être utilisé dans d'autres domaines et il peut être intéressant d'étudier cet aspect plus avant en **appliquant MAS4AT dans d'autre système de surveillance** ;
- la représentation proposée pour uns situation peut sembler restrictive mais elle est adéquate dans le cadre d'un système de surveillance, mais nous pouvons penser à **des équations plus complexes pour représenter d'autres types de systèmes** et l'étude du comportement de MAS4AT peut être significative dans ce cas ;
- pour décider de lever une alerte, les agent-entités se réfèrent à un seuil fixe qui est un paramètre important, ainsi il faudrait mener une étude concernant **l'impact de ce seuil** sur l'apprentissage.

Perspectives scientifiques

D'un point de vue applicatif, nos perspectives sont :

- **des agents plus confiants** dans MAS4AT ; ils réagissent à des stimulus pour s'ajuster et les règles de coopérations obligent l'agent le moins contraint à effectuer un ajustement et des études en ce qui concerne les croyances de l'agent par rapport à ces contraintes peuvent être intéressantes ;
- au delà d'une confiance,les agents peuvent être plus actifs à l'attention des utilisateurs en leur **proposant des solutions** quand un cas semble insoluble par faute de données manquantes ;
- de plus, les opérateurs sont en relation directe avec les agents. Ainsi, il pourrait être intéressant d'étudier **l'impact de leur méfiance envers les agents** ; c'est un travail qui possède des implications philosophiques et sociologiques ;
- **l'amélioration du processus d'apprentissage** ; les agents sont tels qu'ils sont simples et peuvent s'ajuster en fonction de feedbacks, ce qui les rend capable d'apprendre les valeurs des paramètres d'une fonction mathématique ; il faut étudier plus avant les décisions et les actions des agents en vue d'améliorer leur apprentissage, et par la suite, soumettre le système à plus de critères d'évaluation.

Conclusion & Perspectives

« I never learned from a man who agreed with me. »

Robert A. Heinlein

THIS work started with surveillance systems. They are more and more present nowadays, and they are more and more complex, from the areas and entities they monitor to the features they exhibit. We identified a point of interest in such system: the alert triggering. It is currently performed by human operators but it shows several limits: the growing numbers of entities and the growing number of events are both hard to handle by the human spirit. And when automating the alert triggering, questions arise on the learning process: values and importance of events are an unconscious knowledge of the users, thus they are difficult to learn. Besides, the whole system is not necessarily known or represented and actual learning techniques need accurate training data and world representation, if not both.

In this thesis, we propose a generic multi-agent system for alert triggering. This is an instantiation of the AMAS theory based on genericness, simplicity and tasks distribution. This also provides robustness and flexibility to our system.

In this conclusion, we first expose the contributions of this thesis, both applicative and scientific. Then we propose several axes to improve this work and the related research. In the end, a few personal words concludes this manuscript.

Applicative Contributions

The main applicative contribution of our research is the definition of an agent generic architecture for alert triggering, MAS4A4T (Multi-Agent System for Alert Triggering). This architecture has then been instantiated in a real system. Indeed, this work also takes place into the European Project I2C where a working implementation of our model is used.

First, we designed the representation of a situation in a surveillance system. We assess

that a situation at a given time and for a given entity is the result of the accumulation of several events. Thus, the operators involved are able to associate an importance to each identified event, and they use it to determine if the situation is worth the investigation or not. This importance can be represented as a numerical value. Thus, a situation can be represented as a mathematical function cumulating the different importance of each event involved in it. The result of this function may then represent the suspicious level of the situation. In MAS4AT, each entity-agent is able to represent its current situation using a mathematical function –cf. 5.3– and to assess its own suspicious level, its Entity Behaviour Value (EBV).

$$EBV^t = \sum_{i=1}^{n} e_i \times w_i^t \tag{5.3}$$

Second, we designed agents to represent and learn the knowledge of the operators using the alert triggering system. This allows the system to learn from its errors and raise more and more relevant alerts. Indeed, the operators are able to associate an importance to each event, but this is often, not to say always, an abstract importance coming from the experience. Thus it is difficult to ask for a numerical value for each one. MAS4AT is able to learn the values of the events from the operator while running. This learning uses local cooperation rules and behaviours from the event-agents and parameter-agents, as introduced by the Adaptive Multi-Agent System Theory.

In our system, the learning is equivalent to solving a system of inequality. The results we have obtained validate the usefulness and the benefits of MAS4AT in a dynamic, open and complex environment. Finally, the results underline the fact that simple and local agent behaviours can solve complex problem where no models or generic solutions are known.

In addition to MAS4AT, our work propose an agent model to test learning processes of a surveillance system. The SIMulation for Alert Triggering (SIM4AT) is a simulation platform providing tools for the creation of events and scenarios to simulate the inputs from a real surveillance systems. SIM4AT is also able to simulate the operator user of the system as well as the feedbacks he may send.

Scientific Contributions

Our works also bring some scientific contributions. The first one is the definition of the MAS4AT agent model based on the AMAS theory. It highlights the adequacy of the theory when it comes to represent a complex system using local agents. The application in maritime surveillance validate the principle in this domain.

Second, we propose a new approach to the representation of a situation. The common sense of *situation* is the combination of circumstances at a given moment. Indeed, a situation is often represented as a set of events and states, each one being identified, known and

semantically defined. But in the majority of system, the whole knowledge of the system is not available and this kind of representation could not integrate the whole system. Thus, we propose to keep the event combination but let apart the meaning of them. Each event identified event has an agent counterpart and the agents are able to combine themselves in order to form a situation.

Moreover, each event-agent is able to provide a numerical value translating the importance of the agent. Finally, in MAS4AT, we propose to represent a situation by a numerical value resulting from the combination of event represented by agents. The agent organisation is made according to a mathematical function describing the said combination. This method allow the system to be independent from the interpretation and the meaning of the event. This is also an efficient way to handle new events as they appear without having to design a new event each time.

Third, as a direct consequence of the event-agent representation, the learning in MAS4AT is also a contribution. The numerical value assessing a situation is the result of a mathematical function. In this function, the weight of each event is known, by not the value of the events. Indeed, these values are depending on the context of the system, the user, ... And more, the number of events in a given situation is not known beforehand. Existing systems are based on training data to perform comparisons with the observations. The state of the art shows a lack in a dynamic and real-time learning in such systems. Therefore we propose skills and characteristics for our agents to make them able to learn in real-time in a dynamic system. This learning is based on reinforcement learning techniques and the agents are able to tune themselves according to feedbacks.

To sum up, we propose a agent model with two main scientific interests:

- The **representation of a situation** by the combination of events represented and valued by agents. This also allows to assess the situation.
- A **learning method by self-tuning of the agents** in MAS4AT. This tuning is set off by feedbacks from the user of the system, or by the incorporating system.

It has to be noted that the AMAS theory allowed us to focus on the local interactions and knowledge of the agents. This means that starting from the global dynamic and complex problem of alert triggering and learning, we finally solve it using simple and local mechanisms. This is all the paradox and main interest of the AMAS.

Contribution Evaluation

This work aimed to fill a gap in the existing surveillance systems. Indeed, surveillance systems are widely developed and used in the world, however the state of the art underlines several lacks. The currently used systems are designed for specific domains and contexts. They rely on the knowledge of the system they represent and on training data to be efficient.

There is no generic system and for each new one, the whole design process is to be done again from the start.

Furthermore, the majority of these systems have difficulties to face the dynamics and openness of the real world they monitor. This can be addressed at a high cost in term of resources and computation time, even in term of efficiency and accuracy.

Thus, we proposed MAS4AT, an agent model for alert triggering. The agents are fulfilled with local goals, interactions and knowledge matching the events and monitored entities in surveillance systems. MAS4AT has been designed to takes place into an existing project of maritime surveillance and the first results are very encouraging. Indeed, our system seems well suited to face the dynamics and the complexity of the real world application and representation.

Ultimately, the application of our works emphasizes our scientific contributions. This work is completed but has yet to be integrated in the real system. This bring out new challenges and several perspectives.

Applicative Perspectives

Our perspectives at the applicative level are:

- **The validation of MAS4AT** by integrating it in the I2C system and more specifically in the Behaviour Analysis component. This integration is needed to validate both the learning and the alert triggering of our system in real condition. Indeed, the confrontation with the operators and the real system is a central point of our future works. Besides, it would be interesting to conduct more studies on **the robustness of the MAS when subject to errors**.

- More than the integration and the running in the I2C system, we think our system generic enough to be used in other contexts and domains. It could be interesting to study this aspect of our system furthermore. So we would like to **apply MAS4AT to another surveillance system**.

- The representation of a situation we propose, a sum of products representing the cumulation of events, may be seen as restrictive. This is adequate for the representation of situations in a surveillance system. However, we can imagine more complex functions (integrals, differential equations,...) to use in other kinds of systems. Therefore, one possible study could be the uses and results of **MAS4AT concepts and agents using a different representation of situations and behaviours**.

- To decide if an entity-agent has to trigger an alert, it refer itself to a fixed threshold. An interesting study to come is **the impact of the threshold**. It seems that it has none on the learning process, but still, this is an important variable by its meaning.

- SIM4AT has been designed to simulate the environment of MAS4AT and test it. We have made it generic so that it is usable with other surveillance systems using the same representation of events and entities. As it is a prototype, **the extension of SIM4AT** is an interesting work. The objective is to take into account several representations without an extensive external intervention. However, SIM4AT has been useful to conduct our experimentations and the validation of our propositions.
- Both MAS4AT and SIM4AT has been designed to be used by non AMAS/MAS experts. The interactions with the users are kept simple but it would be of help to highlights the remaining issues and challenge in order to improve our system. This also constitutes a **validation of the intuitive use of the AMAS**.

