PATRICK BEAUMONT

Multi-Platform Coordination and Resource Management in Command and Control

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

AOÛT 2004

Résumé

Depuis plusieurs années, nous constatons l'augmentation de l'utilisation des techniques d'agents et multiagent pour assister l'humain dans ses tâches. Ce travail de maîtrise se situe dans la même voie.

Précisément, nous proposons d'utiliser les techniques multiagent de planification et de coordination pour la gestion de ressources dans les systèmes de commande et contrôle (C2) temps réel. Le problème particulier que nous avons étudié est la conception d'un système d'aide à la décision pour les opérations anti-aérienne sur les frégates canadiennes. Dans le cas où plusieurs frégates doivent se défendre contre des menaces, la coordination est un problème d'importance capitale. L'utilisation de mécanismes de coordination efficaces permet d'éviter les actions conflictuelles et la redondance dans les engagements.

Dans ce mémoire, nous présentons quatre mécanismes de coordination basés sur le partage de tâche. Trois sont basés sur les communications : la coordination centrale, le Contract Net, la coordination similaire à celle proposée par Brown; tandis que la défense de zone est basée sur les lois sociales. Nous exposons enfin les résultats auxquels nous sommes arrivés en simulant ces différents mécanismes.

Abstract

The use of agent and multiagent techniques to assist humans in their daily routines has been increasing for many years, notably in Command and Control (C2) systems. This thesis is is situated in this domain.

Precisely, we propose to use multiagent planning and coordination techniques for resource management in real-time C2 systems. The particular problem we studied is the design of a decision-support for anti-air warfare on Canadian frigates. In the case of several frigates defending against incoming threats, multiagent coordination is a complex problem of capital importance. Better coordination mechanisms are important to avoid redundancy in engagements and inefficient defence caused by conflicting actions.

In this thesis, we present four different coordination mechanisms based on task sharing. Three of these mechanisms are based on communications: central coordination, Contract Net coordination and ∼Brown coordination, while the zone defence coordination is based on social laws. Finally, we expose the results obtained while simulating these various mechanisms.

Avant-propos

J'aimerais prendre quelques lignes pour remercier les personnes qui ont contribué à l'aboutissement de ce mémoire. J'aimerais tout d'abord remercier mon directeur de recherche, M. Brahim Chaib-draa, pour sa grande disponibilité, son soutien, ses idées, ainsi que ses précieux conseils.

J'aimerais ensuite remercier les personnes qui ont travaillé avec moi sur le projet NEREUS. Merci à mes collègues étudiants qui m'ont aidé à progresser au fil de nombreuses discussions : Martin Soucy, Pierrick Plamondon, Jean-François Morissette et Étienne Bolduc. Thanks to Dale Blodgett, our contact at LMC, who has always been very professional. Merci à Abder Benaskeur, notre contact au RDDC de Valcartier qui nous a aidés avec ses connaissances sur le domaine.

J'aimerais de plus remercier des gens qui m'ont fait apprécier mon passage au département d'IFT-GLO: Guy Mineau, directeur du département, Abderrahim Alikacem, chargé de cours exceptionnellement passionné par l'enseignement, Lynda Goulet, agente de gestion des études de deuxième cycle et toujours souriante, et Gilles Caron, l'homme qui peut tout faire.

Merci aussi à mes amis de longue date qui ont toujours été là quand j'avais besoin d'eux. Merci à mon frère Mathieu, qui croit en moi plus que n'importe qui d'autre.

Merci à Jacinthe, ma compagne, pour son soutien, sa grande patience et sa compréhension, tout spécialement lors des dernières semaines de rédaction.

Finalement, j'aimerais dédier ce mémoire à mes parents Christian et Lise, pour leur compréhension, leur générosité, leur soutien dans tous mes projets et particulièrement pour l'éducation et les valeurs qu'ils m'ont transmises. Merci d'avoir cru en moi et de m'avoir aidé à devenir celui que je suis.

Patrick Beaumont

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

For many years now, technology has taken increasing importance in many domains. In fact, we can even consider that since the Renaissance, Man has continued an uninterrupted race toward new technologies. Information and communication technologies permit the resolution of complex problems that would have been unthinkable to solve only some years ago. Nowadays, more and more critical domains use new technologies to assist humans in their daily routines, as well as in more complex tasks.

Recently, we have seen the emergence in the Artificial Intelligence (AI) community of agent techniques as a new paradigm. A simple way to describe agents[1] is to say that they are entities existing and perceiving in an environment, and capable of acting rationality (Russell and Norvig [2003]).

However, most environments are complex enough that no single agent can execute every task by himself. As in our society, great accomplishments can be achieved with the collaboration of many individuals. This supports the recent research in Multiagent System (MAS), making it is possible to attack very complex distributed problems. In this kind of system, there is a need to investigate the relations of coexistence, competition and cooperation among agents.

Most of the time, simulation is used to model very complex systems. In real life domains, the complexity of the system is often too broad for an ordinary agent to comprehend in its entirety. Simulation is used in this case to control complexity and limit it to the factors of interest, whatever they might be. Furthermore, in domains such as Command and Control (C2)[2], hard real-time constraints and monetary costs

[1] Agents will be further detailed in Section 2.

[2] The Command and Control is the exercise of authority and direction by a properly designated

legitimate, and often require, the use of simulation.

In this thesis, our interest is focused on multiagent techniques in complex systems. More specifically, we are interested in efficient planning and coordination for resource management in real-time C2 systems.

1.1 Problem

Maritime environments are commonly known to be very complex (Lloyd and Witsenhausen [1986]). Hence, modern Anti-Air Warfare (AAW) is an arduous problem to begin with, because of the sheer volume of data, usually imperfect, that needs to be processed under time-critical conditions. During the last decades, operational crews were trained to operate within a force command structure where they can 1) recognise a threat, 2) know how to react to that threat and, 3) employ measures to defeat that threat . Unfortunately, the problem has become increasingly intricate as threats and defence systems gained in sophistication. Unluckily, the scenarios are also gaining in complexity. This is mostly due to engagements occurring now more on the littoral rather than open sea and the ever-increasing need for cooperation between ships with different resources.

In the case of an aerial attack, the operators have but little time to observe, orient, decide and act[3]. There is often less than a minute between detection of a threat and its impact with the ship. This calls for very fast decisions made by considering several important factors to make sure the best possible plan is carried out. Failing to do so might mean destruction of the ship and death of its crew. Because there is not much time to consider a great number of plans, the operators might overlook the most advantageous prospects, resulting in the choice of a suboptimal plan. Moreover, under such real-time constraints, the commander might make errors simply due to the complexity of the environment or the stress that such a situation generates, which will obviously result in dire consequences.

When there is more than one ship together, some resources have more restrictions than usual, whilst there are synergic interactions that could potentially increase the overall survival of the fleet; the whole is greater than the sum of its parts. Of course, deciding on a course of action, communicating it to allies and adjusting the initial plan

commander over assigned and attached forces in the accomplishment of a mission. C2 will be further discussed in Section 1.1.2.

[3]Observe, Orient, Decide, Act are the four parts of the *OODA Loop*. For more details on the OODA loop, consult Boyd [1987] and Boyd [1996].

to the schedule still has to be done under the time constraints, as stated earlier.

This situation poses significant challenges to future shipboard C2 systems and the operators using these systems to defend the ship. Considering the complexity of the problem and the fact that computers can operate and communicate tremendously faster than humans, it appears that a reliable onboard decision-support system (DSS)[4] would really be welcome in modern AAW.

1.1.1 Project NEREUS Participants

As said earlier, agents can be used in many applications, and particularly in complex environments. For our part, we directed our attention on the particular problem of designing a DSS for AAW on Canadian frigates. The study of these frigates was initiated through a project called Naval Environment for Resource Engagement in Unpredictable Situations (NEREUS). This project has two components. Firstly, it aims at implementing an agent for the Resource Management (RM) on a typical frigate. Secondly, it aims at implementing a multiagent system for the coordination of a fleet of frigates. In other words, the goal of the system is to efficiently manage all the resources (weapons, radars, electronic systems, etc.) present on a ship or in a fleet to increase survival chances of frigates at the time of attack by Anti-Ship Missile (ASM)s. Since resource managing for more than one frigate is a distributed problem, we believe that a MAS is appropriate to manage those resources while increasing the survivability of each frigate (Kropf et al. [2000]).

Project NEREUS got its name from the Greek deity, Nereus:

> Nereus is the righteous and all-wise "old man of the sea", god of the Mediterranean Sea, son of Gaia and Pontus. His wife is Doris and she became by him the mother of the fifty Nereids, friendly sea-nymphs. Nereus is a gentle and very wise old man who has the power to foretell the future, but he will not answer questions unless he was caught and to avoid that he would change his shape (such as when Heracles came to ask him the way to the Garden of the Hesperides). The domain of Nereus and his fifty daughters is especially the Aegean Sea where he has saved many ships from destruction. (Lindemans [2004]).

Project NEREUS is a collaborative project between the DAMAS laboratory, Lock-

[4] A decision-support system is a software agent used to aid human operators in their decision-making.

heed Martin Canada (LMC) and the centre for Defence Research and Development Canada - Valcartier (DRDC - Valcartier).

DAMAS Laboratory

Dialog, Automatic Learning and Multiagent Systems (DAMAS)[5] is a research group supervised by Dr. Brahim Chaib-draa, professor at the Computer Science and Software Engineering Department of Laval University. Dialog, Automatic Learning and Multiagent Systems (DAMAS) interests are essentially:

- Dialogue and communication among agents.
- Multiagent negotiation and coordination.
- Real-time multiagent environments.
- Learning in multiagent environments.
- Cooperation and competition among agents.
- Application of agents and multiagent technologies to: supply chains, e-business, automated highway systems, RobotCupRescue, real-time defence systems, etc.

Lockheed Martin Canada

Lockheed Martin Canada (LMC)[6] is a highly diversified global enterprise principally engaged in the research, design, manufacture, and integration of advanced-technology products. The company is a leader in systems integration, software development and large-scale program management and Canada's premier supplier of electronic defence and surveillance systems. Primary capabilities encompass the integration and management of complex computer-based electronic systems; the design, manufacture and supply of military-standard computers, electronic warfare, sonar and security systems; and the provision of life cycle support for major platforms.

LMC's mission is to be recognised as the Canadian centre of excellence for providing advanced technological solutions to complex problems requiring systems integration and management, satisfying the needs of customers worldwide.

[5]For more information, see: www.damas.ift.ulaval.ca
[6]For more information, see: www.lockheedmartin.com/canada/about.htm

Lockheed Martin Canada employs over 600 people across Canada with its head office in Kanata and supporting operations in Montréal, Halifax, and Victoria.

Defence Research and Development Canada - Valcartier

Defence Research and Development Canada - Valcartier (DRDC - Valcartier)[7] scientists are striving to advance defence technology in optronic systems, information systems and combat systems, to extend the limits of knowledge, and to exploit the economic opportunities created by their investment of time and energy. In this way, technologies and processes developed are more easily carried through to innovative defence and civilian applications. With highly qualified personnel and unique facilities, DRDC - Valcartier centre is renowned for the leading-edge work it has done through many bilateral and multilateral alliances and under NATO agreements. DRDC - Valcartier expertise, besides contributing to the success of the Canadian Forces operations, enriches the pool of knowledge accessible, via Business Development Service, to the private sector, and the entire scientific and technological community.

DRDC - Valcartier's mission is to improve Canada's defence capabilities, through research and development, by providing independent expert advice and by investigating, demonstrating and exploiting innovative technological concepts for combat, electro-optical and command and control information systems. The second part of this mission is to be recognised as a world-class research and development team of committed personnel with diverse and complementary skills, demonstrating leadership in science and technology and support services tailored to suit the Canadian Force's needs.

1.1.2 Command and Control

Command and Control (C2) is the exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of a mission. C2 functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the accomplishment of a mission. C2 tasks usually include weapon and sensor systems control, tactical picture compilation, situation interpretation and threat evaluation, weapon selection, engagement monitoring and mission planning and evaluation. In fact, the C2 tasks cover what is called the Observe, Orient, Decide, Act (OODA) loop. This theory, put forth

[7]For more information, see: www.drev.dnd.ca

by Colonel John R. Boyd (See Boyd [1987] and Boyd [1996]), essentially states that, in a confrontation, whoever is quicker to react to changes will prevail. Boyd describes this readiness to react to changes as "having a tight OODA loop". Therefore, Boyd's theory can be expressed as "whoever has a tighter OODA loop, will prevail".

Thus, the C2 process necessitates a highly dynamic flow of information. Decision-making involves a number of operators and sophisticated DSSs with a concomitant requirement for developing a common, shared representation of the situation. In this context, there are many generic issues worth considering. Indeed, the development of a relevant C2 theory will have significant impact upon the analysis and design of both military and civil C2 systems. The major considerations are the following (Chaib-draa et al. [2001]):

- *C2 is a multiagent environment:* A C2 system is a multiagent organisation in which the decision-makers are both human and artificial agents. The decision-makers are often geographically scattered due to the operational environment and the physical nature of sensors and resources. Cooperation, coordination and communication between the decision-makers are thus critical in such a distributed C2 architecture.