Scientific Perspectives

Our perspectives at the applicative level are:

- **More confident agents**. In MAS4AT, the agents are designed to react to stimuli, the feedbacks, to tune themselves. There is cooperation as well when none of the agents reacts and they are then able to determine the less constraint among them. However, each agent is able to provide a confidence in its result. Future works will be conducted on the impact of the critical level and the tuning history of the agents.
- More than their confidence, the agents could be made more proactive towards the users. When reaching a deadlock, the agents assume that they it is their fault. However, there can be several external causes: missing data, wring observation, malignancy,... For this reason, we are interested in the **improvement of the agents to find the external solutions of encountered issues**.
- Speaking of external causes, the operators are in direct relation with the agents. Therefore, an interesting study would be **the impact of the questioning of the operator by the agents**. This work should involve philosophical and sociological works as well.
- **Improve the learning process**. The agents of MAS4AT are designed to be simple and perform a tuning according to feedbacks. With this, they are able to learn the parameter values of a mathematical function. More studies on the learning process and the agents decision and action have to be conducted in order to refine it. Besides, **more criteria should be investigating to further prove the principle of MAS4AT**, and thus helping its improvement.

Thoughts about Emergent Learning

My personal thoughts on the learning process we have imagined, designed and implemented are that it is emergent. There are several definition of emergence but they

all have in common the surprise element. Taking the dictionary definition, emergence is a *sudden appearance*. In several domains the emergence appears when several simple systems act together and a complex behaviour appears without being predictable, or hardly.

Multi-agent systems are well suited to design emergent system, and the AMAS theory is built around this concept. Local interaction and local behaviours leading to an emergent complex function to solve complex systems. In MAS4AT, the parameter-agents and the event-agents are designed according to this theory: only local knowledge and interactions. Besides, they only learn their own value and have no idea of the common objective. However, the final system is able to provide right and accurate values for each parameter of each event.

From my point of view, emergence can be found here. We do not ask the agent to learn the values of a function, we made them reacting to stimuli to adapt themselves. We have no control on how they search the value and on their decision to do so. Nevertheless, the efficiency of this learning is to be found here.

Final words

Multi-Agent System. Five years ago, these three words mean nothing to me. It took the Master degree and these four year works to begin to grasp it. Now, I think this an elegant and efficient way to solve complex problems or to build complex simulations.

Furthermore, introducing to me the notions of cooperation between the agents, my work within the SMAC team led me to an understanding of the AMAS theory. On paper, it has it all: simple local mechanisms and agent cooperation are enough to build complex systems. It is now our work, me and the other members of the team, to formalise it and make this theory a new paradigm. I have to thank my supervisors and also the other professors and PhD Students for the discovery of this new way of thinking. I am convinced it can only improve in the future.

I would just finish by something I observed and experienced myself. If cooperation leads to the emergence of new ideas, of results, it may take several faces. Indeed, cooperation is working together. Commonly, it means to put various skills together to reach a higher goal. In research, skills and knowledge are brought by the member of the research team. But I found that, sometimes, confrontation and disagreement is a better way to reach understanding and new solutions. Finally, maybe cooperation is not working together for a solution but rather let it emerge whatever the way is.

Thesaurus

Agent: An autonomous cooperative entity able to perceive its environment, decide for a course of action and act to pursue it.

Anomaly: An anomaly is something that is not expected, a deviation from the normality. It can be the consequence of a failure, of a malicious act,.... It has to be investigated in order to put the things back as normal. The section 3.3 introduces the **anomaly-agent**, which is an agent built to represent its real-world counterpart in our multi-agent system.

Common Operational Traffic Picture: A visual representation of the monitored ships for the project I2C. This also associates each ship with all the information gathered and aggregated on it.

Entity: An entity is what is monitored in a surveillance systems and it has expected behaviours. As said previously, an entity may be abstract or concrete. The section 3.3 introduces the **entity-agent**, which is an agent built to represent its real-world counterpart in our multi-agent system.

Entity Behaviour Value: The entity behaviour value is a numerical value we have introduced in order to value the observations on the monitored entities (see section 3.3.1). The higher is this value, the higher the entity has a suspicious behaviour.

Event: An event is an information on a monitored entity. This can be the addition of an information (name, identification, color, origin,...) but also a modification of the current state of the entity (stop, running, high speed, ...). Potentially, for each piece of information we can aggregate on a given entity, there is a corresponding event.

Operator: An operator is a human user of the system, considered as an expert in the corresponding domain. The operator is in charge of the analysis, or the analysis of the analysis of a situation, as well as the decision of actions to perform.

Parameter: The parameter is a notion introduced in section 3.3 and serve as an indication on a related anomaly. More precisely, for each anomaly, three parameters have been identified [Mano et al., 2010]: the *initial parameter*, called *init*, representing the

importance of the anomaly when it appears; the *increment parameter*, called *incr*, representing the growing importance of the anomaly while it lasts; the *decrement parameter*, called *decr*, representing the decreasing importance of the anomaly once it has disappeared. These are more detailed in section 3.3.1. The section 3.3 introduces the **parameter-agents**, which are agents built to represent these parameters.

Situation: A situation is a set of events at a given time and happening on one ore more entities. For example, a red ship called *Drug Dealer* has stopped in open sea next to a ship called *Drug Seller*. This situation might be worth an investigation.

Weight: The weight is a numerical value associated to an anomaly and translating its global importance according to the associated parameters and the given situation. This is detailed in section 3.3.1.

Personal Bibliography

1. Nicolas Brax, Eric Andonoff, Jean-Pierre Georgé, Marie-Pierre Gleizes and Jean-Pierre Mano. *MAS4AT : un SMA auto-adaptatif pour le déclenchement d'alertes dans le cadre de la surveillance maritime.* In Revue d'Intelligence Artificielle (RIA), vol. 3/2013, pp. xx–xx, 2013.

2. Nicolas Brax, Eric Andonoff and Marie-Pierre Gleizes. *A Self-Adaptive Multi-Agent System for Abnormal Behavior Detection in Maritime Surveillance.* In KES International Conference (KES-AMSTA), Dubrovnik, vol. 7327, pp. 174–185, 2012.

3. Nicolas Brax, Eric Andonoff, Jean-Pierre Georgé, Marie-Pierre Gleizes and Jean-Pierre Mano. *Détection de comportements illicites par SMA adaptatif : application à la surveillance maritime.* In Journés Francophones sur les Systèmes Multi-Agents (JFSMA), Cépaduès, pp. 105–114, 2011.

4. Nicolas Brax, Frédéric Amblard, Nicolas Becu, Laure Santoni, Yvon Haradji and Samuel Thiriot. *When predictive modelling meet participatory simulation: a feedback on potential and issues of a combined approach.* In MAPS2 Conference, Teachin of/with Agent-Based Models in the Social Sciences, 2010.

5. Nicolas Brax and Frédéric Amblard *A Self-Repairing Solution for the Resilience of Networks to Attacks and Failures.* In European Conference on Complex Systems, 2009.

Bibliography

AHA, D. W. 1998. Feature weighting for lazy learning algorithms. In *Feature Extraction, Construction and Selection*. Springer, 13–32.

AKAIKE, H. 1974. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on 19*, 6, 716–723.

ALIAKBARPOUR, H., KHOSHHAL, K., QUINTAS, J., MEKHNACHA, K., ROS, J., ANDERSSON, M., AND DIAS, J. 2011. Hmm-based abnormal behaviour detection using heterogeneous sensor network. *Technological Innovation for Sustainability*, p.277–285.

ALREFAEI, M. AND ANDRADÓTTIR, S. 2001. A modification of the stochastic ruler method for discrete stochastic optimization. *European Journal of Operational Research 133*, 1, 160–182.

ANDERSON, J., MICHALSKI, R., MICHALSKI, R., CARBONELL, J., AND MITCHELL, T. 1986. *Machine learning: An artificial intelligence approach*. Vol. 2. Morgan Kaufmann.

ANDLER, S. F., FREDIN, M., GUSTAVSSON, P. M., VAN LAERE, J., NILSSON, M., AND SVENSON, P. 2009. Smartracin: a concept for spoof resistant tracking of vessels and detection of adverse intentions. In *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 73050G–73050G.

ANDRADE, M., CHACON, P., MERELO, J., AND MORAN, F. 1993. Evaluation of secondary structure of proteins from uv circular dichroism spectra using an unsupervised learning neural network. *Protein Engineering 6*, 4, 383–390.

ANDRADÓTTIR, S. 1995. A stochastic approximation algorithm with varying bounds. *Operations Research*, 1037–1048.

ANDRADÓTTIR, S. 1996. A global search method for discrete stochastic optimization. *SIAM Journal on Optimization 6*, 513.

ANDRADÓTTIR, S. 1998. A review of simulation optimization techniques. In *Simulation Conference Proceedings, 1998. Winter*. Vol. 1. IEEE, 151–158.

ANSCOMBE, F. J. 1960. Rejection of outliers. *Technometrics 2*, 2, 123–146.

AOKI, K., TAKAGI, H., AND FUJIMURA, N. 1996. Interactive ga-based design support system for lighting design in computer graphics. In *Proceedings of the 4th International Conference on Soft Computing*. Vol. 2. 533–536.

ASHBY, W. 1962. Principles of the self-organizing system. *Principles of Self-organization*, 255–278.

AUSLANDER, B., GUPTA, K., AND AHA, D. 2011. A comparative evaluation of anomaly detection algorithms for maritime video surveillance. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*. Vol. vol.8019. p.3.

AUSLANDER, B., GUPTA, K. M., AND AHA, D. W. 2012. Maritime threat detection using probabilistic graphical models. In *Proceedings of the Twenty-Fifth International FLAIRS Conference*.

AXTELL, R., AXELROD, R., EPSTEIN, J., AND COHEN, M. 1996. Aligning simulation models: A case study and results. *Computational & Mathematical Organization Theory 1*, 2, 123–141.

AZADIVAR, F. 1999. Simulation optimization methodologies. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*. ACM, 93–100.

BÄCK, T., BEIELSTEIN, T., NAUJOKS, B., AND HEISTERMANN, J. 1995. Evolutionary algorithms for the optimization of simulation models using pvm. In *EURO PVM*. Citeseer.

BALCH, T. 1997. Learning roles: Behavioral diversity in robot teams. In *AAAI Workshop on Multiagent Learning*.

BALCH, T. 1998a. Behavioral diversity in learning robot teams.

BALCH, T. 1998b. Behavioral diversity in learning robot teams.

BANKS, G. 1986. Artificial intelligence in medical diagnosis: the internist/caduceus approach. *Critical reviews in medical informatics 1*, 1, 23.

BANZHAF, W. 2000. Interactive evolution. *Evolutionary Computation 1*, 228–234.

BANZHAF, W. 2009. Self-organizing systems. *Encyclopedia of Physical Science & Technology*.

BARABÁSI, A. AND ALBERT, R. 1999. Emergence of scaling in random networks. *science 286*, 5439, 509.

BARABÁSI, A., ALBERT, R., AND JEONG, H. 1999. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications 272*, 1, 173–187.

BARBARA, D., WU, N., AND JAJODIA, S. 2001. Detecting novel network intrusions using bayes estimators. In *First SIAM Conference on Data Mining*. Citeseer.

BARESEL, A., STHAMER, H., AND SCHMIDT, M. 2002. Fitness function design to improve evolutionary structural testing. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1329–1336.

BARTZ-BEIELSTEIN, T., LASARCZYK, C., AND PREUSS, M. 2005. Sequential parameter optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. Vol. 1. IEEE, 773–780.

BASU, S., BILENKO, M., AND MOONEY, R. J. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 59–68.

BATTISTELLO, G., ULMKE, M., AND KOCH, W. 2011. Knowledge-aided multisensor data fusion for maritime surveillance. In *Proceedings of SPIE*. Vol. vol.8047.

BEDAU, M. 1997. Weak emergence. *Nous 31*, 375–399.

BEKER, T. AND HADANY, L. 2002. Noise and elitism in evolutionary computation. *Soft Computing Systems-Design, Management and Applications*, 193–203.

BENVENISTE, A., PRIOURET, P., AND MÉTIVIER, M. 1990. *Adaptive algorithms and stochastic approximations*. Springer-Verlag New York, Inc.

BERNON, C., CAMPS, V., GLEIZES, M., AND PICARD, G. 2005. Engineering adaptive multi-agent systems: The adelfe methodology. *Agent-oriented methodologies*, 172–202.

BERNON, C., CHEVRIER, V., HILAIRE, V., MARROW, P., ET AL. 2006. Applications of self-organising multi-agent systems: an initial framework for comparison. *INFORMATICA-LJUBLJANA- 30,* 1, 73.

BERNON, C., GLEIZES, M., PEYRUQUEOU, S., AND PICARD, G. 2003. Adelfe: A methodology for adaptive multi-agent systems engineering. *Engineering Societies in the Agents World III*, 70–81.

BERNON, C., GLEIZES, M., PICARD, G., AND GLIZE, P. 2002. The adelfe methodology for an intranet system design. In *Proc. of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS)*. Citeseer.

BEYER, H. 1995. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation 3,* 3, 311–347.

BEYER, H. 2000. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *Computer methods in applied mechanics and engineering 186,* 2-4, 239–267.

BIANCHI, C., CIRILLO, P., GALLEGATI, M., AND VAGLIASINDI, P. 2007. Validating and calibrating agent-based models: a case study. *Computational Economics 30,* 3, 245–264.

BILCHEV, G. AND PARMEE, I. 1995. The ant colony metaphor for searching continuous design spaces. *Evolutionary Computing*, 25–39.

BILES, J. 1994. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of the International Computer Music Conference*. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 131–131.

BILES, J., ANDERSON, P., AND LOGGI, L. 1996. Neural network fitness functions for a musical iga.

BIRD, S. 1993. Toward a taxonomy of multi-agent systems. *International Journal of Man-Machine Studies*.

BOLTON, R. J., HAND, D. J., ET AL. 2001. Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, 235–255.

BONABEAU, E. 2002. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America 99,* Suppl 3, 7280.

BONABEAU, E. AND DESSALLES, J. 2011. Detection and emergence. *Arxiv preprint arXiv:1108.4279*.

BONABEAU, E., DORIGO, M., AND THERAULAZ, G. 1999. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford University Press, USA.

BOND, A. AND GASSER, L. 1988. Distributed artificial intelligence. *Communication, Co*.

BONJEAN, N. 2009. Ingenierie des systemes multi-agents adaptatifs : vers un guide pour la conception du comportement d'agent cooperatif. M.S. thesis, Universite Paul Sabatier.

BONJEAN, N., BERNON, C., AND GLIZE, P. 2009. Engineering development of agents using the cooperative behaviour of their components. *MAS&S@ MALLOW 9*, 113.

BORIAH, S., CHANDOLA, V., AND KUMAR, V. 2008. Similarity measures for categorical data: A comparative evaluation. *red 30,* 2, 3.

BOUMA, W., FUDOS, I., HOFFMANN, C., CAI, J., AND PAIGE, R. 1995. Geometric constraint solver. *Computer-Aided Design 27,* 6, 487–501.

BOUSQUET, F. AND LE PAGE, C. 2004. Multi-agent simulations and ecosystem management: a review. *Ecological modelling 176,* 3-4, 313–332.

BOUTILIER, C. 1996. Learning conventions in multiagent stochastic domains using likelihood estimates. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 106–114.

BOUZY, B. 2007. Apprentissage multi-agent. Communication personnelle - 2007.

BRAX, N., ANDONOFF, E., GLEIZES, M., AND GLIZE, P. 2013a. Mas4at : un sma auto-adaptatif pour le déclenchement d'alertes dans le cadre de la surveillance maritime. *Revue d'Intelligence Artificielle (à paraître)*.

BRAX, N., ANDONOFF, E., GLEIZES, M., AND GLIZE, P. 2013b. Self-adapted aided decision-making: Application to maritime surveillance. In *5th International Conference on Agents and Artificial Intelligence (forthcoming)*.

BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., AND SANDER, J. 2000. Lof: identifying density-based local outliers. In *ACM Sigmod Record*. Vol. 29. ACM, 93–104.

BURKS, A. 1970. *Essays on cellular automata*. University of Illinois Press.

BUTLER, D. AND WINNE, P. 1995. Feedback and self-regulated learning: A theoretical synthesis. *Review of educational research 65*, 3, 245.

BUXTON, H. 2003. Learning and understanding dynamic scene activity: a review. *Image and vision computing 21*, 1, 125–136.

CAI, Y. AND FRANCO, R. D. M. 2009. Interactive visualization of network anomalous events. In *Computational Science–ICCS 2009*. Springer, 450–459.

CALVEZ, B. 2007. Le calibrage de modeles a base d'agents pour la simulation de systemes complexes. Ph.D. thesis, Universite d'Evry-Val d'Essonne.

CALVEZ, B. AND HUTZLER, G. Automatic tuning of agent-based models using genetic algorithms. *Multi-Agent-Based Simulation VI*, 41–57.

CALVEZ, B. AND HUTZLER, G. 2005. Parameter space exploration of agent-based models. In *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 633–639.

CALVEZ, B., HUTZLER, G., ET AL. 2005. Exploration de l'espace de paramètres d'un modèle à base d'agents.

CAMPS, V. AND GLEIZES, M. 1996. Attitudes cooperatives individuelles pour l'adaptation collective. *Journees Francophones IAD SMA, Port Camargue 1*, 04, 1996–03.

CAPERA, D., GEORGÉ, J., GLEIZES, M., AND GLIZE, P. 2003. The amas theory for complex problem solving based on self-organizing cooperative agents. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*. IEEE, 383–388.

CAPERA, D., GLEIZES, M.-P., AND GLIZE, P. 2004. Mechanism type synthesis based on self-assembling agents. *Journal of Applied Artificial Intelligence 18*, Numbers 9-10 (octobre), 921–936.

CARPENTER, J., CLIFFORD, P., AND FEARNHEAD, P. 1999. Improved particle filter for nonlinear problems. In *Radar, Sonar and Navigation, IEE Proceedings-*. Vol. 146. IET, 2–7.

CARUANA, R. AND NICULESCU-MIZIL, A. 2004. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 69–78.

CELIK, M., DADASER-CELIK, F., AND DOKUZ, A. 2011a. Anomaly detection in temperature data using dbscan algorithm. In *Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, p.91–95.

CELIK, M., DADASER-CELIK, F., AND DOKUZ, A. 2011b. Anomaly detection in temperature data using dbscan algorithm. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*. IEEE, 91–95.

CHAIB-DRAA, B., JARRAS, I., AND MOULIN, B. 2001. Systemes multiagents: principes generaux et applications. *Principes et architectures des systemes multi-agents 1*, ? plop.

CHANDOLA, V., BANERJEE, A., AND KUMAR, V. 2009a. Anomaly detection: A survey. *ACM Computing Surveys (CSUR) 41*, 3, 15.

CHANDOLA, V., BANERJEE, A., AND KUMAR, V. 2009b. Anomaly detection: A survey. *ACM Computing Surveys (CSUR) 41*, 3, 15.

CHANDY, K. AND LAMPORT, L. 1985. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems (TOCS) 3*, 1, 63–75.

CHANG, Y.-H., HO, T., AND KAELBLING, L. P. 2004. All learning is local: Multi-agent learning in global reward games.

CHEN, X. AND WHITE, H. 1998. Nonparametric adaptive learning with feedback. *Journal of Economic Theory 82*, 1, 190–222.

CHIABERGE, M., MERELO, J., REYNERI, L., PRIETO, A., AND ZOCCA, L. 1994. A comparison of neural networks, linear controllers, genetic algorithms and simulated annealing for real time control. In *Proceedings of the European Symposium on Artificial Neural Networks*. Citeseer, 205–210.