- *C2 has a functional architecture:* Another key element of the C2 process is its functional decomposition. Indeed, the C2 process can be decomposed into a set of generally accepted C2 functions that must be executed in a reasonable time frame to ensure success.

- *C2 is a complex process:* The complexity of most C2 problems rises from the multitude, the heterogeneity and the interrelations of the resources involved. Generally, no decider alone can deal with the inherent complexity of the global situation. This leads to a decomposition of the decision process along distinct expertise dimensions. In light of these considerations, team training is essential in any C2 organisation to achieve superior coordination and to make the best utilisation of common resources. Moreover, a military C2 system must take into account the specific established command and decision hierarchy.

- *C2 deals with large volumes of data under stringent time constraints:* Perceptual and cognitive processing is further complicated by the fact that the underlying information is derived by continuously integrating and merging data from a variety of sources to build a coherent situational picture. Particular processing problems arise from information with different accurateness and timeliness. The integrated data is generally imperfect; it can be uncertain, incomplete, imprecise, inconsistent and ambiguous, due to limited sensor coverage, report ambiguities, report

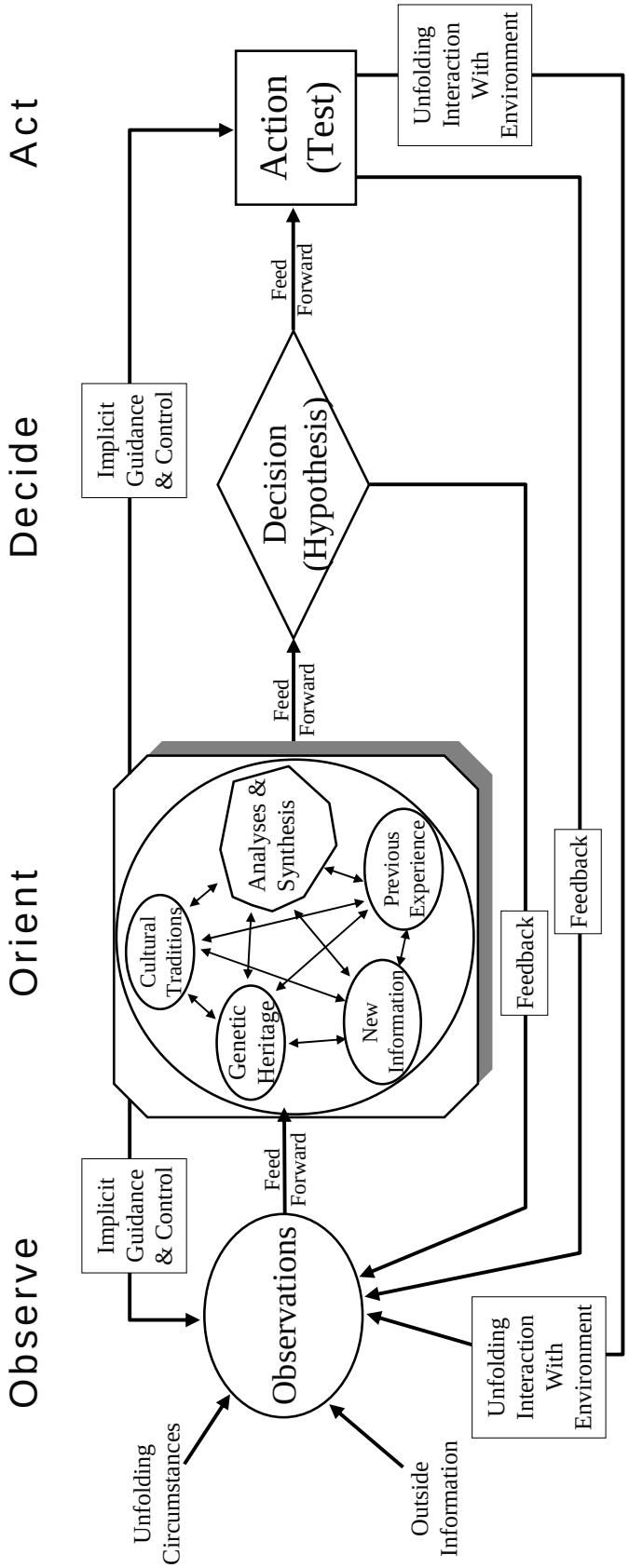

Figure 1.1: The OODA loop (Boyd [1987]).

conflicts or inaccuracies in measured data (Waltz and Buede [1986]). It follows that 1) operators may have to handle potentially large situation uncertainties and 2) at any given moment, there may be several likely interpretations of the tactical picture. This leads to processing large volumes of data under stringent time constraints.

In the case of AAW, the list of functions of the C2 architecture is as follows:

1. *Threat detection*: Based on data from several sensors.
2. *Target tracking*: Usually based on data fusion.
3. *Discrimination*: Results in the resolution of true threats from decoys.
4. *Identification*: In this step, the threats are identified.
5. *Battle planning*: In this process, decisions are made on how to deal with the identified threats.
6. *Resource assignments*: Resources are assigned to engage each threat.
7. *Engagement control*: The process by which decisions in the two preceding steps are executed in real-time.
8. *Damage assessment*: This process evaluates the outcome of the engagement control.

In this thesis, we address how we developed a multiagent system that focuses specifically on some particular aspects of the C2, in order to reduce the complexity of the domain. Our primary focus is on the *battle planning*, *resource assignment* and *engagement control* processes. In this project, we consider the Situation and Threat Assessment (STA), which consists of *threat detection*, *target tracking*, *discrimination* and *identification*, as a black box. Therefore, the output of these steps is taken directly as available data, since it is not in the scope of our project to work on the STA. The *damage assessment*, which covers the evaluation of the damage to the ship, is not currently in our research goals, since we only need to evaluate the damages in a simple fashion: destroyed, damaged or intact. Not working on STA and damage assessment (DA) reduces the large volume of data that needs to be processed, which helps reducing the system's complexity.

1.1.3 Frigate Resources

Since the main goal of AAW is to destroy incoming threats, it is important to define such threats. Throughout this thesis, the reader will sometimes find references to *ASM*, *Anti-Ship Missile* or *threat*, which are used without distinction to represent Anti-Ship Missile (ASM) threats coming toward the frigate. A description of a typical threat can be found in Appendix A.3. It is important to note that, at this stage of the project, only one type of threat is used. However, it has been planned to introduce diversity in incoming threats in the near future of the project.

Our decision support agent, which is the reasoning part of the actual DSS for a frigate, uses many different resources to defend the frigate against threats. The exact nature of the specifications and capabilities of the various AAW weapons on real frigates is obviously very complex and much of that information is classified. Thus, we cannot use the actual data concerning these systems. To avoid this issue and in order to maintain emphasis on the research interests and not be burdened by the complexity and fidelity of the representation of weapon systems, a considerably simplified (and non-classified) model of the relevant AAW hardkill and softkill weapons for a typical frigate was used.

The AAW hardkill weapons are directed to intercept a threat and actively destroy it through direct impact or explosive detonation in the proximity of the threat. Effectiveness of these weapons depends on a variety of factors: distance, type and speed of the threat, environment, etc. The AAW hardkill weapons for a typical frigate include Surface-to-Air Missile (SAM)s, which are long-range interception missiles, an intermediate range gun, and a Close-in Weapon System (CIWS) that is a short-range, rapid-fire gun. The gun can fire rounds up to a rate of 200 rounds/minute, while the CIWS can fire high-velocity rounds up to a rate of 3,300 rounds/minute.

The AAW softkill weapons use techniques to deceive or disorient a threat to cause its self-destruction, or at least lose its fix on its intended target. The softkill resources consist of chaff shells and jamming systems. The jamming systems use electromagnetic emissions to confuse the threat's sensors, causing the threat either to lose its fix on its intended target, or to improperly assess the position of its target. The chaff system launches a shell that produces a burst at a designated position; the resultant chaff cloud is conceived to screen the frigate or produce an alternate target on which a radar-guided threat can fix.

An exhaustive description of the resources of a typical frigate is included in Appendix A.1.

Resource Interactions

In AAW, the complexity of hardkill and softkill systems of the frigate creates many different interactions that we can observe. These interactions can be organised into two types: positive and negative. Such interactions must be managed in order to obtain the best survival answer. To achieve this, it is necessary to benefit from positive interactions, and to avoid negative interactions. The following interactions, not supposed to be an exhaustive list, indicate the range of possible interactions (Liang and Liem [1992], Liang [1995], and Plamondon [2003]).

Positive Interactions

- *Jamming and chaff:* Using chaff in conjunction with jamming creates a synergy between the two systems. The jamming leads the threat to wrongly assess the position of its target. This process is easier when the jammer can direct said threat to a false position where a chaff cloud is already present. Indeed, it is easier for the threat to *believe* the frigate is located at this false position, since there is also a strong Infrared (IR) signature and a large radar cross section (RCS) confirming this belief.

Negative Interactions

- *SAM and SAM:* If two SAMs are launched against two different threats near each other, it is possible that the explosion of the first SAM damages the second.
- *Chaff and hardkill systems:* When a Separate Tracking and Illuminating Radar (STIR) or CIWS radar is trying to guide a hardkill weapon through a chaff cloud, its range might be greatly diminished, or the guidance can even become completely impossible.
- *Jamming and hardkill systems:* As we have seen earlier, the jamming systems can change the direction of an incoming ASM. Thus, it is possible that the used hardkill weapon misses its target. Some could argue, not without reason, that if we suppose that the jamming deflected the threat, the later will not hit independently of whether the SAM intercepts or not. However, considering the fact that an ASM can get its fix back on the target soon enough to still be effective, the actual interaction could effectively be negative.
- *Hardkill and softkill systems:* In certain cases, the ship orientation needed to support the deployment of hardkill weapons may make it impossible to deploy softkill weapons against some of the incoming threats, and vice versa.

1.2 Motivation

Having described the problem, we can now enumerate the motivations that lead us to work on this particular subject:

- We address a real-world problem within a very complex environment with hard real-time constraints.
- We have the possibility to experiment such complex environment with some of the latest techniques in artificial intelligence, especially agents and multiagent technologies.
- We lack a stable simulation platform to develop real-time multiagent systems such as the ones used in project NEREUS.
- We need an efficient model of coordination among the different frigates working in a task group.

1.3 Objectives

The main objective of the current thesis is to specify, develop and validate coordination mechanisms to coordinate frigates' defence resources in a real-time, multi-platform system. These mechanisms, in addition to having to respect hard real-time constraints, must maximise the survival rate of the frigates while minimising resource utilisation. Furthermore, it is important to minimise the negative interactions among these resources while promoting positive interactions.

In order to achieve these objectives, we must:

- Study strong temporal constraints and therefore real-time aspects, as they are reflected in our problem.
- Develop a real-time naval combat simulator: the Naval Defence Simulator (NDS).
- Develop coordination mechanisms among several frigates.
- Integrate the developed coordination mechanisms in NDS.
- Implement a tool to validate the conceived mechanisms with empiric testing.

1.4 Organisation of the Thesis

This thesis is organised as follows: Firstly, chapter 2 introduces the basics of agents, multiagent systems and coordination. Then, chapter 3 presents the types of real-time systems. This chapter also demonstrates how agents can plan in such systems. Chapter 4 presents the Naval Defence Simulator (NDS), developed at DAMAS for the purpose of simulating a naval anti-air warfare environment. Then, chapter 5 describes past work that has been done on planning for a single frigate, and a new approach to frigate positioning. Chapter 6 presents the coordination in project NEREUS and different mechanisms developed to coordinate frigates actions in anti-air warfare. Finally, in chapter 7 we conclude the thesis and propose ideas for future work.

Chapter 2

Agents and Multiagent Systems

For centuries, Man has prided himself of being the most intelligent species known. Yet, humankind seems to be driven by a strong desire to give birth to *something*. The desire to create an intelligent being is not recent, even not from our century. In 1816, Mary Shelley, an 18 years-old Englishwoman, wrote the famous novel *Frankenstein, or the Modern Prometheus*[1] whose inspiration goes as far back as Ovid's *Metamorphoses*[2]. The desire of man to create Artificial Intelligence (AI) has continued since then, though it was not in the form we now acknowledge as a field in itself.

Nowadays, AI is used in many areas, in a world where automation is taking a greater place everyday. Thus, AI is a domain that covers many branches. It includes, while not being limited to, concepts of philosophy, mathematics, psychology, computer engineering, operational research and linguistics.

According to Russell and Norvig [2003], modern AI varies along two main dimensions. The first dimension concerns the *thought process* versus the *observed behaviour*. The other dimension is related to the type of performance aimed: *human performance* versus *ideal rationality*.