CLAISSE, S., MOREL, M., AND ORMSBY, W. 2010. Enhanced sismaris:(système d'information et de surveillance maritime etendu pour l'identification des comportements suspects): Enhanced maritime surveillance and information system for abnormal behaviour identification. In *Waterside Security Conference (WSS), 2010 International*. IEEE, 1–5.

COHEN, C. J., MORELLI, F., AND SCOTT, K. A. 2008. A surveillance system for the recognition of intent within individuals and crowds. In *Technologies for Homeland Security, 2008 IEEE Conference on*. IEEE, 559–565.

COHN, D. A., GHAHRAMANI, Z., AND JORDAN, M. I. 1996. Active learning with statistical models. *CoRR cs.AI/9603104*.

COLLINS, R. T., LIPTON, A., KANADE, T., FUJIYOSHI, H., DUGGINS, D., TSIN, Y., TOLLIVER, D., ENOMOTO, N., HASEGAWA, O., BURT, P., ET AL. 2000. *A system for video surveillance and monitoring*. Vol. 102. Carnegie Mellon University, the Robotics Institute Pittsburg.

COULOM, R. 2002. Reinforcement learning using neural networks, with applications to motor control.

CRAM, D. 2010. Découverte interactive et complète de chroniques: application à la co-construction de connaissances à partir de traces. Ph.D. thesis, Thèse de doctorat en informatique, Université Claude Bernard Lyon 1.

CUPPENS, F. 2001. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference*. Vol. 32.

DALPIAZ, F., CHOPRA, A. K., MYLOPOULOS, J., AND GIORGINI, P. 2011. From intentions to social commitments: Adaptation in multiagent systems. *Knowing, Reasoning, and Acting: Essays in Honour of Hector J. Levesque 1*.

DARLEY, V. 1994. Emergent phenomena and complexity. In *Artificial Life IV*. 411–416.

DAS, K. AND SCHNEIDER, J. 2007. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 220–229.

DASGUPTA, D. 1999. Immunity-based intrusion detection system: a general framework. In *Proc. of the 22nd NISSC*. Vol. 1. 147–160.

D'ATHÈNES, A., CARRIÈRE, J., MASSONIE, B., AND APOLLODORE. 1991. *La" Bibliothèque" d'Apollodore*. Université de Besançon.

DAUTENHAHN, K. 1995. Getting to know each other–artificial social intelligence for autonomous robots. *Robotics and autonomous systems 16*, 2-4, 333–356.

DAVIDSSON, P., JOHANSSON, S., PERSSON, J., AND WERNSTEDT, F. 2003. Agent-based approaches and classical optimization techniques for dynamic distributed resource allocation: A preliminary study. In *AAMAS, 2003 workshop on Representations and Approaches for Time-Critical Decentralized Resource Role Task Allocation*.

DAVIDSSON, P. AND WERNSTEDT, F. 2002. A multi-agent system architecture for coordination of just-in-time production and distribution. *The Knowledge Engineering Review 17*, 04, 317–329.

DAWKINS, R. 2003. The evolution of evolvability. *On growth, form and computers*, 239.

DCNS. 2010. Description of work.

DE JONG, K. 1985. Genetic algorithms: A 10 year perspective. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*. Vol. 1. 9–177.

DEGROOT, M. 2004. *Optimal statistical decisions*. Vol. 82. Wiley-IEEE.

DEGUET, J., DEMAZEAU, Y., AND MAGNIN, L. 2006a. Elements about the emergence issue: A survey of emergence definitions. *ComPlexUs 3,* 1-3, 24–31.

DEGUET, J., DEMAZEAU, Y., AND MAGNIN, L. 2006b. Elements about the emergence issue: A survey of emergence definitions. *ComPlexUs 3,* 1, 24–31.

DEL CASTILLO, E. 2006. *Bayesian process monitoring, control and optimization*. Chapman & Hall/CRC.

DEMAZEAU, Y. 1995. From interactions to collective behaviour in agent-based systems. In *In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo*. Citeseer.

DENNING, D. E. 1987. An intrusion-detection model. *Software Engineering, IEEE Transactions on* 2, 222–232.

DI LALLO, A., FARINA, A., FULCONI, R., STILE, A., TIMMONERI, L., AND VIGILANTE, D. 2006. A real time test bed for 2d and 3d multi-radar tracking and data fusion with application to border control. In *Radar, 2006. CIE'06. International Conference on*. IEEE, 1–6.

DI MARZO SERUGENDO, G., GLEIZES, M., AND KARAGEORGOS, A. 2005. Self-organization in multi-agent systems. *The Knowledge Engineering Review 20,* 2, 165–189.

DI MARZO SERUGENDO, G., GLEIZES, M.-P., AND KARAGEORGOS, A., Eds. 2011. *Self-organising Software – From Natural to Artificial Adaptation*. Natural Computing Series. Springer, http://www.springerlink.com.

DIEHL, C. P., HAMPSHIRE, J., ET AL. 2002. Real-time object classification and novelty detection for collaborative video surveillance. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*. Vol. 3. IEEE, 2620–2625.

DIGINEXT. Stradivarius. European Project.

DORIGO, M. AND DI CARO, G. 1999. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 2. IEEE.

DOUSSON, C. AND GHALLAB, M. 1994. Suivi d'evolutions et reconnaissance de chroniques.

DOUSSON, C. AND LE MAIGAT, P. 2007. Chronicle recognition improvement using temporal focusing and hierarchization. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. 324–329.

DOZIER, G. 2001. Evolving robot behavior via interactive evolutionary computation: From real-world to simulation. In *Proceedings of the 2001 ACM symposium on Applied computing*. ACM, 340–344.

DRACHMANN, A. AND DRACHMANN, A. 1963. *The mechanical technology of Greek and Roman antiquity: a study of the literary sources*. Munksgaard.

DROGOUL, A. 1993. De la simulation multi-agents à la résolution collective de problèmes. *Une étude de l'émergence de structures d'organisation dans les systèmes multi-agent. Paris, Paris VI*.

DROGOUL, A. AND FERBER, J. 1994. Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies. *Artificial Social Systems*, 2–23.

DUBOZ, R. 2004. Intégration de modeles hétérogenes pour la modélisation et la simulation de systemes complexes. *Application à la modélisation multiéchelles en écologie marine*.

DUDA, R. O., HART, P. E., AND STORK, D. G. 2012. *Pattern classification*. Wiley-interscience.

EDMONDS, B. AND BRYSON, J. 2004. The insufficiency of formal design methods the necessity of an experimental approach-for the understanding and control of complex mas. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. IEEE Computer Society, 938–945.

ERCEAU, J. AND FERBER, J. 1991. L'intelligence artificielle distribuée. *Recherche* 233, 750–758.

ERTÖZ, L., STEINBACH, M., AND KUMAR, V. 2003. Finding topics in collections of documents: A shared nearest neighbor approach. *Clustering and Information Retrieval 11*, 83–103.

ESKIN, E., ARNOLD, A., PRERAU, M., PORTNOY, L., AND STOLFO, S. 2002. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*. Springer, 77–101.

FAUSETT, L. 1994a. *Fundamentals of neural networks*. Vol. 1. Prentice-Hall Englewood Cliffs, NJ.

FAUSETT, L. V. 1994b. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall Englewood Cliffs.

FEHLER, M., KLUGL, F., AND PUPPE, F. 2004. Techniques for analysis and calibration of multi-agent simulations. In *ESAW*. Springer, 305–321.

FEHLER, M., KLÜGL, F., AND PUPPE, F. 2005. Techniques for analysis and calibration of multi-agent simulations. *Engineering Societies in the Agents World V*, 898–898.

FERBER, J. 1995. Les sma: vers une intelligence collective. *Inter Editions, Paris*.

FERBER, J. 1999. Multi-agent systems: An introduction to distributed artificial intelligence.

FERBER, J., GUTKNECHT, O., AND MICHEL, F. 2004. From agents to organizations: an organizational view of multi-agent systems. *Agent-Oriented Software Engineering IV*, 443–459.

FERBER, J., MICHEL, F., AND BÁEZ, J. 2005. Agre: Integrating environments with organizations. In *Environments for multi-agent systems*. Springer, 48–56.

FILIPIC, B. AND JURICIC, D. 1993. An interactive genetic algorithm for controller parameter optimization. In *International Conference on Artificial Neural Nets and Genetic Algorithms*. 458–462.

FINLAY-MORREALE, H., LOUIE, C., AND TOY, P. 2008. Computer-generated automatic alerts of respiratory distress after blood transfusion. *Journal of the American Medical Informatics Association 15,* 3, 383–385.

FOK, V. AND LINGARD, D. 2011. Using a genetic algorithm to optimise maritime surveillance performed by space-based sensors. In *National Committee for Space Science & National Space Society of Australia, Barry Drive, Australian National University, Canberra, ACT*.

FOX, B. AND HEINE, G. 1995. Probabilistic search with overrides. *The Annals of Applied Probability 5,* 4, 1087–1094.

FRANKLIN, S. AND GRAESSER, A. 1997. Is it an agent, or just a program?: A taxonomy for autonomous agents. *Intelligent Agents III Agent Theories, Architectures, and Languages*, 21–35.

FU, M., ANDRADÓTTIR, S., CARSON, J., GLOVER, F., HARRELL, C., HO, Y., KELLY, J., AND ROBINSON, S. 2000. Integrating optimization and simulation: research and practice. In *Proceedings of the 32nd conference on Winter simulation*. Society for Computer Simulation International, 610–616.

GARCIA, F., GINNOW-MERKERT, H., ANDERSON, P., LINDE, D., AND HUDSON, B. 1986. Glucose medical monitoring system. US Patent 4,627,445.

GARCIA-TEODORO, P., DIAZ-VERDEJO, J., MACIA-FERNANDEZ, G., AND VAZQUEZ, E. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security 28,* 1-2, 18–28.

GEMAN, S., BIENENSTOCK, E., AND DOURSAT, R. 1992. Neural networks and the bias/variance dilemma. *Neural computation 4,* 1, 1–58.

GEORGE, J. 2004. Resolution de problemes par emergence. Ph.D. thesis, Universite Toulouse III - Paul Sabatier, Route de Narbonne, Toulouse.

GEORGÉ, J., EDMONDS, B., AND GLIZE, P. 2004. Making self-organising adaptive multiagent systems work. *Methodologies and Software Engineering for Agent Systems*, 321–340.

GEORGE, J., PEYRUQUEOU, S., REGIS, C., AND GLIZE, P. 2009. Experiencing self-adaptive mas for real-time decision support systems. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*. Springer, 302–309.

GEORGE, J.-P., EDMONDS, B., AND GLIZE, P. 2004. Making self-organising adaptive multiagent systems work. In *Methodologies and Software Engineering for Agent Systems*, F. Bergenti, M.-P. Gleizes, and F. Zombonelli, Eds. Kluwer, http://www.wkap.nl/, 319–338.

GEORGE, J.-P., GLEIZES, M.-P., AND GLIZE, P. 2003. Conception de systemes adaptatifs a fonctionnalite émergente : la theorie des amas. *Revue d'Intelligence Artificielle 17,* 4/2003, 591–626.

GERHARDT, L. 1969. Self-organizing system. US Patent 3,435,422.

GHOSH, A. K., WANKEN, J., AND CHARRON, F. 1998. Detecting anomalous and unknown intrusions against programs. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*. IEEE, 259–267.

GINSBERG, M. 1988. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational intelligence 4,* 3, 265–316.

GLIZE, P., GLEIZES, M.-P., AND CAMPS, V. 1998. Une théorie de l'apprentissage fondée sur l'auto-organisation par coopération. In *Apprentissage : des principes naturels aux méthodes artificielles* , Ritschard, Berchtold, Duc, and Zighed, Eds. Hermes, Paris, 117–131.

GMYTRASIEWICZ, P. J. 1992. A decision-theoretic model of coordination and communication in autonomous systems.

GOLDBERG, D. 1989. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley.

GOLDBERG, D., DEB, K., AND CLARK, J. 1992. Genetic algorithms, noise, and the sizing of populations. *COMPLEX SYSTEMS-CHAMPAIGN- 6*, 333–333.

GOLDSTEIN, I. AND PAPERT, S. 1977. Artificial intelligence, language, and the study of knowledge. *Cognitive Science 1,* 1, 84–123.

GOLDSTEIN, J. 1999. Emergence as a construct: History and issues. *Emergence 1,* 1, 49–72.

GONZALEZ, F., DASGUPTA, D., AND KOZMA, R. 2002. Combining negative selection and classification techniques for anomaly detection. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Vol. 1. IEEE, 705–710.

GORI, M. AND TESI, A. 1992. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14,* 1, 76–86.

GRAF, J. AND BANZHAF, W. 1995. An expansion operator for interactive evolution. In *Evolutionary Computation, 1995., IEEE International Conference on*. Vol. 2. IEEE, 798–802.

GRASSÉ, P. 1959a. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux 6,* 1, 41–80.

GRASSÉ, P.-P. 1959b. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux 6,* 1, 41–80.

GRIMM, V., BERGER, U., BASTIANSEN, F., ELIASSEN, S., GINOT, V., GISKE, J., GOSS-CUSTARD, J., GRAND, T., HEINZ, S., HUSE, G., ET AL. 2006. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling 198,* 1-2, 115–126.

GUERRIERO, M., WILLETT, P., CORALUPPI, S., AND CARTHEL, C. 2008. Radar/ais data fusion and sar tasking for maritime surveillance. In *Information Fusion, 2008 11th International Conference on*. IEEE, 1–5.

GUPTA, K., AHA, D., AND MOORE, P. 2009. Case-based collective inference for maritime object classification. In *Proceedings of the Eighth International Conference on Case-Based Reasoning*. Springer, p.443–449.

GUPTA, S. AND HOSSAIN, L. 2011. Towards near-real-time detection of insider trading behaviour through social networks. *Computer Fraud & Security vol.2011-1*, p.7–16.

HAAS, J., PEYSAKHOV, M., AND MANCORIDIS, S. 2005. Ga-based parameter tuning for multi-agent systems. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, 1086.

HADDOCK, J. AND MITTENTHAL, J. 1992. Simulation optimization using simulated annealing. *Computers & industrial engineering 22,* 4, 387–395.

HALL, D. L. AND LLINAS, J. 1997. An introduction to multisensor data fusion. *Proceedings of the IEEE 85,* 1, 6–23.

HE, H., WANG, J., GRACO, W., AND HAWKINS, S. 1997. Application of neural networks to detection of medical fraud. *Expert Systems with Applications 13,* 4, 329–336.

HECKERMAN, D. ET AL. 1998. A tutorial on learning with bayesian networks. *Nato Asi Series D Behavioural And Social Sciences 89*, 301–354.

HEINZE, C. 2004. Modelling intention recognition for intelligent agent systems.

HENOCQUE, Y. AND LAFON, X. 2011. Eu's strategy on maritime & environmental issues in the four seas: multilateral approaches in the baltic, black, caspian & mediterranean seas. In *The EU and Europe's Sea Basins, Tallinn*.

HERDY, M. 1996. Evolution strategies with subjective selection. *Parallel Problem Solving from Nature,PPSN IV*, 22–31.

HIMMELSPACH, J., ROHL, M., AND UHRMACHER, A. 2003. Simulation for testing software agents-an exploration based on james. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*. Vol. 1. IEEE, 799–807.

HINTERDING, R., MICHALEWICZ, Z., AND EIBEN, A. 1997. Adaptation in evolutionary computation: A survey. In *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 65–69.

HINTON, G. E. AND SEJNOWSKI, T. J. 1999. *Unsupervised learning: foundations of neural computation*. The MIT press.

HOFFMANN, C. AND JOAN-ARINYO, R. 2005. A brief on constraint solving. *Computer-Aided Design and Applications 2*, 5, 655–663.

HOGG, D. 1983. Model-based vision: a program to see a walking person. *Image and vision computing 1*, 1, 5–20.

HOLLAND, J. 1975. Adaptation in natural and artificial systems. *Ann Arbor MI: University of Michigan Press*.

HOLLAND, J. 1992. *Adaptation in natural and artificial systems*. MIT press Cambridge, MA.

HORNIK, K. 1993. Some new results on neural network approximation. *Neural Networks 6*, 8, 1069–1072.

INCE, A., TOPUZ, E., AND PANAYIRCI, E. 1999. *Principles of integrated maritime surveillance systems*. Vol. 527. Springer.

IVANOV, Y., STAUFFER, C., BOBICK, A., AND GRIMSON, W. 1999. Video surveillance of interactions. In *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*. IEEE, 82–89.

JAIN, A. K. AND DUBES, R. C. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.

JAKOB, M., VANĚK, O., URBAN, Š., BENDA, P., AND PĚCHOUČEK, M. 2010. Agentc: agent-based testbed for adversarial modeling and reasoning in the maritime domain. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 1641–1642.

JAKOB, M., VANEK, O., URBAN, Š., BENDA, P., AND PECHOUCEK, M. 2010a. Agentc: Agent-based testbed for adversarial modeling and reasoning in the maritime domain. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)-demotrack*.

JAKOB, M., VANEK, O., URBAN, Š., BENDA, P., AND PECHOUCEK, M. 2010b. Employing agents to improve the security of international maritime transport.

JAKUBEK, S. AND STRASSER, T. 2002. Fault-diagnosis using neural networks with ellipsoidal basis functions. In *American Control Conference, 2002. Proceedings of the 2002*. Vol. 5. IEEE, 3846–3851.

JANAKIRAM, D., ADI MALLIKARJUNA REDDY, V., AND PHANI KUMAR, A. 2006. Outlier detection in wireless sensor networks using bayesian belief networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*. IEEE, 1–6.

JANGAL, F., GEORGÉ, J., BONNOT, A., GIRAUD, M., MOREL, M., NAPOLI, A., AND LITTAYE, A. 2009. Toward a complete system for surveillance of the whole eez: Scanmaris and associated projects. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*. IEEE, 1–4.

JANSEN, T. AND WIEGAND, R. P. 2003. Exploring the explorative advantage of the cooperative coevolutionary (1+1) ea. In *Genetic and Evolutionary Computation–GECCO 2003*. Springer, 310–321.

JEFFREYS, H. 1998. *Theory of probability*. Oxford University Press, USA.

JENSEN, F. 1996. *An introduction to Bayesian networks*. Vol. 74. UCL press London.

JIN, Y. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications 9*, 1, 3–12.

JOLLY, K., RAVINDRAN, K., VIJAYAKUMAR, R., AND SREERAMA KUMAR, R. 2007. Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks. *Robotics and Autonomous Systems 55*, 7, 589–596.

JUNG, J. AND REGGIA, J. 2009. Evolving an autonomous agent for non-markovian reinforcement learning. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 971–978.

JURAFSKY, D. AND JAMES, H. 2000. Speech and language processing an introduction to natural language processing, computational linguistics, and speech.

KADDOUM, E., RAIBULET, C., GEORGÉ, J.-P., PICARD, G., AND GLEIZES, M.-P. 2010. Criteria for the evaluation of self-* systems (regular paper). In *Workshop on Software*

Engineering for Adaptive and Self-Managing Systems (at ICSE 2010), Cape Town, South Africa, 03/05/2010-04/05/2010. ACM, http://www.acm.org/, 29–38.

KAELBLING, L. P., LITTMAN, M. L., AND MOORE, A. W. 1996. Reinforcement learning: A survey. *arXiv preprint cs/9605103*.