These two dimensions are better shown in the following table:

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

[1] See Shelley [1818]
[2] See Publius Ovidius Naso [1955]

In this thesis, we are oriented toward *agents thinking rationally*[3]. It is important to note that rationality is not necessarily synonym of omniscience or omnipotence. In fact, we want our agents to act rationally, *in the limit of their current knowledge*. And yes, that implies that agents might fail to do their task given they lack important information.

According to many authors (Shoham [1993], Wooldridge [1997] and Jennings [2001] to cite some), development of agent-based systems, also called Agent-Oriented Programming (AOP) can be viewed as a new paradigm. Autonomous agents and multiagent systems represent a new way of analysing, designing and implementing complex software systems. Initially proposed by Shoham (Shoham [1990], Shoham [1993]), the key idea is to directly program agents in terms of the mental and intentional notions that agent theorists have developed to represent the properties of agents.

A fully developed agent-oriented programming system would have three components:

- A logical system for defining the mental state of agents.
- An interpreted language for programming agents.
- An *agentification* process, for compiling agent programs into low-level executable systems.

2.1 Autonomous Agents

Before going any further, we need to define what an *agent* is, since the notion of an agent itself is at the centre of the subject of this thesis. The first problem we confront is that there is still no universally accepted definition of what is an agent. According to Wooldridge and Jennings [1995]:

> An agent is a computer system, **situated** in some environment, that is capable of **flexible autonomous** action in order to meet its design objectives.

If we analyse this definition, we see that there are thus three key concepts: situatedness, autonomy, and flexibility (Jennings et al. [1998]).

[3] Although we chose this orientation, we do not mean to say that the other ways are not appropriate. It is a fact that every orientation has yielded significant results in the AI domain.

Situatedness: In our context, it means that the agent receives sensory input from its environment and that it can perform actions that change the environment in some way. Examples of environments in which agents may be situated include the physical world or the Internet. Such situatedness may be contrasted with the notion of disembodied intelligence that is often found in expert systems. In those systems, information is not received via sensors, but through a user acting as a middleman. In the same way, they do not act on any environment, but rather it gives feedback or advice to a third party.

Autonomy: This is a difficult concept to pin down precisely, but it is intended simply in the sense that the system should be able to act rationally without the direct intervention of humans (or other agents), and that it should have control over its own actions and internal state. Some other authors use it in a stronger sense, to mean systems that are capable of learning from experience (Russell and Norvig [2003]).

Of course, situated, autonomous computer systems are not a new development. There are many examples of such systems in existence.

- Any process control system, which must monitor a real-world environment and perform actions to modify it as conditions change (typically in real time). Such systems range from the very simple (for example, thermostats) to the extremely complex (for example, nuclear reactor control systems).
- Software daemons, monitoring a software environment and perform actions to modify the environment as conditions change. A simple example is the email functionality of MSN messenger, which monitors incoming email and obtains users attention by displaying a popup box when new, incoming email is detected.

As we can see, the situatedness and autonomy described are clearly not enough to define an agent since previous examples cannot be considered as agents. In fact, they lack the flexibility to act proactively in their environment to meet their goals.

Flexible: A flexible system needs to be (Wooldridge and Jennings [1995]):

- *Responsive:* Agents should perceive their environment and respond in a timely[4] fashion to changes that occur in it.
- *Proactive:* Agents should not simply act in response to their environment; they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate.

[4]The whole concept of timeliness of real-time systems will be explored in Section 3.1

- *Social:* Agents should be able to interact, when appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities[5].

Figure 2.1 (from Russell and Norvig [1995]) gathers those explanations in a simple representation of what an agent is. This figure illustrates what is called the Performance, Environment, Actuators, Sensors (PEAS) (Russell and Norvig [2003]), which is the description of the *task environment* (the "problem" to which agents are the solution). An agent perceives its *environment* through *sensors*, evaluates the best actions with its *performance* measure, and acts on it through *actuators* (or effectors).

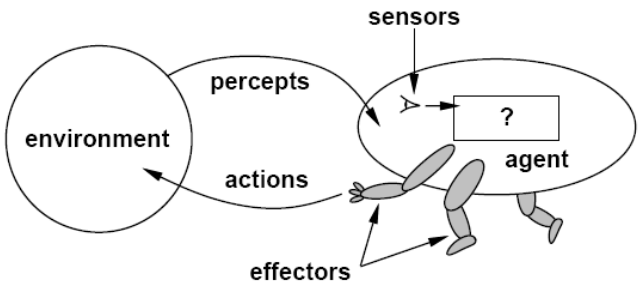

Figure 2.1: Agents interacting with its environment (Russell and Norvig [1995]).

2.1.1 Environments

We said earlier that an agent is situated in an environment, but how can we qualify this environment? Russell and Norvig [2003] define an environment on the following attributes:

Fully observable vs. partially observable: If the sensory apparatus of an agent gives it access to the complete state of the environment, then we say that the environment is fully observable to that agent. An environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action. A fully observable environment is convenient because the agent need not maintain any internal state to keep track of the world.

Deterministic vs. stochastic: If the next state of the environment is completely determined by the current state and the actions selected by the agents, then we

[5]The cooperation aspect will be further elaborated in Section 2.2.2

say the environment is deterministic. In principle, an agent need not worry about uncertainty in an fully observable, deterministic environment. If the environment is partially observable, however, then it may appear to be stochastic. This is particularly true if the environment is complex, making it hard to keep track of all the inaccessible aspects. Usually, most real-world domains are complex enough that, even if they are deterministic, they can be considered as stochastic for practical purposes. Thus, it is often better to think of an environment as deterministic or stochastic from the point of view of the agent.

Episodic vs. sequential: In an episodic environment, the agent's experience is divided into *episodes*. Each episode consists of the agent perceiving and then acting. The quality of its action depends only on the episode itself, because subsequent episodes do not depend on what actions occur in previous episodes. Episodic environments are much simpler because the agent does not need to think ahead. On the other hand, sequential environments cannot be divided into independent episodes.

Static vs. dynamic: If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static. Static environments are easy to deal with because the agent needs not keep looking at the world while it is deciding on an action, nor needs it worry about the passage of time. If the environment does not change with the passage of time but the agent's performance score does, then we say the environment is semi dynamic.

Discrete vs. continuous: If there are a limited number of distinct, clearly defined percepts and actions, then we say that the environment is discrete. Chess is discrete; there are a fixed number of possible moves on each turn. On the other side, environments with continuous variables, like time-based systems, are said to be continuous.

According to these characteristics, the realistic and hardest case is one that is *partially observable*, *stochastic*, *sequential*, *dynamic* and *continuous*. In Section 5.1, we will define the specific attributes of agents and environment for project NEREUS.

2.2 Multiagent System

We have seen earlier that agents are social entities and therefore must be able to interact with other entities (human or artificial agent). This sociability is required since usually,

single agents are not versatile enough to model complex systems. Indeed, much real-world systems are populated by more than one agent.

Traditionally, research into systems composed of multiple agents was carried out under the banner of Distributed Artificial Intelligence (DAI), and has historically been divided into two main camps (Bond and Gasser [1988]): Distributed Problem Solving (DPS) and Multiagent System (MAS). More recently, the term Multiagent System has come to have a more general meaning. MAS now refers to every type of system composed of multiple (semi-)autonomous components.

In distributed problem solving, the consideration is directed to how a particular problem can be solved by a number of modules that cooperate in dividing and sharing knowledge about the problem and its evolving solution. In pure distributed problem solving systems, all interaction strategies are incorporated as an integral part of the system. In contrast, research in MAS is concerned with the behaviour of a collection of possibly pre-existing autonomous agents aiming at solving a given problem. A MAS can be defined as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver (Durfee and Lesser [1989]). These problem solvers (agents) are autonomous and may be heterogeneous in nature. The characteristics of MAS are:

- Each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint.
- There is no global control system .
- Data is decentralised.
- Computation is asynchronous.

To complete the description of MAS, the characteristics of multiagent environments are the following (Huhns and Stephens [1999]):

- Multiagent environments contain agents that are autonomous and distributed, and may be self-interested or cooperative.
- Multiagent environments provide an infrastructure specifying communication and interaction protocols.
- Multiagent environments are typically open and have no centralised designer.

	Attribute	*Range*
Agents	Number	From two upwards
	Uniformity	Homogeneous ... Heterogeneous
	Goals	Contradicting ... Complementary
	Architecture	Reactive ... Deliberative
	Abilities	Simple ... Advanced
Interactions	Frequency	Low ... High
	Persistence	Short-term ... Long-term
	Level	Signal-passing ... Knowledge-intensive
	Pattern	Decentralised ... Hierarchical
	Variability	Fixed ... Changeable
	Purpose	Competitive ... Cooperative
Environment	Predictability	Foreseeable ... Unforeseeable
	Accessibility	Unlimited ... Limited
	Dynamics	Fixed ... Variable
	Diversity	Poor ... Rich
	Resource availability	Restricted ... Ample

Table 2.1: Characteristics of a multiagent system (Weiss [1999]).

Agent, *interactions* and *environment* represent the major facets in a MAS. To gain a better insight of our system, it could be useful to detail these aspects in finer granularity. Thus, Table 2.1, taken from Weiss [1999], adapted from Huhns and Singh [1998], shows the different characteristics of a MAS.

Even if we could design almost any problem with a multiagent model, the added complexity of such a system may not always be worth the trouble. Usually, some factors hint that developing a MAS for a particular problem would be a good idea (Bond and Gasser [1988], Jennings and Wooldridge [1998]). Firstly, if the environment is inherently open, dynamic, uncertain or complex, MAS are usually indicated. Another factor might be that agents are a natural metaphor, as in auction mechanisms or RobocupRescue[6]. In this kind of system, the environment is naturally modeled by a society of entities. Finally, one of the strongest points for natural modeling of a problem in a MAS is the inherent distribution of said problem. In some environments, data, controls and expertise are naturally distributed. In this kind of distributed problem, a centralised solution is often at best difficult, and impossible in the worst case.

Considering the complexity and natural distribution of the multi-frigate environment in project NEREUS, we believe that a MAS is clearly appropriate to model and work

[6]See: www.rescuesystem.org/robocuprescue/

on this problem.

2.2.1 Interactions between Agents

In MAS, agents are usually different in many aspects, such as their reasoning process, their current beliefs, the way they sense the environment, the actions they can undertake or their goals. In MAS, agents have goals, which have dependency relations among them. The four following types of dependencies were described by Wooldridge [2002].

- *Independence:* There is no dependency among the goals of the agents. The success (or failure) of the goals of one agent has no effect on the goals of the other agent. This relation is bilateral.
- *Unilateral:* The goals of one agent depend on the goals of the second agent. However, the relation is not bilateral, as the goals of the second agent are not affected by the success or failure of the goals of the first agent.
- *Mutual:* Both agents depend on each other with respect to some of their goals, because the same resource has to be used in order to attain the goal or the accomplishment of the goals of one agent will impede the achievement of the goals of the other agent. Therefore, the success or failure of an agent's goals will have an influence on the success of the goals of the other agent, and vice versa.
- *Reciprocal:* The first agent depends on the other for some goal, while the second also depends on the first for some goal (the two goals are not necessarily the same). Note that mutual dependence implies reciprocal dependence. A reciprocal dependency can be considered as a combination of unilateral dependencies.

However, having these kind of relations between the goals of the agents does not mean that they are acknowledged by both or even one agent. Since agents have their own beliefs, it is possible that they do not believe in their dependency.

These dependencies between the goals of agents serve to derive interaction types relating to the way the agents interact among themselves. There are three types of agent interaction recognised:

Coexistence: This relation only happens when the mutual goals are independent. The agents simply do not interact together. They exist in the same environment, but

their goals are completely unrelated and therefore they have no obligation toward each other in any way. Note that having independent goals does not necessarily mean that the agents will not cooperate.

Competition: This relation happens when the goals of the agents are mutually dependant. The agents will have to compete to achieve their own goals.

Cooperation: In this relation, the agents cooperate to facilitate the achievement of their respective goals. This relation is possible when the agents have goals that are unilaterally or reciprocally dependent. The cooperation can take place with or without explicit requests from either agent.

Both competition and cooperation are coordination relations. The main difference is whether the agents are collaborative or not. Therefore, competition usually occurs with agents that are self-interested.

In MAS, two issues are of importance (Wooldridge [2002]). The *degree of coordination* is the extent to which agents avoid extraneous activities, reduce global resource consumption, and keep conflicts to a minimum. The *coherence* is how well the system behaves as a single unit, and is maintained when a certain degree of coordination is conserved by the agents. To achieve this, the agents must be able to determine their goal and coordinate their actions with other agents.

In most real-world systems, coordination must be done in an environment with several constraints, like allowed time, number and size of communications, quality of the plans, etc. To coordinate successfully, agents must have 1) a model of other agents and 2) a model of interactions. This presupposes sociability, as exposed in Section 2.1.