KANAL, L. AND KUMAR, V. 1988. *Search in artificial intelligence*. Springer-Verlag.

KATO, Y., YAIRI, T., AND HORI, K. 2001. Integrating data mining techniques and design information management for failure prevention. In *New Frontiers in Artificial Intelligence*. Springer, 475–480.

KAWATO, M. 1990. Feedback-error-learning neural network for supervised motor learning. *Advanced neural computers 6*, 3, 365–372.

KAZEMI, S., ABGHARI, S., LAVESSON, N., JOHNSON, H., AND RYMAN, P. 2013. Open data for anomaly detection in maritime surveillance. *Expert Systems with Applications*.

KENNEDY, J. 2006. Swarm intelligence. *Handbook of Nature-Inspired and Innovative Computing*, 187–219.

KESANIEMI, J., KATASONOV, A., AND TERZIYAN, V. 2009. An observation framework for multi-agent systems. In *Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems-Volume 00*. IEEE Computer Society, 336–341.

KIEFER, P. AND SCHLIEDER, C. 2007. Exploring context-sensitivity in spatial intention recognition. In *Intl. Workshop on Behavioral Monitoring and Interpretation, TZI-Bericht*. Vol. 42. 102–116.

KIEFER, P. AND STEIN, K. 2008. A framework for mobile intention recognition in spatially structured environments. In *2nd Workshop on Behavior Monitoring and Interpretation (BMI08)*. Citeseer, 28–41.

KIM, J. AND BENTLEY, P. 1999. The artificial immune model for network intrusion detection. In *7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany*. Vol. 158. Citeseer.

KLEIN, F., BOURJOT, C., AND CHEVRIER, V. 2005. Dynamical design of experiment with mas to approximate the behavior of complex systems.

KLÜGL, F., OECHSLEIN, C., PUPPE, F., DORNHAUS, A., ET AL. 2002. Multi-agent modelling in comparison to standard modelling. In *AIS*. Citeseer, 105–110.

KOHAVI, R. AND JOHN, G. 1995. Automatic parameter selection by minimizing estimated error. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer, 304–312.

KOJIMA, K. AND ITO, K. 1999. Autonomous learning of novel patterns by utilizing chaotic dynamics. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*. Vol. 1. IEEE, 284–289.

KOMENDA, A., VOKRINEK, J., CAP, M., AND PECHOUCEK, M. 2013. Developing multiagent algorithms for tactical missions using simulation. *IEEE Intelligent Systems 28 (1)*, 42 – 49.

KONONENKO, I. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine 23,* 1, 89–109.

KOUTKIAS, V., CHOUVARDA, I., AND MAGLAVERAS, N. 2002. Agent-based monitoring and alert generation for a home care telemedicine system. In *Proceedings of the AMIA Symposium*. American Medical Informatics Association, 395.

KRAIMAN, J. B., AROUH, S. L., AND WEBB, M. L. 2002. Automated anomaly detection processor. In *AeroSense 2002*. International Society for Optics and Photonics, 128–137.

KRAPIVSKY, P. AND REDNER, S. 2001. Organization of growing random networks. *Physical Review E 63,* 6, 066123.

KUBÍ, A. 2003. Toward a formalization of emergence. *Artificial Life 9,* 1, 41–65.

KUSHNER, H. AND YANG, J. 1993. Stochastic approximation with averaging of the iterates: Optimal asymptotic rate of convergence for general processes. *SIAM Journal on Control and Optimization 31*, 1045.

KUSHNER, H. AND YIN, G. 1997. Stochastic approximation algorithms and applications.

LANGTON, C. 1990. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena 42,* 1-3, 12–37.

LAUER, M. AND RIEDMILLER, M. 2000. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer.

LEE, W., STOLFO, S. J., AND MOK, K. W. 2000. Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review 14,* 6, 533–567.

LEMOUZY, S., CAMPS, V., AND GLIZE, P. 2011. Principles and properties of a mas learning algorithm: a comparison with standard learning algorithms. In *International Conference on Intelligent Agent Technology (IAT)*. Conference Publishing Services, p.228–235.

LERICHE, S. 2006. Architectures à composants et agents pour la conception d'applications réparties adaptables. *Toulouse, Thèse de doctorat*.

LI, X., XUE, Y., CHEN, Y., AND MALI, B. 2011. Context-aware anomaly detection for electronic medical record systems. In *Proceedings of the 2nd USENIX Conference on Health Security and Privacy*. USENIX Association, p.8–8.

LICHBACH, M. I. 1996. *The cooperator's dilemma*. University of Michigan Press.

LIU, J. 2001. *Autonomous agents and multi-agent systems: explorations in learning, self-organization, and adaptive computation*. World Scientific Pub Co Inc.

LOU, J.-G., LIU, Q.-F., TAN, T.-N., AND HU, W.-M. 2003. 3-d model based visual traffic surveillance. *Acta Automatic Sinica 29,* 3, 434–449.

LUND, H. AND MIGLINO, O. 1998. Evolving and breeding robots. In *Evolutionary Robotics*. Springer, 192–210.

LUND, H., MIGLINO, O., PAGLIARINI, L., BILLARD, A., AND IJSPEERT, A. 1998. Evolutionary robotics-a children's game. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 154–158.

LUX, T. AND MARCHESI, M. 1999. Scaling and criticality in a stochastic multi-agent model of a financial market. *Nature 397,* February, 498–500.

MACREADY, W., BIENIAWSKI, S., AND WOLPERT, D. 2004. Adaptive multi-agent systems for constrained optimization. *Submitted to AAAI*.

MAHONEY, M. V. AND CHAN, P. K. 2002. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 376–385.

MAILLER, R. AND LESSER, V. 2004. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. IEEE Computer Society, 438–445.

MAMEI, M., MENEZES, R., TOLKSDORF, R., AND ZAMBONELLI, F. 2006. Case studies for self-organization in computer science. *Journal of Systems Architecture 52,* 8, 443–460.

MANEVITZ, L. AND YOUSEF, M. 2000. Learning from positive data for document classification using neural networks. In *Proceedings of the 2nd Bar-Ilan Workshop on Knowledge Discovery and Learning*.

MANO, J., GEORGE, J., AND GLEIZES, M. 2010. Adaptive multi-agent system for multi-sensor maritime surveillance. *Advances in Practical Applications of Agents and Multiagent Systems 70*, 285–290.

MATHIEU, P., ROUTIER, J., AND SECQ, Y. 2002. Dynamic organization of multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. ACM, 451–452.

MATTINGLEY, J. AND BOYD, S. 2009. Automatic code generation for real-time convex optimization. *Convex optimization in signal processing and communications*.

MCLACHLAN, G. J. 2004. *Discriminant analysis and statistical pattern recognition*. Vol. 544. Wiley-Interscience.

MEHRAN, R., OYAMA, A., AND SHAH, M. 2009a. Abnormal crowd behavior detection using social force model. IEEE Conference on Computer Vision and Pattern Recognition.

MEHRAN, R., OYAMA, A., AND SHAH, M. 2009b. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 935–942.

MEHROTRA, K., MOHAN, C. K., AND RANKA, S. 1997. *Artificial Neural Networks*. the MIT Press.

METROPOLIS, N. 1987. The beginning of the monte carlo method. *Los Alamos Science special issue 15*, 125–130.

METROPOLIS, N. AND ULAM, S. 1949. The monte carlo method. *Journal of the American Statistical Association 44,* 247 (September), 335–341.

MILLER, R. A. 1994. Medical diagnostic decision support systems - past, present, and future a threaded bibliography and brief commentary. *Journal of the American Medical Informatics Association 1,* 1, 8–27.

MILLER, W., SUTTON, R., AND WERBOS, P. 1995. *Neural networks for control*. MIT Press (MA).

MINAR, N. AND SANTA FE INSTITUTE (SANTA FE, N. 1996. The swarm simulation system: A toolkit for building multi-agent simulations. Santa Fe Institute Santa Fe, NM.

MITCHELL, T. 1997. Machine learning. *Mac Graw Hill*, 368.

MODI, P., SHEN, W., TAMBE, M., AND YOKOO, M. 2005. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence 161,* 1-2, 149–180.

MOREL, M. AND BROUSSOLLE, J. 2011. I2c, interoperable sensors & information sources for common detection of abnormal vessel behaviours and collaborative suspect events analysis. In *MAST 2011 Conference Session. MAST Events Ltd, Marseille, France*.

MOREL, M. AND CLAISSE, S. 2010a. Integrated system for interoperable sensors & information sources for common abnormal vessel behaviour detection & collaborative identification of threat (i2c). *Proc. of the Ocean and Coastal Observation: sensors and observing systems, numerical models and information systems, Brest, France*.

MOREL, M. AND CLAISSE, S. 2010b. Integrated system for interoperable sensors & information sources for common abnormal vessel behaviour detection & collaborative identification of threat (i2c). In *Ocean and Coastal Observation: sensors and observing systems, numerical models and information systems, Brest, France*.

MOTULSKY, H. 1995. Intuitive biostatistics: Choosing a statistical test, chapter-17. Tech. rep., ISBN 0-19-508607-4), Oxford University Press Inc available on Google Book.

MOYA, M., KOCH, M., AND HOSTETLER, L. 1993. One-class classifier networks for target recognition applications. Tech. rep., Sandia National Labs., Albuquerque, NM (United States).

MUNDHE, M. AND SEN, S. 2000. Evaluating concurrent reinforcement learners. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*. IEEE, 421–422.

NANNEN, V. AND EIBEN, A. 2007. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th international joint conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., 975–980.

NEGNEVITSKY, M. 2005. *Artificial intelligence: a guide to intelligent systems*. Addison-Wesley Longman.

NEMHAUSER, G. L. 1966. *Introduction to dynamic programming*. Wiley New York.