2.2.2 Coordination

Coordination is not only one of the processes of the "life" of an agent; it is rather encompassing every aspect of the agent. Far from a mature domain, coordination in MAS has not even reached a consensus on a static and accepted definition, though there is no real debate on a definition for coordination (in fact, that is part of the problem). For us: *Coordination is the process by which the inter-dependencies between the activities of agents are managed.* Another problem with the study of coordination is that, since it includes every aspect of an agent's design, it is a problem too complex to be apprehended in its completeness.

Without being exhaustive, we present here a list of some facets of the coordination problem.

A first aspect is the representation of mental patterns of agents. This is the first step when modeling an agent, as it allows knowing *what should be done*. These patterns are usually expressed in terms of goals and Beliefs, Desires, Intentions (BDI) architectures. Another aspect is the internal modeling of other agents. This allows determining *with whom to coordinate* and *what type of coordination will be done* (from competition to complete cooperation). Then, the agent has to determine communication languages to determine *how messages will be exchanged*. To know *what to communicate* and *when to use these communications*, agents will have to define communication protocols, using conventions or predefined sequences of actions. By defining commitments and applicable sanctions for the non-respect of these commitments, agents with conflicting goals (i.e. competing agents) will be able to *take and manage engagements* to coordinate their interactions. On the other side, cooperating agents will have to obtain a common view of their goals and intentions to be able to *form teams*. In teams, they will be able to act together toward a common objective. This can be done by sharing and *coordinating the tasks among agents* or by sharing and *coordinating on the end results*. There are multiple ways of *distributing the tasks among agents*; in the case of cooperative agents, this particular problem is known as the connectivity problem.

Most of these different sides of coordination are currently under research by many known authors and we can see numerous attempts to come up with a united theory. This is encouraging as we can hope to eventually see the emergence of an integrated coordination theory for the future of MAS.

Coordination mechanisms are usually separated in three different types, as shown in Figure 2.2, which are the solutions based on communication, conventions (or social laws) and learning (Boutilier [1996]). Solutions based on learning can use communications, conventions or both. However, these solutions are distinguished by the fact that the strategies used are learned by the agents over the situations they encounter rather than decided at design time, as in the two other classes of coordination mechanisms.

We cannot compare the work of different authors, as most are not aimed at the same aspect of coordination. However, we will still expose some of the latest researches while specifying to which aspect of coordination they are relevant. The first aspect (*what should be done*), is not specific to MAS, but is a problem that AI researchers have been working on for many years. Thus, even though agents situated in multiagent environment need to deliberate and choose their course of action, those reasoning processes will not be further detailed in this chapter. Many authors (as Russell and Norvig [2003])

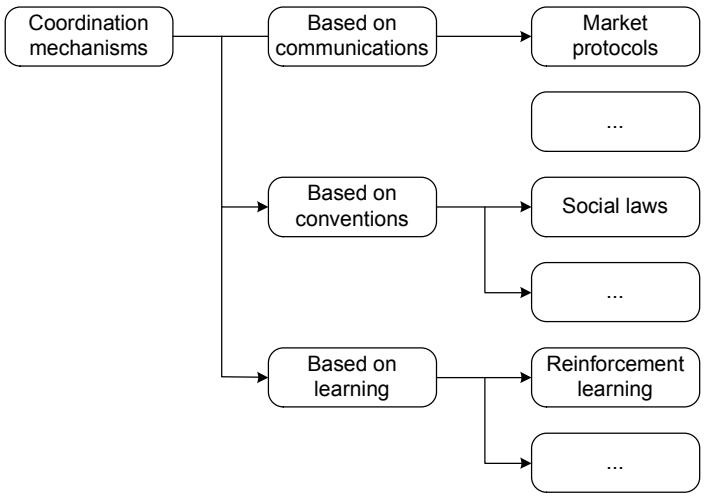

Figure 2.2: Taxonomy of coordination mechanisms.

speak abundantly of such aspects.

Cognitive Coherence

Even though agent techniques have been around for some years now, there is still work undergone to improve the mental patterns of agents. This section presents the work from Pasquier and Chaib-draa [2003], which, in regard to the coordination aspects presented earlier, is aimed more specifically at the sub-problems of *what to communicate*, *with who to coordinate*, and *when to use communications*.

Most of the recent work on communication focuses on interactional theories, which articulate around the notion of structural coherence, or conversational coherence, and deal with the structure of communication. On the other hand, cognitive coherence theories deal with message production as well as message perception and reception. This gives two different coherence dimensions, the respect of dialogue structural constraints and the agents' cognitive satisfaction. Those two dimensions are often connected and working on a cognitive theory does not allow denying the need for an interactional theory.

In cognitive sciences, cognitions gather all cognitive elements: perceptions, propositional attitudes such as beliefs, desires and intentions, feelings and emotional constituents, as well as social commitments. From the set of all cognitions result attitudes which are positive or negative psychological dispositions towards a concrete or abstract object or behaviour. All attitude theories, also called cognitive coherence theories, ap-

peal to the concept of homeostasis, i.e. the human faculty to maintain or restore some physiological or psychological constants despite the outside environment variations. All these theories share as a premise the coherence principle, which puts coherence as the main organising mechanism: the individual is more satisfied with coherence than with incoherence. The individual forms an open system whose purpose is to maintain coherence as much as possible. Attitude changes result from this principle in incoherent cases.

The cognitive dissonance theory, initially presented by Festinger [1957] is one of the most important theories of social psychology. Pasquier and Chaib-draa [2003] proposed a model adapted for AI and MAS. In this model, elements are agent's cognitions: beliefs, desires, intentions and social commitments. They are divided in two sets: accepted elements (which are interpreted as true, activated or invalid, according to the type of elements) and rejected elements (which are interpreted as false, inactivated or not valid). Every non-explicitly accepted element is rejected. Two types of binary constraints on these elements are inferred from the pre-existing relations that hold between them in the agent's cognitive model:

- *Positive constraints:* Positive constraints are inferred from coherence or consonance relations, which can be explanation relations, deduction relations, facilitation relations and all other positive associations.
- *Negative constraints:* Negative constraints are inferred from incoherence or dissonance relations: mutual exclusion, incompatibility, inconsistent and all the negative relations.

These constraints can be satisfied or not; a positive constraint is satisfied if and only if the two elements that it binds are both accepted or both rejected. On the contrary, a negative constraint is satisfied if and only if one of the two elements that it binds is accepted and the other rejected. Therefore, two elements are said to be coherent if they are connected by a relation to which a satisfied constraint corresponds. Conversely, two elements are said to be incoherent if and only if they are connected by a relation to which a non-satisfied constraint corresponds. For each of these constraints, a weight reflecting the importance and validity degree for the underlying relation is attributed.

One can measure the coherence and incoherence degree of a single element or a set of elements by calculating the sum of the weights of the satisfied incoming constraints, divided by the number of concerned constraints. The basic hypothesis of the cognitive dissonance theory is that incoherence (what Festinger names dissonance) produces for the agent a tension that incites him to change. The more intense the incoherence, the

stronger are the dissatisfaction and the motivation to reduce it. A cognition incoherence degree can be reduced by: 1) abolishing or reducing the importance of incoherent cognitions, 2) adding or increasing the importance of coherent cognitions. To reduce incoherence, an individual is going to either change his cognitions or try to change those of others. Thus, this theory can be used to decide what to communicate, with whom and when. Furthermore, it provides the mental frameworks for communications such as arguing and negotiating.

Market Protocols

An approach used by many for coordinating multiple agents is to view the MAS as an economic market (Malone [1987], Wellman [1993], Huberman and Hogg [1995]). This section, as the ones on Social Laws and PGP, is inspired by Excelente Toledo [2003] who describes market protocols and social laws as a coordination mechanism for multiagent systems. While not being only confined to it, the particular work domain of market mechanisms can be viewed as *taking engagements*.

The basic idea of market protocols is that agents play the roles of sellers and buyers in a market fashion to agree about the price of a service or a task. There are basically two elements in this kind of system: 1) the role of the agents that take part in the process and 2) the market structure itself. The former consists of the agents that buy or sell goods by responding to the changes in the price (the actual response is directed by their particular preferences). The latter is the interaction protocol (the coordinating algorithm) that establishes the rules of how the participants agree about the price of the good.

To this end, the most well known market structures take the form of auction houses (Sandholm et al. [1999], Wurman et al. [2001], He et al. [2003]), in which the type of auction indicates a prescribed guide of how the bids are treated. However, although the protocol specifies the rules of how agents can bid, it is clear that agents still have to consider their preferences to decide how to actually bid.

Social Laws

The Social Laws approach is a way used by some authors to coordinate complex multiagent systems (Shoham and Tennenholtz [1992], Briggs and Cook [1995], Barbuceanu et al. [1998]). Compared to market protocols, the *rules* of the environment are fixed, while we

focus on *managing engagements* (in regards to the explanations given earlier). Therefore, the challenge is to define correct rules and norms that will make the correct behaviour an emerging characteristic of the MAS. The most interesting challenge of Social Laws approaches is to make such rules emerge directly from the behaviours of the agents, not by static design at system definition.

Social laws are a mean of coordinating systems in which there are a large number of interactions among agents. The general idea is that agents are designed to follow local rules of behaviour, leading them to act in coordinated ways. Agents plan their actions but their decisions about which goals to pursue at which time are ruled by the social conventions.

In any given situation, agents identify their current state, and only a subset of the possible actions that the agents could follow are designated as legal according to the social laws in force. Flexible laws can also be used, where there is an associated degree of strictness to the laws. The more restrictive the laws, the fewer actions that are possible. This principle can be extended to include obligations to the restricting laws. In this case, the agent might have to *add* actions to their planning in order to respect the conventions. In any case, the use of laws and conventions orients the decision-making of individual agents in a certain way. When reporting this concept in a society of agents, a good design will bring out emergence of behaviours in the system.

When comparing market protocols and social laws, it can be seen that in the former case the requirements that are needed to coordinate agents' activities are mostly performed at run-time, whereas in the latter case the interactions are defined by a system designer. An appropriate way to coordinate could use market mechanisms to contract engagements and social laws to enforce them, which would results in the desired coordination.

Team (Re)Formation

There is currently much work on teams and team formation (Cohen et al. [1997], Tambe [1997], Nair et al. [2003], Pynadath and Tambe [2003]). A team can be defined as a group of agents with common goals. As such, the problems considering teams can be viewed as a subset of the coordination problem in MAS. Usually, agents in a team are more disinterested as they concede some importance to the common welfare or the accomplishment of team goals.

Incomplete or incorrect knowledge, due to constrained sensing and uncertainty in

the environment, often motivates the need for agents to explicitly work in teams. A key precursor to teamwork is team formation; how to organise the agents into collaborating teams to perform the tasks that arise.

In dynamic multiagent environments, teams must be formed rapidly so tasks are performed within given deadlines. Teams must be reformed in response to the dynamic appearance or disappearance of tasks. As the tasks change, or members of the team fail, a team needs to evolve to handle the changes. Clearly, the configuration of agents is relevant to how quickly and well they can be reorganised in the future. Each reorganisation of the teams should be such that the resulting team is effective at performing the existing tasks but also able to adapt quickly to new scenarios. This is what is referred as *Team Formation for Reformation*.

Some approaches, as the one from Ambroszkiewicz et al. [1998], rely on the fact that the team is expanding locally, starting with a single agent. If the common goal of the expanding team becomes inconsistent according to the knowledge acquired from the dynamically changing environment, then the team starts shrinking by removing some of its members. As soon as the expanding team becomes complete (i.e., it has all the resources and capabilities needed to achieve the common goal), the members of the team perform the cooperative work. Another interesting feature of this approach is that the plan of the final cooperative work of the team is constructed and revised gradually during its formation.

Contract Net

This kind of protocol, if not used with cooperative agents, is simply another market protocol. However, in the case where a collaborative team is composed of agents that are not self-interested, this becomes an efficient mechanism for *distributing the tasks among agents*. This section exposes the Contract Net protocol, as defined by the Foundation for Intelligent Physical Agents (FIPA) (The Foundation for Intelligent Physical Agents [2002a]).

The Contract Net Interaction Protocol (IP) was first proposed by Smith [1980] and comprises one agent (the initiator) wishing to have some task performed by one or more other agents (the participants). The FIPA Contract Net interaction protocol is a minor modification of the original contract net interaction protocol pattern in that it adds rejection and confirmation communicative acts. The representation of this interaction protocol is given in Figure 2.3, which is based on Agent UML (AUML) (Odell et al. [2001]). In this figure, m is the number of call for proposals (`cfp`) sent, n is the number

of agents that answer before the deadline, and k is the number of proposals chosen by the initiator.

We see in this protocol that the initiator first solicits proposals (or bids) from other agents by issuing a `cfp`, which specifies the task, as well as any conditions the initiator is placing upon the execution of the task. Participants receiving the call for proposals are viewed as potential contractors and are able to generate responses.