NETO, G. 2005. From single-agent to multi-agent reinforcement learning: Foundational concepts and methods.

NEWTON, J. 1978. *Philosophiæ naturalis principia mathematica: Londini 1687*. William Dawson.

NILSSON, D. 1998. An efficient algorithm for finding the m most probable configurationsin probabilistic expert systems. *Statistics and Computing 8*, 2, 159–173.

NILSSON, M., VAN LAERE, J., ZIEMKE, T., AND EDLUND, J. 2008. Extracting rules from expert operators to support situation awareness in maritime surveillance. In *Proceedings of the Eleventh International Conference on Information Fusion*. IEEE.

NILSSON, N. 1982. *Principles of artificial intelligence*. Springer Verlag.

NWANA, H. 1996. Software agents: An overview. *Knowledge Engineering Review 11*, 3, 205–244.

OHKO, T., HIRAKI, K., AND ANZAI, Y. 1997. Addressee learning and message interception for communication load reduction in multiple robot environments. In *Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments*. Springer, 242–258.

OHSAKI, M. AND TAKAGI, H. 2000. Human interface of interactive evolutionary computation and its evaluation. In *Genetic Algorithms*. Vol. 4. Tokyo, Japan: Sangyo, 397–438.

OLIVEIRA, E., FISCHER, K., AND STEPANKOVA, O. 1999. Multi-agent systems: which research for which applications. *Robotics and Autonomous Systems 27,* 1, 91–106.

OPTRONICS, C. Z. Autonomous maritime surveillance system. European Project.

PANAIT, L. AND LUKE, S. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems 11,* 3, 387–434.

PAO, Y. 1989. Adaptive pattern recognition and neural networks.

PARMEE, I. 1999. Exploring the design potential of evolutionary search, exploration and optimisation. *Evolutionary design by computers*, 119–143.

PAVON, R., GLEZ-PEIIA, D., LAZA, R., DIAZ, F., AND LUZON, M. 2009. A multi-agent system approach for algorithm parameter tuning. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09)*. Springer, 140.

PEARL, J. 1984. Heuristics: intelligent search strategies for computer problem solving.

PETCU, A. AND FALTINGS, B. 2005. A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*. Vol. 19. Citeseer, 266.

PICARD, G. 2004. Méthodologie de développement de systèmes multi-agents adaptatifs.

PLAMONDON, R. AND SRIHARI, S. N. 2000. Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22,* 1, 63–84.

POLHILL, J., PARKER, D., BROWN, D., AND GRIMM, V. 2008. Using the odd protocol for describing three agent-based social simulation models of land-use change. *Journal of Artificial Societies and Social Simulation 11,* 2, 3.

POTTER, M. A., MEEDEN, L. A., AND SCHULTZ, A. C. 2001. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *International joint conference on artificial intelligence*. Vol. 17. Citeseer, 1337–1343.

POZZOBON, A., SCIUTTO, G., AND RECAGNO, V. 1999. Security in ports: the user requirements for surveillance system. In *Advanced Video-Based Surveillance Systems*. Springer, 18–26.

PRITSKER, A. 1979. Compilation of definitions of simulation. *Simulation 33,* 2, 61.

RAIEVSKY, C. 2004. Emotions artificielles pour la structuration sociale d'un groupe de robots. Ph.D. thesis, Universite de Sherbrooke.

RAMASWAMY, S., RASTOGI, R., AND SHIM, K. 2000. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*. Vol. 29. ACM, 427–438.

RANJBAR-SAHRAEI, B., WEISS, G., AND NAKISAEE, A. 2012. A multi-robot coverage approach based on stigmergic communication. In *Multiagent System Technologies*. Springer, 126–138.

RASHEED, Z., SHAFIQUE, K., YU, L., LEE, M., RAMNATH, K., CHOE, T., JAVED, O., AND HAERING, N. 2011. Distributed sensor networks for visual surveillance. *Distributed Video Sensor Networks*, 439–449.

REGAZZONI, C. S., FABRI, G., AND VERNAZZA, G. 1999. *Advanced video-based surveillance systems*. Vol. 488. Kluwer Academic Pub.

REMAGNINO, P., BAUMBERG, A., GROVE, T., HOGG, D., TAN, T., WORRALL, A., BAKER, K., ET AL. 1997. An integrated traffic and pedestrian model-based vision system. In *Proceedings of the eighth british machine vision conference (BMVC97)*. 380–389.

RISTIC, B., LA SCALA, B., MORELANDE, M., AND GORDON, N. 2008. Statistical analysis of motion patterns in ais data: Anomaly detection and motion prediction. In *11th International Conference on Information Fusion*. IEEE, p.1–7.

RIVERA-ILLINGWORTH, F., CALLAGHAN, V., AND HAGRAS, H. 2010. Detection of normal and novel behaviours in ubiquitous domestic environments. *The Computer Journal 53,* 2, 142–151.

RONETTI, N. AND DAMBRA, C. 2000. Railway station surveillance: the italian case. In *Multimedia Video-Based Surveillance Systems*. Springer, 13–20.

ROSENBLATT, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review 65,* 6, 386–408.

ROTH, V. 2006. Kernel fisher discriminants for outlier detection. *Neural computation 18,* 4, 942–960.

ROUGEMAILLE, S., ARCANGELI, J., GLEIZES, M., AND MIGEON, F. 2009. Adelfe design, amas-ml in action. *Engineering Societies in the Agents World IX*, 105–120.

ROY, J. 2008. Anomaly detection in the maritime domain. *SH Craig, L. Daniel, and TS Theodore, Eds 6945*, 69450W1–14.

RUSSELL, S., NORVIG, P., AND ARTIFICIAL INTELLIGENCE, A. 1995. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*.

RYOO, M. S. AND AGGARWAL, J. K. 2009. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 1593–1600.

SAITO, K. AND NAKANO, R. 1988. Medical diagnostic expert system based on pdp model. In *Neural Networks, 1988., IEEE International Conference on*. IEEE, 255–262.

SANDHOLM, T. AND CRITES, R. 1996a. Multiagent reinforcement learning in the iterated prisoner's dilemma. *BioSystems 37*, 1-2, 147–166.

SANDHOLM, T. W. AND CRITES, R. H. 1996b. On multiagent q-learning in a semi-competitive domain. In *Adaption and Learning in Multi-Agent Systems*. Springer, 191–205.

SATHIYA KEERTHI, S. AND RAVINDRAN, B. 1994. A tutorial survey of reinforcement learning. *Sadhana 19*, 6, 851–889.

SCHAERF, A., SHOHAM, Y., AND TENNENHOLTZ, M. 1995. Adaptive load balancing: A study in multi-agent learning. *Arxiv preprint cs/9505102*.

SCHEEPENS, R., WILLEMS, N., VAN DE WETERING H., ANDRIENKO, G., ANDRIENKO, N., AND VAN WIJK, J. 2011. Composite density maps for multivariate trajectories. In *IEEE Transactions on Visualization and Computer Graphics*, IEEE, Ed. Vol. 17. IEEE, 2518–2527.

SCHMIDT, M. AND SHANKER, T. 2012. To smuggle more drugs, traffickers go under the sea. *New York Times 9-12*, 9 – 12.

SCHREMPF, O., ALBRECHT, D., AND HANEBECK, U. 2007. Tractable probabilistic models for intention recognition based on expert knowledge. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 1429–1434.

SCHWEFEL, H. 1993. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc.

SERUGENDO, G. D. M., GLEIZES, M.-P., AND KARAGEORGOS, A. 2011. Self-organising systems. In *Self-organising Software*. Springer, 7–32.

SHAPIRO, J. 2008. Learning from rewards, game theory, learning and games. *Foundation Lectures on Adaptative Systems 1*, 1.

SHI, Y., TIAN, Y., KOU, G., PENG, Y., AND LI, J. 2011. Network intrustion detection.

SHOHAM, Y., POWERS, R., AND GRENAGER, T. 2003. Multi-agent reinforcement learning: a critical survey. *Unpublished survey. http://robotics. stanford. edu/shoham*.

SIERRA, C., SABATER, J., AGUSTÍ, J., AND GARCIA, P. 2002. Evolutionary programming in sadde. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*. ACM, 1270–1271.

SIMS, K. 1993. Interactive evolution of equations for procedural models. *The Visual Computer 9*, 8, 466–476.

SIMS, K. 1994. Interactive evolution of dynamical systems. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. 171–178.

SISLAK, D., VOLF, P., AND PECHOUCEK, M. 2010. Large-scale agent-based simulation of air-traffic.

SMIT, S. AND EIBEN, A. 2009. Comparing parameter tuning methods for evolutionary algorithms. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 399–406.

SOCIETY, R., Ed. 2009. *Three and a half centuries of Royal Society publishing*. 1, vol. 1. Royal Society.

SOULE, A., SALAMATIAN, K., AND TAFT, N. 2005. Combining filtering and statistical methods for anomaly detection. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. USENIX Association, 31–31.

SRINIVAS, M. AND PATNAIK, L. 1991. Learning neural network weights using genetic algorithms-improving performance by search-space reduction. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*. IEEE, 2331–2336.

STEFANUK, V. 2000. Dynamic expert systems. *Kybernetes 29*, 5/6, 702–709.

SUBAGDJA, B., SONENBERG, L., AND RAHWAN, I. 2009. Intentional learning agent architecture. *Autonomous Agents and Multi-Agent Systems 18*, 3, 417–470.

SUN, J., XIE, Y., ZHANG, H., AND FALOUTSOS, C. 2007. Less is more: Compact matrix decomposition for large sparse graphs.

SURYADI, D. AND GMYTRASIEWICZ, P. J. 1999. Learning models of other agents using influence diagrams. *COURSES AND LECTURES-INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES*, 223–234.

SUTTON, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning 3*, 1, 9–44.

SUTTON, R. S. AND BARTO, A. G. 1998. *Reinforcement learning: An introduction*. Vol. 1. Cambridge Univ Press.