The returned proposal includes the preconditions that the participant is setting out for the task, which may be the price, time of execution, etc. Once the deadline passes, the initiator evaluates the received proposals and selects agents to perform the task; one, several or no agents may be chosen. The selected agents will be sent an acceptance message while remaining agents will receive a rejection message. The proposals are binding on the participant, so that once the initiator accepts the proposal, the participant acquires a commitment to perform the task. A completion message is sent to the initiator once the task has been completed by the participant. On the other side, the participant sends a failure message in the case that it fails to accomplish the task.

Note that the basic Contract Net interaction protocol would require the initiator to know when it has received all replies. In the case that a participant fails to reply with either a `propose` or a `refuse` act, the initiator would potentially be left waiting indefinitely. To guard against this, the call for proposals includes a deadline by which replies should be received by the initiator. Proposals received after the deadline are automatically rejected.

2.2.3 Communication

We presented three coordination approaches in the previous section, one of them being the coordination mechanisms based on communications. The reader can note that we presented many aspects of coordination and detailed them, with the exception of one facet: *how messages will be exchanged*. Most coordination mechanisms will require, at least once, communications among the coordinating agents; failing to do so would result in an uncoordinated situation. In fact, in most mechanisms, we will use more than one communication in the complete coordination process.

Obviously, there is need for an agent communication model that represents the flow of knowledge, and attitudes about such knowledge, within the agent community. An Agent Communication Language (ACL) provides language primitives that implement

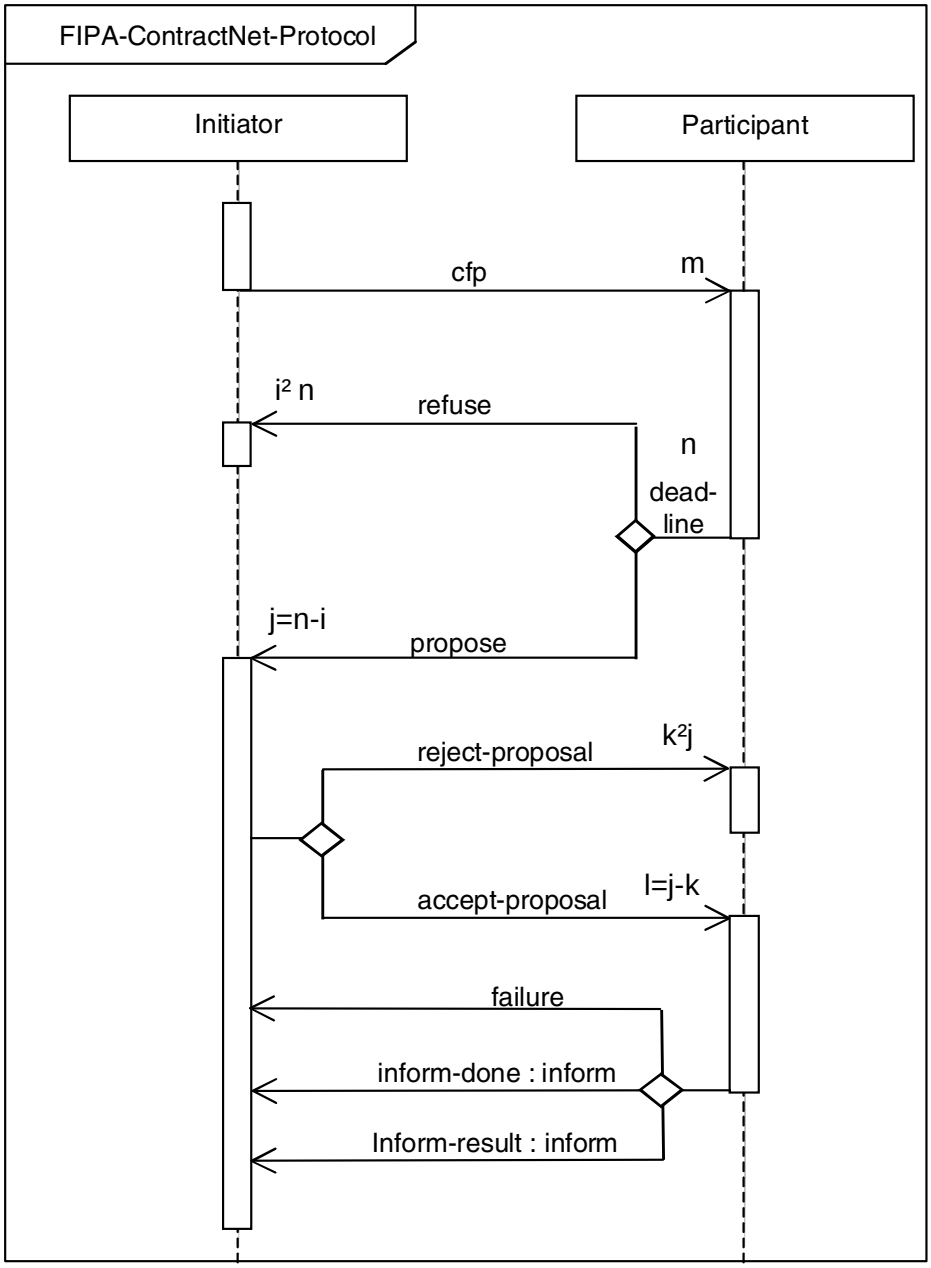

Figure 2.3: FIPA Contract Net interaction protocol (The Foundation for Intelligent Physical Agents [2002a]).

the agent communication model. ACLs are commonly thought of as wrapper languages in that they implement a knowledge-level communication protocol that is unaware of the choice of content language and ontology specification mechanism.

Nearly all the ACLs derive their language primitives from the linguistic theory of speech acts. Speech act theory categorises *utterances* into different categories depending on the intent of the speaker, the effect on the listener, and any other physical manifestations of the act of uttering the utterance.

While there is still work done on ACLs, one is slowly becoming a standard. This ACL, presented by FIPA, closely resembles the Knowledge Query and Manipulation Language (KQML), as it is designed to work with any content language and any ontology specification approach. The reader interested in more on FIPA-ACL versus KQML can refer to Vasudevan [1998]. We will now explain what the FIPA-ACL is (The Foundation for Intelligent Physical Agents [2002b]).

FIPA-ACL

The Foundation for Intelligent Physical Agents (FIPA) is an international organisation that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications. The intention is to provide conversational logic to agents, thus raising the semantic level of agent communication to a higher level than existing technologies. The objectives of standardising the form of a FIPA-compliant ACL message are: 1) to help ensure interoperability by providing a standard set of ACL message structure and 2) to provide a well-defined process for maintaining this set.

In order to achieve this, each of the FIPA-ACL communication primitives, called *communicative acts*, is given a precise semantics by providing pre- and post-conditions expressed in a first order modal logic. With this semantics, the agent is able to express his personal attitude (e.g. belief, desire, uncertainty, choice, intention) towards his achieved knowledge rather than the semantics of the knowledge itself. Based on this underlying semantic model, the agent can compile sensible options for his next action.

A FIPA-ACL message contains a set of one or more message parameters. Precisely which parameters are needed for effective agent communication will vary according to the situation; the only parameter that is mandatory in all ACL messages is the `performative`, although it is expected that most ACL messages will also contain sender, receiver and content parameters. If an agent does not recognise or is unable to process

one or more of the parameters, or parameter values, it can reply with the appropriate `not-understood` message.

FIPA-ACL does not place constraints on the content language itself (how the content of a message is expressed). It only provides the conversation envelope for the actual information being exchanged.

The following is an example of a FIPA-ACL message applied to the specific problem of Contract Net in project NEREUS.

```
(cfp
  :sender (agent-identifier :name agent-1)
  :receiver (set (agent-identifier :name agent-3))
  :content
    "(action (agent-identifier :name agent-3)
      (engage threat-4))"
  :ontology nereus-contract-net
  :language nereus-lang)
```

FIPA-ACL, as KQML, is an approach based on mental attitudes and intentions. This kind of approach has been criticised (Maudet and Chaib-draa [2002], Singh [2000]) as it as two major drawbacks:

1. The inferences are very complex, as agents have to model beliefs, desires and intentions of other agents.
2. The message semantics use private mental states, which cannot be verified since we do not have access to mental states of other agents. Thus, we often must suppose the sincerity of the interlocutors, which is deemed too constraining by some authors (Dignum and Greaves [2000]).

Another approach, based on dialogue games, has been developed at DAMAS (Labrie [2004]). The language developed, called DIAlogue-Game based Agent Language (DIAGAL) uses game structures allowing the agents to engage in dialogues. This approach focuses on the notion of social commitments and sanctions. More information on DIAGAL can be found at: www.damas.ift.ulaval.ca/projet.en.php.

2.3 Summary

We have seen in this chapter what an agent is, how it is defined and the importance of its relation to the environment in which it is situated. We have also seen what a MAS is and how the agents are related together.

In a project such as NEREUS, the real-time constraints are very important and must be considered before and while developing agents that will evolve in this system. Indeed, agents situated in such an environment cannot deliberate without considering

deadlines and timeliness of solutions. The next chapter will describe real-time systems and agent planning in such systems.

Chapter 3

Agent Planning in Real-Time Systems

As presented earlier, we focused our researches on the coordination of agents in real-time systems. However, coordinating agents still have to deliberate and plan their actions. In fact, the coordination acts must be planned, as well as the individual plans that must be coordinated.

Planning in real-time is an arduous problem and must be specifically addressed. As explained by Ash and Dabija [2000], real-time domains are fields in which an agent has a limited amount of time to produce an output, and is generally better off coming up with a good, but necessarily imperfect, solution by a deadline rather than failing to act. The agent has some information about its deadline, but without necessarily knowing when it will occur. With their inherent complexity, real-time domains let wide open the possibilities for experimentation in research domains such as AI. However, this also has a severe drawback since the developed solutions have to be able to grasp those levels of complexity. Another problem when increasing the complexity of a system is the added difficulty to validate specific details of a solution. Indeed, since most aspects are closely interrelated, it is hard to isolate and validate specific aspects of a solution.

3.1 Real-Time Systems

According to Gillies [2004], a real-time system is one in which the correctness of the computations not only depends upon the *logical correctness* of the computation but also

upon the *time at which the result is produced*. If the timing constraints of the system are not met, system failure is said to have occurred.

Defining the logical and temporal correctness (Gillies [2004]):

- *Logical correctness:* The system outputs adequate results, as a function of the inputs, ensuring the desired behaviour of the system.
- *Temporal correctness:* The system respects time constraints by outputting results before the deadlines.

In other words, it is not only the quality of the obtained solution that counts, but when we got it. It is useful to note that this happens because the environment is in constant evolution. A static system is not a real-time system since there is no difference between two states, unless an action is done in the environment. Thus, a solution in a static system will be as good even if it is delayed. Considering the preceding canonical definition, it is easy to find many examples of real-time systems in our daily life.

- *Situation 1:* An automated system informs stockholders of changes in the price of actions they asked to watch.
- *Situation 2:* An airport system controls the deviation of flights whenever two airplanes are engaged in the same flight corridor and there is a collision risk.

Intuitively, these two situations are far different as much for their criticality as for their domains. Intuitively, the second situation seems a lot more pressing than the first one. How can we distinguish these real-time systems?

3.1.1 Categories

By accepted definition, a deadline is a time limit, as for completion of an assignment[1]. Deadlines can be categorised in three classes (Hiller [1998]): soft, firm and hard deadlines.

Soft deadlines: A deadline is said to be soft if the usefulness of the results produced by the corresponding task decreases over time after the deadline has expired. Here

[1] The American Heritage Dictionary of the English Language: Fourth Edition [2004]

is an example of a soft deadline: suppose an ASM is relatively far and coming toward a frigate. A solution envisaged is to position the ship as to reduce the Radar Cross Section (RCS). This serves to reduce the possibility of a lock from the threat. The soft deadline is when the frigate comes in the range of the threat's radar, since after that, each passing second increases the chances that the threat locks on the frigate.

Firm deadlines: A deadline is said to be firm if the results produced cease to be useful as soon as the deadline expires, but consequences of not meeting the deadline are not very severe. The deadlines of many aperiodic tasks belong to this category. An example of a firm deadline: taking the situation presented for the soft deadline, but the threat having succeeded in locking onto the frigate. A solution envisaged is to use a jamming antenna to deceive the incoming threat. The firm deadline of this action is the moment at which it is too late to start the action. Giving this solution after the deadline has no utility, considering the jamming systems, since there is not enough time left to bring the action to completion. However, the deadline of this action is not catastrophic since there is enough time to provide other solutions. Thus, the deadline *of this action* is firm, not hard.

Hard deadlines: A hard deadline is a firm deadline that can result in catastrophic consequences if missed. Periodic tasks usually have deadlines of this kind. An example of a hard deadline: again taking the situation described for the soft deadline, but the threat is now closer and the jamming system failed to deceive the threat. A solution is to engage the threat with the CIWS. In this case, the hard deadline is the latest time at which the order to fire the CIWS can be issued to effectively intercept the threat. Indeed, not providing this solution before the deadline is catastrophic: there is no other system left to seduce/deceive/destroy the incoming ASM; it will hit the frigate.

However, we believe that firm deadlines are merely a subset of hard deadlines; they differ only in the consequences of failing to act on time. Figure 3.1 represents soft deadlines, while Figure 3.2 represents hard and firm deadlines. We see in these figures that the soft deadline causes a gradual degradation of the quality of the solution, while the solution goes instantly from a good quality to a null quality when encountering a hard deadline.

These different deadlines can now help us characterise the two systems shown earlier in Section 3.1. Obviously, situation 1 represent a soft real-time system, because missing the deadline reduces the usefulness of the results, since the prices will have moved and the results will not be as interesting, but still of interest. The second system is clearly

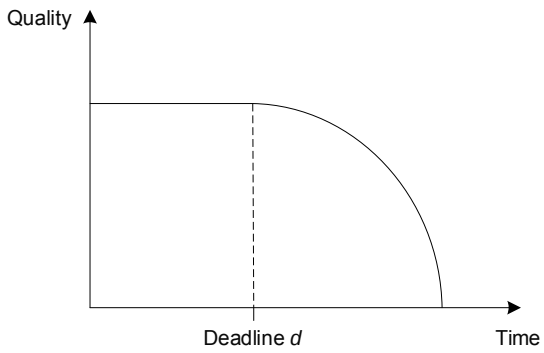

Figure 3.1: Soft deadline.

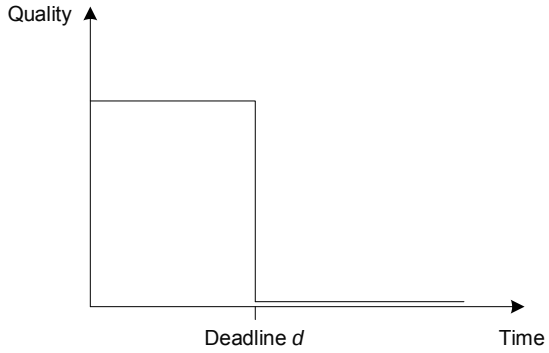

Figure 3.2: Hard/firm deadlines.

a hard real-time system, as missing the deadline will result in a direct collision of two planes and the death of hundreds of persons.

As we have seen, we are obviously working in a hard real-time system in project NEREUS. Not obtaining a solution in the given time limit will cause a threat to reach the frigate unimpeded, maybe sinking the ship and killing many crew members.

3.1.2 Real-Time Systems Structure

Figure 3.3 presents a generic architecture for a real-time system. In a real-time system, the *environment* is primordial since the actions, with associated deadlines, are effectuated in this environment. *Sensors* continuously receive and monitor information about the current state of the environment. This is the normal input for the *control system*, which uses this data to produce a *view* for external *operators* (users) via a graphic user interface. The commands of users, which could be suggestion, imposition or prohibition of operations, are entered as inputs for the control system. Afterwards, the control system, having decided of a course of action, commands the *actuators* to act on the environment.

3.2 Planning

Having defined real-time systems, we can now look into agents planning in these systems. However, we will need to introduce some definitions before continuing:

Reacting: Reacting is the use of predefined sequence of actions, in reaction to a par-

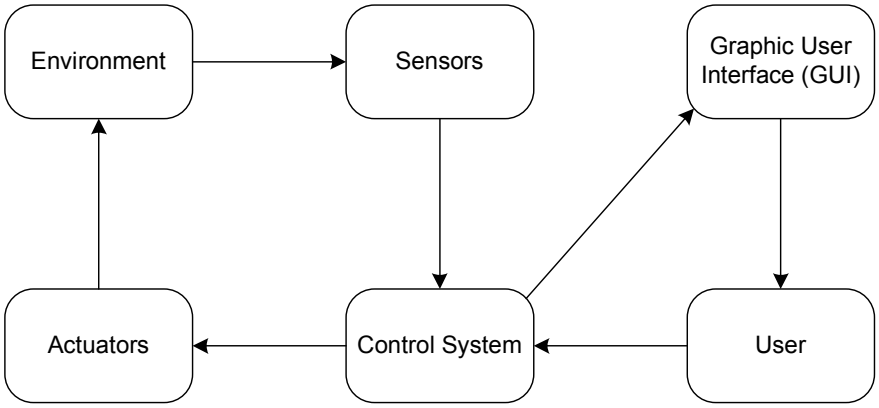

Figure 3.3: Architecture of a real-time system.

ticular state environment.

Planning: Planning is the design of a sequence of actions to carry out a particular task or achieve a particular goal.

Planning is the AI domain that allows, in regards to the aspects of coordination defined in Section 2.2.2, to answer the question as to *what should the agent do? What are the actions required to attain the goals*?

3.2.1 Contingency Space

A *contingency* is any state of the world, entered by the executing agent while following a plan, which should not have occurred as a result of executing the plan up to that point (Ash and Dabija [2000]). Contingency occurs because of unpredictability, either in the environment or in the agent's execution subsystems. In the real world, the possibilities of contingencies are unlimited, and the contingency space represents the set of possible contingencies of a particular domain. An ideal planner should be able to develop a plan for all contingencies. Of course, building a plan considering unlimited possibilities within a limited time span is quite infeasible. Luckily, most of these contingencies can be ignored when planning.

We can characterise the different contingencies by considering some of their aspects. Firstly, some are definitely more probable than others. In a naval warfare environment, there is some (not negligible) chance that a new threat might appear in a certain bearing at any moment. However, the chance that a meteor might strike the frigate is quite remote. Secondly, we can categorise contingencies by the set of possible actions to take

under this circumstance. A frigate *can* react with the full range of its resources against a new threat, while there is definitely fewer actions it can undertake to counter a thirty-foot wave coming at it. Finally, we can discriminate the contingencies along a third line: the upcoming results of an untreated contingency. An unhindered threat coming right at the frigate most probably means destruction, while not taking any actions when detecting a whale in near proximity will probably not endanger the frigate in any case. Another point worth mentioning is that contingencies do not have negative impacts by definition; some contingencies might even have positive consequences. A threat can be disoriented because of an unpredicted meteorological phenomenon, causing its self-destruction. This contingency is, indeed, good news as far as we are concerned.

Note that there are some important questions in planning, among them: *what is to be considered when planning* and *when to stop planning*? The problem of answering such questions is complex in itself and there are still discussions on this topic. In regards to planning, there are three types of contingencies worth considering:

- *Treated in planning:* Contingencies the agent must plan on and that are added in a complete conditional plan.
- *Treated by reacting:* Contingencies that can be reacted on by the use of pre-computed action sequences.
- *Treated by replanning:* Contingencies that can be left aside in the planning process. In this case, we decide that the small probability of this occurrence and broad latitude to act on this situation are not worth the additional time to consider it in a plan. Should this particular contingency happen, the agent would create a patch for the current plan, create a completely new plan from scratch, or leave the plan as it is. This is of course decided by the severity of the situation and the availability of resources.

3.2.2 Plan Representations

Since we just defined contingencies, we can now describe the representation of agents' plans. A plan will be constituted of actions and evaluation nodes. Since contingency nodes are evaluations about the environment, they can be viewed as decision nodes. Thus, we can arrange these nodes in a decision tree, since we cannot know with certainty if a missile, or any other weapon, will destroy or not its target. A decision tree makes it possible to provide sub-plans for each possible result of weapon engagements.

An example of a plan with contingencies is presented in Figure 3.4. The temporal constraints were deliberately omitted in this figure for the sake of simplicity. This simple plan presents a situation where a frigate engages two incoming threats. It will first use a SAM against the first threat. Then, depending on the results of the first SAM engagement, the frigate will either reengage the first threat or engage the second.

3.2.3 Anytime Algorithms

An anytime algorithm is an algorithm whose output quality improves gradually over time, so that it has a reasonable decision ready whenever it is interrupted. The principal characteristics of such algorithms are 1) they can be suspended and resumed with negligible overhead, 2) they can be terminated at any time and will return some answer and 3) the answer returned improves in some well-behaved manner as a function of time (Dean and Boddy [1988]). It is the last two of these three characteristics that really distinguish anytime algorithms from more traditional algorithms. Figure 3.5 shows a comparison between an anytime and a classic algorithm. At time T_1, the standard algorithm has no solution, while the anytime algorithm returns a solution of quality Q_1. Though the solution returned is far from an optimal solution, it really is better than no solution at all.

Anytime algorithms must be controlled by a metalevel decision procedure, which decides whether further computation is worthwhile (Hansen and Zilberstein [2001]). How do anytime algorithms relate to real-time systems? We have seen before that real-time systems are concerned not only with the logical qualities of a solution, but also with its temporal nature. That is to say, it is important to receive a timely answer, and in the case of hard real-time systems, missing the deadline might result in a catastrophic situation. Evidently, the uncertainty and unpredictability of most real-time systems cause problems. This, combined with the fact that most environments are only partially observable, often makes it hard to determine *when* that deadline occurs. On one side, we have systems that cannot tolerate taking too much time to answer a problem; on the other side, we have algorithms that can supply an answer at any time. Quite simply, an anytime algorithm ensures not to go past a hard deadline without an answer, whenever it might be.

To better fix ideas, we will now present an anytime algorithm, constructed from a classic algorithm.

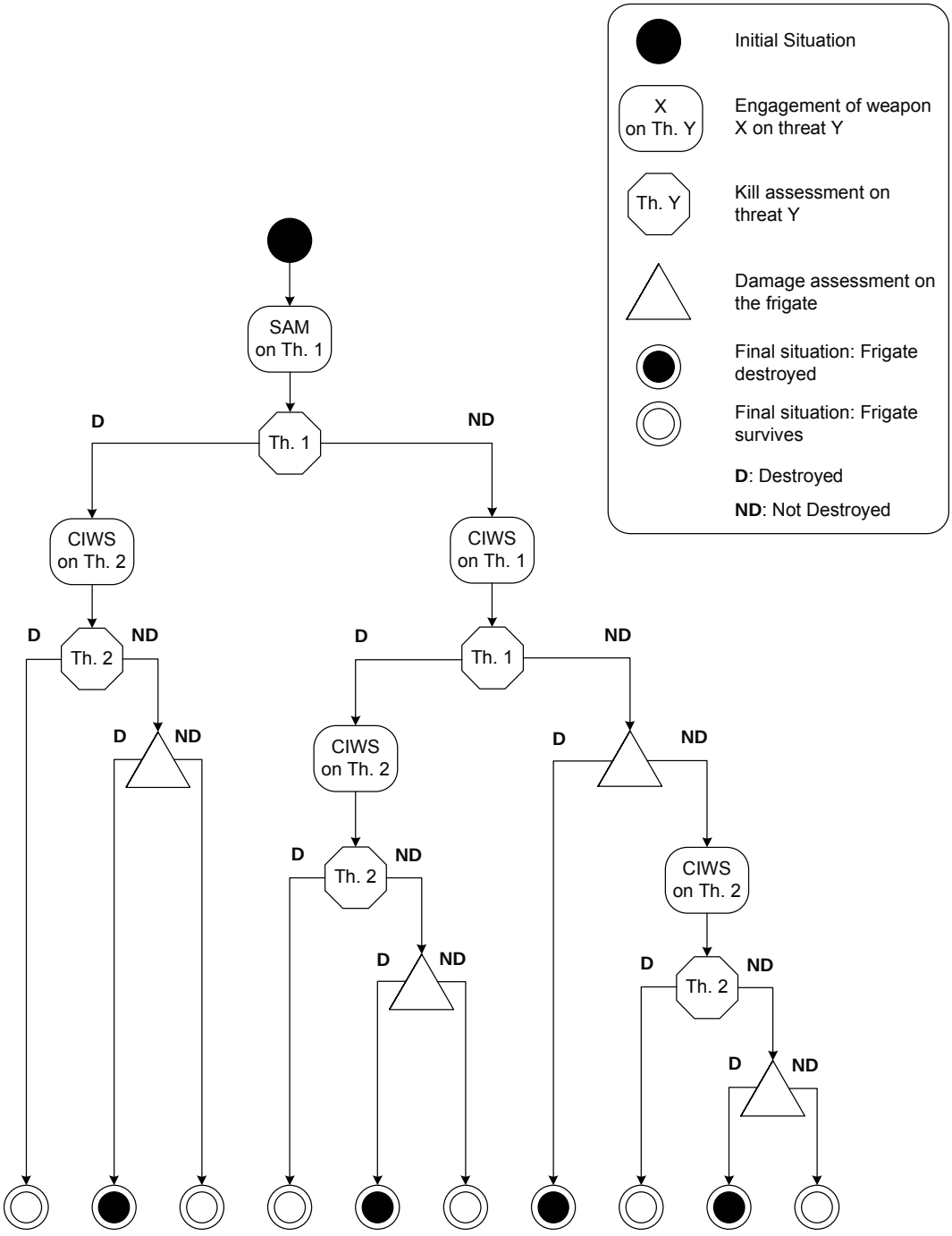

Figure 3.4: Example of a contingency plan.