TABUCHI, M. AND TAURA, T. 1996. Methodology for interactive knowledge acquisition between genetic learning engine and human. *JOURNAL-JAPANESE SOCIETY FOR ARTIFICIAL INTELLIGENCE 11*, 600–607.

TAKAGI, H. 1996. System optimization without numerical target. In *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*. IEEE, 351–354.

TAKAGI, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE 89*, 9, 1275–1296.

TAKAGI, H., OHSAKI, M., AND INGU, T. 1998. The methods to reduce the burden of human interactive ec operators. In *Workshop on Interactive Evolutionary Computation*. 47–52.

TAN, A. C. AND GILBERT, D. 2003. An empirical comparison of supervised machine learning techniques in bioinformatics. In *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003-Volume 19*. Australian Computer Society, Inc., 219–222.

TAN, K. 2005a. A multi-agent system for tracking the intent of surface contacts in ports and waterways. Tech. rep., DTIC Document.

TAN, K. S. 2005b. A multi-agent system for tracking the intent of surface contacts in ports and waterways. M.S. thesis, Naval Postgraduate School Monterey CA.

TANGAMCHIT, P., DOLAN, J. M., AND KHOSLA, P. K. 2002. The necessity of average rewards in cooperative multirobot learning. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. Vol. 2. IEEE, 1296–1301.

TAO, G. AND KOKOTOVIC, P. 1995. Discrete-time adaptive control of systems with unknown deadzones. *International Journal of Control 61*, 1, 1–17.

TERANO, T. AND ISHINO, Y. 1996. Knowledge acquisition from questionnaire data using simulated breeding and inductive learning methods. *Expert Systems With Applications 11*, 4, 507–518.

TESAURO, G. 2004. Extending q-learning to general adaptive multi-agent systems. *Advances in neural information processing systems 16*.

THERAULAZ, G. AND BONABEAU, E. 1999. A brief history of stigmergy. *Artificial life 5*, 2, 97–116.

THOMPSON, W. W., COMANOR, L., AND SHAY, D. K. 2006. Epidemiology of seasonal influenza: use of surveillance data and statistical models to estimate the burden of disease. *Journal of Infectious Diseases 194*, Supplement 2, S82–S91.

TREUILLET, S., DRIOUCHI, D., AND RIBEREAU, P. 2004. Ajustement des parametres d'une chaine de traitements d'images par un plan d'experiences factoriel fractionnaire 2k-p. *Traitement du signal 21*, 2, 141.

TSANKOVA, D. AND GEORGIEVA, V. 2004. From local actions to global tasks: simulation of stigmergy based foraging behavior. In *Intelligent Systems*. IEEE, p.353–358.

TURK, M. A. AND PENTLAND, A. P. 1991. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 586–591.

VALERA, M. AND VELASTIN, S. 2005. Intelligent distributed surveillance systems: a review. In *Vision, Image and Signal Processing, IEE Proceedings-*. Vol. 152. IET, 192–204.

VALLAT, F. 2007a. L'agence européenne de sécurité maritime taille bien la route. *Défense nationale et sécurité collective 2*.

VALLAT, F. 2007b. L'agence europeenne de securite maritime taille bien la route. *DEFENSE NATIONALE-PARIS-ETUDES POLITIQUES STRATEGIQUES MILITAIRES ECONOMIQUES SCIENTIFIQUES 63*, 2, 109.

VAN DE VIJVER, G. 1997. Emergence et explication. *Intellectica 25*, 7-23, 22.

VAN DYKE PARUNAK, H. 1997. "go to the ant": Engineering principles from natural multi-agent systems. *Annals of Operations Research 75*, 69–101.

VANEK, O. 2010. Agent-based simulation of the maritime domain.

VANĚK, O., JAKOB, M., HRSTKA, O., AND PĚCHOUČEK, M. 2012. Using multi-agent simulation to improve the security of maritime transit. In *Multi-Agent-Based Simulation XII*. Springer, 44–58.

VENTURINI, G. 1994. Apprentissage adaptatif et apprentissage supervise par algorithme genetique. Ph.D. thesis.

VESPE, M., SCIOTTI, M., BURRO, F., BATTISTELLO, G., AND SORGE, S. 2008. Maritime multi-sensor data association based on geographic and navigational knowledge. In *Radar Conference, 2008. RADAR'08. IEEE*. IEEE, 1–6.

VICINI, P., GASTONGUAY, M., AND FOSTER, D. 2002. Model-based approaches to biomarker discovery and evaluation: a multidisciplinary integrated review. *Critical reviews in biomedical engineering 30*, 4-6, 379–418.

VON FOERSTER, H. 1960. On self-organizing systems and their environments. *Self-organizing systems*, 31–50.

WALDROP, M. 1992. *Complexity: The emerging science at the edge of order and chaos*. Vol. 12. Simon & Schuster New York.

WANG, B., YE, M., LI, X., ZHAO, F., AND DING, J. 2011. Abnormal crowd behavior detection using high-frequency and spatio-temporal features. *Machine Vision and Applications*, p.1–11.

WATKINS, C. J. AND DAYAN, P. 1992. Q-learning. *Machine learning 8*, 3-4, 279–292.

WATKINS, C. J. C. H. 1989. Learning from delayed rewards. Ph.D. thesis, University of Cambridge.

WEISS, G. 1995. Adaptation and learning in multi-agent systems: Some remarks and a bibliography. *Adaption and learning in multi-agent systems*, 1–21.

WEISS, G. 1999. *Multiagent systems: a modern approach to distributed artificial intelligence*. The MIT press.

WELCOMME, J., GLEIZES, P., REDON, R., AND DRUOT, T. 2006. Self-regulating multi-agent system for multi-disciplinary optimisation process. In *Proceedings of the EUMAS 4rd European Workshop on Multi-Agent Systems*.

WILLIAMS, R. AND ZIPSER, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation 1,* 2, 270–280.

WITTENMARK, B. 2002. Adaptive dual control.

WOLPERT, D. AND MACREADY, W. 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation 1,* 1, 67–82.

WOLPERT, D. H. AND TUMER, K. 2001. Optimal payoff functions for members of collectives. *Advances in Complex Systems 4,* 02n03, 265–279.

WOOLDRIDGE, M. AND JENNINGS, N. 1995. Agent theories, architectures, and languages: a survey. *Intelligent agents*, 1–39.

WOOLDRIDGE, M., JENNINGS, N., AND KINNY, D. 2000. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems 3,* 3, 285–312.

XU, Y., MURAL, R. J., EINSTEIN, J. R., SHAH, M. B., AND UBERBACHER, E. C. 1996. Grail: A multi-agent neural network system for gene identification. *Proceedings of the IEEE 84,* 10, 1544–1552.

ZAÏANE, O. R. AND LEE, C.-H. 2002. Clustering spatial data in the presence of obstacles: a density-based approach. In *Database Engineering and Applications Symposium, 2002. Proceedings. International*. IEEE, 214–223.

ZANDER, S., NGUYEN, T., AND ARMITAGE, G. 2005. Automated traffic classification and application identification using machine learning. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*. IEEE, 250–257.

ZHANG, R., QIAN, D., BA, C., WU, W., AND GUO, X. 2001. Multi-agent based intrusion detection architecture. In *Computer Networks and Mobile Computing, 2001. Proceedings. 2001 International Conference on*. IEEE, 494–501.

ZIEGLER, J. AND NICHOLS, N. 1942. Optimum settings for automatic controllers. *trans. ASME 64,* 8, 759–768.

List of Figures

List of Tables

Title : Self-Adaptive Multi-Agent System for Aided Decision-Making : An Application to Maritime Surveillance

English Summary :

The maritime activity has widely grow in the last few years and is the witness of several illegal activities. It has become necessary that the organizations involved in the maritime surveillance possess efficient systems to help them in their identification. The maritime surveillance systems must observe a wide maritime area, identify the anomalies in the behaviours of the monitored ships et trigger alerts when these anomalies leads to a suspicious behavior.

We propose a generic agent model, called MAS4AT, able to fulfil two main roles of a surveillance system: the numerical representation of the behaviours of the monitored entities and learning mechanisms for a better efficiency. MAS4AT is integrated in the system I2C.

KEYWORDS: Multi-Agent System, Maritime Surveillance, Learning, Aided Decision-Making

AUTEUR : Nicolas Brax

TITRE : Systèmes multi-agents adaptatifs pour l'aide à la prise de décision : application à la surveillance maritime.

DIRECTEUR DE THESE : Marie-Pierre Gleizes & Eric Andonoff

LIEU ET DATE DE SOUTENANCE : IRIT, le 27/11/2013

RESUME en français

L'activité maritime s'est fortement développée ces dernières années et sert de support à de nombreuses activités illicites. Il est devenu nécessaire que les organismes impliqués dans la surveillance maritime disposent de systèmes efficaces pour les aider à identifier ces activités illicites. Les Systèmes de Surveillance Maritime doivent observer de manière efficace un espace maritime large, à identifier des anomalies de comportement des navires évoluant dans l'espace en question, et à déclencher des alertes documentées si ces anomalies amènent à penser que les navires ont un comportement suspect.

Nous proposons un modèle générique de système multi-agents, que nous appelons MAS4AT, capable de remplir deux des différents rôles d'un système de surveillance : la représentation numérique des comportements des entités surveillées et des mécanismes d'apprentissage pour une meilleure efficacité. MAS4AT est intégré au système I2C.

MOTS CLES : système multi-agent, surveillance maritime, apprentissage, aide à la décision

---

ED MITT : STIC - Intelligence Artificielle

INTITULE ET ADRESSE DE L'U.F.R. OU DU LABORATOIRE : Institut de Recherche en Informatique de Toulouse

IRIT, Univesité Paul Sabatier, 118 route de Narbonne, F-31062 Toulouse Cedex 9