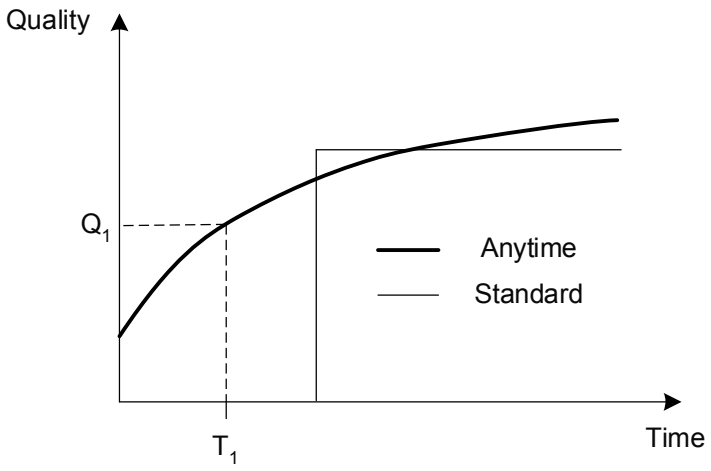

Figure 3.5: Anytime versus classic algorithms.

Anytime A^* Search

Many algorithms can be designed to become anytime. Taking a classic algorithm, the A^* search (Hart et al. [1968]), we will see what needs to be done in this case to make it anytime.

First, let us describe the classic A^* in its standard form. The idea behind the A^* is to provide an optimal outcome to a given search problem without having to actually search all possible paths through the search space.

The A^* algorithm is a well-known and well-studied best-first search algorithm. This algorithm searches outward from the starting node until it reaches the goal node, always expanding the current fringe node that looks the most promising. The value of a node is the addition of the minimum *cost* from the start node to this node, plus the expected remaining cost, evaluated by a simple *heuristic*, as shown in Figure 3.6. The heuristic used plays a crucial role in A^*. An important property of the heuristic is that it must be admissible if A^* is to find the optimal solution. That is, it must always *underestimate* the cost from any node to the goal. However, if the heuristic is optimistic (underestimates very far from the real value), then A^* will expand too many nodes and use too much time before a solution is found. On the other hand, if the heuristic is pessimistic (overestimates), then a solution will be found quickly, but it will almost certainly be suboptimal. The A^* algorithm is explained in Algorithm 3.1.

Consider the problem of finding the shortest possible path, in Figure 3.6, from the root node R to the goal node G. The A^* algorithm estimates the best following node by considering the cost to the fringe nodes and the evaluations of the total left to incur.

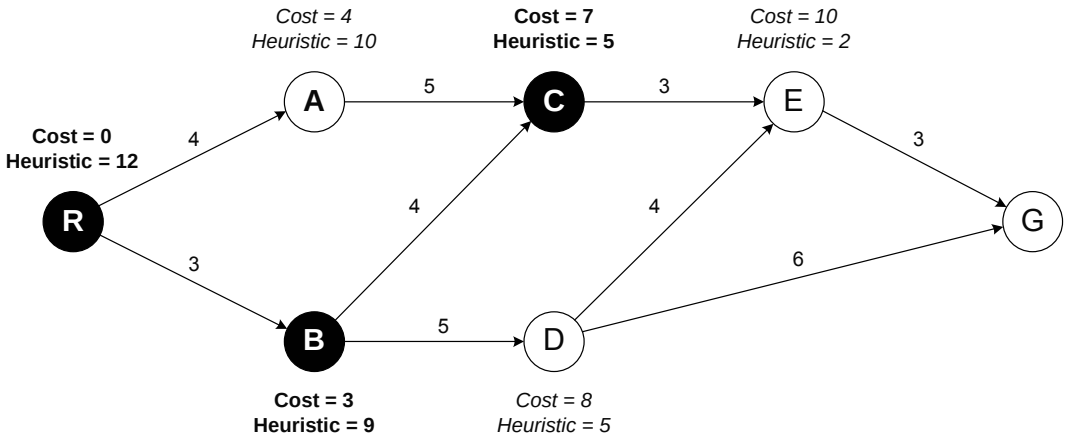

Figure 3.6: Search with A^*.

In the particular case where nodes R, B and C where developed, node E would be the next node to be developed, with a total value of 12 (*10 + 2*).

This algorithm is not anytime. It runs until it finds an answer, which is the path with the lowest cost. However, it might be preferable that the algorithm finds a good first step rather than spending a great deal of time finding the optimal solution. To transform this algorithm into an *Anytime* A^* (Algorithm 3.2) that terminates upon request, some simple modifications have to be made.

We notice the slight differences (in **bold**) from the previous algorithm. The key element is simply noticing that, at the step indicated by a dagger (†), a node is identified that appears to provide the most promising path to the goal node identified so far; this node is tagged as the *first_step* to the goal. By making the algorithm return, on an anytime basis, this *first_step* in the current path, the algorithm is then transformed into an anytime algorithm. In the example of Figure 3.6, the current path is *R-B-C*, and the *first_step* in that path is *B*.

This kind of algorithm is very interesting in a real-time domain such as anti-air warfare. The trick is to determine a way to adapt usual algorithms into anytime algorithms. Ample discussion of anytime algorithms in project NEREUS is provided by Soucy [2003]. We will present the work of Soucy on anytime approach in project NEREUS in Section 5.3.

Algorithm 3.1 A^* (Root node R) **returns** the optimal path to the goal node G.

```
E ← {}; C ← {R}; COST(R) ← 0;
TOTAL(R) ← HEURISTIC(R); PATH(R) ← R
loop
  if Goal node G ∈ E then
    return PATH(G)
  N ← node ∈ C that minimises TOTAL(N)
  C ← C - N
  E ← E ∩ N
  for all N* connected to N do
    if N* ∉ E then
      if N* ∈ C then
        if COST(N) + COST(N,N*) < COST(N*) then
          COST(N*) ← COST(N) + COST(N,N*)
          TOTAL(N*) ← COST(N*) + HEURISTIC(N*)
          PATH(N*) ← APPEND(PATH(N),N*)
      else
        C ← C ∩ N*
        COST(N*) ← COST(N) + COST(N,N*)
        TOTAL(N*) ← COST(N*) + HEURISTIC(N*)
        PATH(N*) ← APPEND(PATH(N),N*)
```

3.3 Metalevel Reasoning Process

Of course, having an anytime algorithm does not completely solve a real-time problem. Even hard real-time systems also possess soft real-time deadlines. We already agreed that there is an ultimate time that, should no prior actions be taken, would result in an unacceptable situation, or, in other words, a solution of null quality. The point is that, by the time the hard deadline is met (and we stop the algorithm), a soft deadline might have been passed and the solution might have been degrading for a while. That is why a metalevel reasoning process is necessary. The process needs to be able to evaluate the expected utility gained by pursuing the deliberation, versus the loss of quality caused by going over the soft deadline. In the next chapter, we will describe the simulator we designed to develop our anytime algorithms.

Algorithm 3.2 Anytime A* (Root node R) **returns** the first step to take or the optimal path to the goal node G.

first_step $\leftarrow$ **FIRST(PATH(*N**))** {***N** is a random node, connected to *R***}
$E \leftarrow \{\}$; $C \leftarrow \{\text{R}\}$; COST($R$) $\leftarrow 0$;
TOTAL(R) $\leftarrow$ HEURISTIC(R); PATH(R) $\leftarrow R$
repeat
  if Goal node $G \in E$ **then**
    return PATH(G)
    {**The first step to take is FIRST(PATH(*G*))**}
  if $N \neq I$ **then**
    first_step $\leftarrow$ **FIRST(PATH(*N*))**
  $N \leftarrow$ node $\in C$ that minimises TOTAL(N)†
  $C \leftarrow C$ - N
  $E \leftarrow E \cap N$
  for all N^* connected to N **do**
    if $N^* \notin E$ **then**
      if $N^* \in C$ **then**
        if COST(N) + COST(N,N^*) < COST(N^*) **then**
          COST(N^*) $\leftarrow$ COST(N) + COST(N,N^*)
          TOTAL(N^*) $\leftarrow$ COST(N^*) + HEURISTIC(N^*)
          PATH(N^*) $\leftarrow$ APPEND(PATH(N),N^*)
      else
        $C \leftarrow C \cap N^*$
        COST(N^*) $\leftarrow$ COST(N) + COST(N,N^*)
        TOTAL(N^*) $\leftarrow$ COST(N^*) + HEURISTIC(N^*)
        PATH(N^*) $\leftarrow$ APPEND(PATH(N),N^*)
until The deadline is met

Chapter 4

The Naval Defence Simulator

Major projects, be they materials or computer-based, cost considerable sums of money. It is important to be able to know if the invested money is well placed, before even starting the production of the final system. In this case, simulation is a relatively easy way to determine the feasibility of a project. Of course, the sums required to develop such simulation systems are much lower than those required to develop a complete "real" system. It is important to find any major flaws early in the system design. Indeed, if flaws are found later in the development process, repairing them might consume as much as hundreds of times more resources.

In the case of modern AAW, as in project NEREUS, it is impractical to prove a concept without resorting to simulation. Obviously, given the cost of modern weaponry and the sheer number of people involved, it is not practical to develop prototypes for new concepts on real ships. For all these reasons, we have developed a naval AAW simulator, called Naval Defence Simulator (NDS). In fact, NDS is the second simulator developed at DAMAS in the context of project NEREUS.

4.1 Early Simulator

A first simulator (shown in Figure 4.1) was initially developed in the process of understanding the domain and the constraints of the project, and has been very useful while determining and implementing the first planning solutions (Paquet [2001a], Chaib-Draa et al.). However, with the new comprehension gained with the growth of the project, we needed to expand the research through new avenues, which the first

simulator did not permit. Indeed, having been developed iteratively with the initial (and imperfect) comprehension of the domain, the first simulator was not very scalable. The efforts required to further develop and maintain this simulator would have been greater than the development of a new simulator. More details about this simulator can be found in Plamondon [2003].

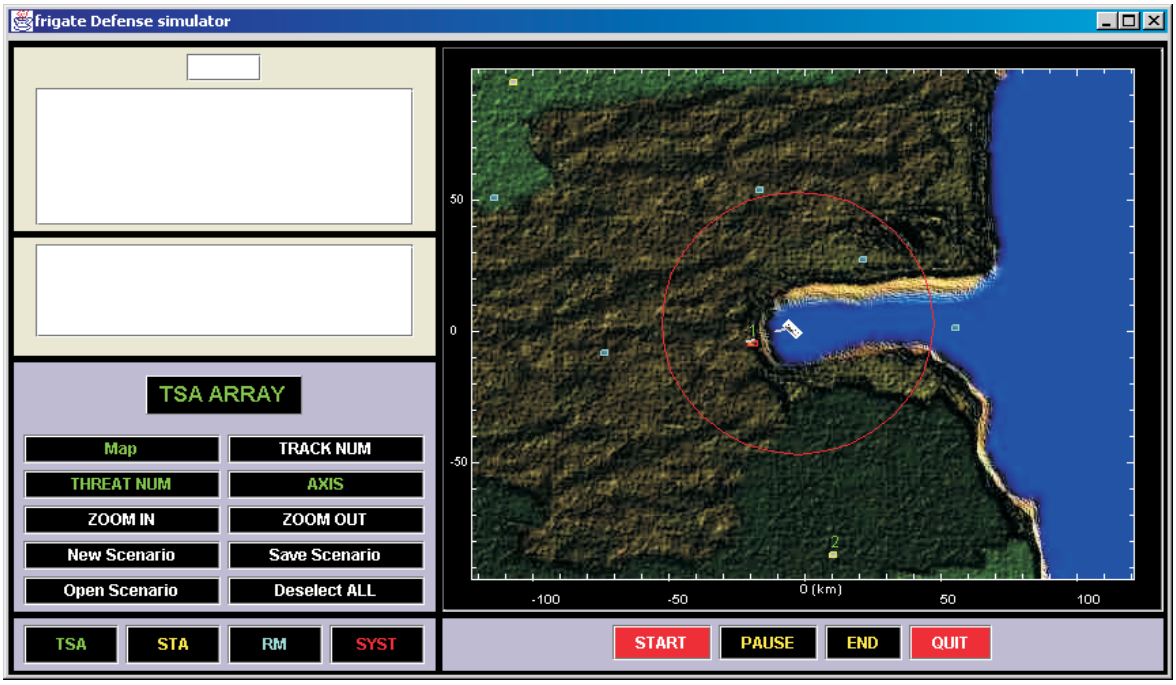

Figure 4.1: The first simulator in project NEREUS (Plamondon [2003]).

4.2 NDS: a New Simulator

The new simulator, NDS (Figure 4.2), is a stable, industrial grade simulator, which entirely complies with the problem specifications. NDS permits a large quantity of tests, including some very complex scenarios. With NDS, we can reproduce specific scenarios a great number of times, limited only by processing power availability. Furthermore, specific tests can be replicated perfectly as many times as desired, which is obviously impossible to match on real-life systems. With low costs compared to real-life demonstrations, this allows us to develop, implement, validate and compare a broad range of concepts. Another advantage of having a simulator is that it allows us to focus on particular aspects of the C2 process. If necessary, we can modify this focus to become wider or narrower. In the particular case of project NEREUS, we focus on resource

management (RM) and coordination between such resources, while leaving aside the aspects of situation and threat assessment (STA) and damage assessment (DA).

Finally, having a visual simulation allows us to visually demonstrate the elaborated concepts. This is particularly useful 1) as a demo via our website[1], 2) as a debugging tool and 3) as a demo showing the latest developments to our partners at LMC and DRDC - Valcartier.

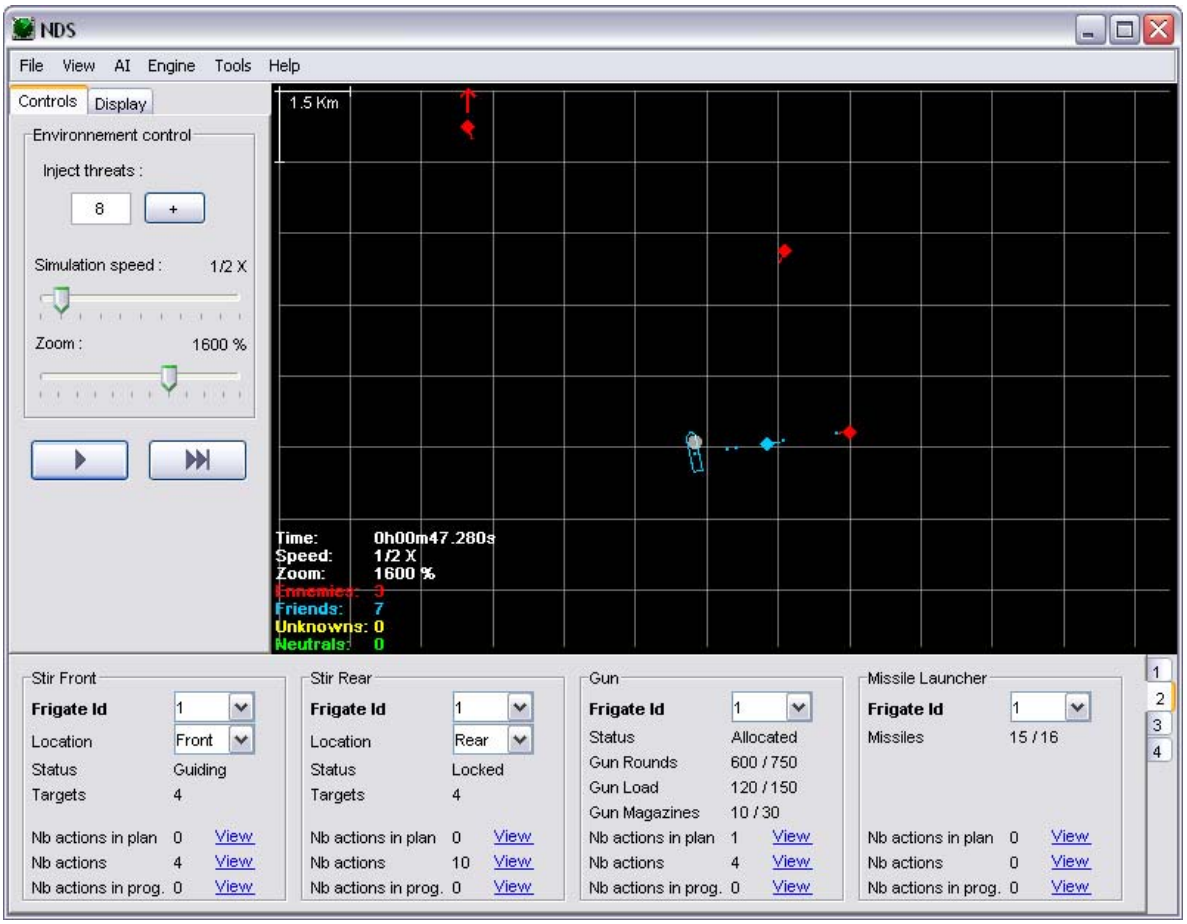

Figure 4.2: The Naval Defence Simulator (NDS).

4.2.1 NDS Architecture

The Naval Defence Simulator is a simulation test bed, developed in three-tier architecture. The programming language used is Java, for its ease of use, flexibility and portability.

[1] www.damas.ift.ulaval.ca/projets/TeamWork/

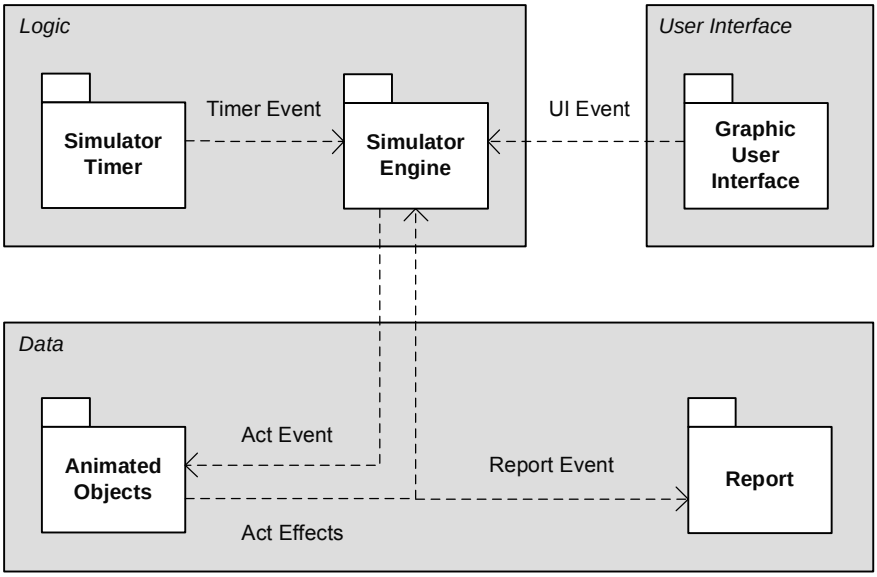

Figure 4.3: Naval defence simulator architecture.

Figure 4.3 shows the architecture of the simulator. The first tier, the *Data*, is composed of the simulation objects. The second tier, the *Logic*, is composed of many subsystems and is responsible for the kinetics, time flow, agents and communications management. When an object needs to be inserted (e.g., when firing a SAM) or deleted (e.g., an ASM has been destroyed), it is the task of the *engine* to evaluate the relevance of the action and take the appropriate steps. The last tier, the *User Interface*, is the medium of interaction between the end users (students and researchers) and the engine. It is with the Graphic User Interface (GUI) that users create and record complex scenarios, get a view of the internal values of objects and start batch tests. Of course, the design of the simulator itself makes it easy to deactivate the GUI and use automated test modes.

NDS design and implementation were driven by two major concerns: extendibility and reusability. We started the analysis with those concerns and they are still primary guidelines in the project. Secondly, since we focus on coordination between agents in a real-time environment, it is also important to describe how time, agents and communications between agents are managed.

Extendibility

Extendibility was the most important point that we considered when we started the development of NDS. As the current project was ending, and a renewal with new con-

straints was to be expected, it was important to be able to expand the simulator to the new specifications of the system. We took special care while introducing specifications and behaviours in NDS; changing the specifications is a matter of minutes. Furthermore, since project NEREUS employs many students, the developments are quite fast and the simulator extendibility is often solicited. As each new package and module is developed, there is always the concern of developing it in a way leaving the most latitude for future developments.

Reusability

Reusability was another focus, since the efforts needed to develop a simulator consume both time and resources. It was important to be sure that the current developments could be used by as many projects at DAMAS as possible. Of course, we also wanted the new simulator to be used by future projects. With this in mind, we developed the simulator in two parts. The first one, the core, is the common simulation package, which is generic and can be used by any other project. The second one is the NDS module, developed exclusively for project NEREUS and built on the simulation core. Currently, two projects other than NEREUS also use the core of the simulator: *Auto21*[2] and *Supply chain*[3].

Auto21: DAMAS is part of the Auto21 network of centres of excellence founded recently by the Natural Sciences and Engineering Research Council of Canada (NSERC) with the goal of strengthening the competitive position of Canada in the automotive industry. Over 200 researchers in 28 universities across Canada are working on the Auto21 program. This project is a completely new initiative in Canada, with the objective of developing an innovative concept called the Collaborative Driving System (CDS) and initiating the development of the associated technologies. DAMAS is taking part in research around *Intelligent systems and sensors*, and more particularly *Coordination and Communication Architecture*. In this project, DAMAS's group aims to define an autonomous driving system that reacts to its automated highway environment, using a guidance system, and coordinates itself with other vehicles using a communication system. The guidance system is used as a reactive vehicle controller that can drive on planned trajectories, while reacting to emergencies. A second deliberative level will use MAS coordination strategies to coordinate the autonomous vehicle with its neighbours, thus maintaining platoon formations on the highway (Hallé et al. [2003a],

[2]For more information, see: www.damas.ift.ulaval.ca/projets/auto21/en/
[3]For more information, see: www.damas.ift.ulaval.ca/~moyaux/travailE.html

Hallé et al. [2003b]).

Supply chain: A supply chain can be defined as a network of autonomous or semi-autonomous business entities collectively responsible for procurement, manufacturing and distribution activities associated with one or more families of related products. At DAMAS, work is done on the supply chain in order to reduce what is known as the *Bullwhip Effect*. The Bullwhip Effect can be described as an amplification of the first mini fluctuations throughout the whole supply chain (Moyaux et al. [2003], Moyaux et al. [2004]).

Time Management

Perhaps the most important choice that we made when analysing how to simulate the frigate problem was the choice to use a discrete time mechanism. In this structure, every object then has, for acting, the same virtual time to act. Considering the hard real-time constraints that we should address, this was a primary concern. The *timer* triggers time events in the engine, which then runs each object for a specific time quantum. Once an object is run by the engine, it acts, deletes impossible actions and moves. After all objects have moved, collisions are evaluated and destroyed objects are cleared from the simulation. While acting, objects look for valid actions in their list to execute. Of course, objects implementing AI take the extra step of planning what actions should be done beforehand. This is illustrated in Figure 4.3 that was presented in Section 4.2.1.

An interesting advantage of this mechanism is that we can easily speed up or slow down the simulation. There are two factors that we can change: the interval at which time events are sent, and the time quantum in which each object must act. Thus, we can virtually speed up the simulation to hundreds of times faster than in real life, but still can let some part execute in real-time when necessary (as with some anytime algorithms[4]). In fact, when sped up to its maximum value, a typical simulation of 5 minutes lasts less than 1 second on the computer used to tests scenarios[5]. Furthermore, the simulator has been designed in such a way that it is possible to vary the simulation speed while leaving the normal CPU time to the planning algorithms.

Moreover, work is currently ongoing to develop a variable step discrete-time engine. This engine will allow different time quantum in the steps taken, so as to further speed up the simulation. This variable step engine will also allow us to attain a greater level of precision in the simulation.

[4] As seen in Section 3.2.3

[5] More details on how the tests were conducted are given in Section 6.2

Agent Management

In the current project, there is exactly one agent for each frigate[6], which is responsible for deliberating on the situation of its attributed ship. In NDS, each object, including agents, act for a specified time quantum each simulation *round*. The inner control loop of the various agents let them monitor their environment and plan on the evolving situation. Moreover, they can receive messages and react at any time during a simulation. These messages are received through the *Communication Central*.

Communication Management

In MAS, cooperating agents often need to exchange messages. In NDS, this is done via the *Communication Central*, which receives messages from agents and dispatches them to the correct recipients. Mostly, it serves to model communication waves in the simulator environment, and accordingly delays the reception of messages by the receiving agents. Three different delays are introduced for each message sent.

- *Message preparation delay:* This is a constant delay, representing the time needed to ready the physical communication channel and prepare the message by wrapping it with the appropriate headers.
- *Distance induced delay:* This is the delay induced by the physical distance between sender and receiver. This is derived from the speed of radio waves, set at 300,000 m/s in our case. Therefore, the beginning of a message sent to an ally 3 km away will be received 10 milliseconds later.
- *Bandwidth induced delay:* This is the delay induced by the total length of the message. By varying this parameter, we can simulate various communication conditions. For example, we can simulate stronger encryption, thus reducing the bandwidth and decreasing the total throughput of the system. Even though this is not yet implemented, we could simulate an appropriate reduction in bandwidth when jamming systems are used, thereby modeling the background noise of the system.

[6]However, it is possible to have more than one agent for each frigate. For example, we could use one agent for hardkill systems and one for softkill systems. We could also have specific agents responsible for multi-platform coordination, etc.

4.2.2 Simulator utilisation

As we have presented earlier, NDS is an environment that we can use to simulate naval battles. Developed using knowledge in graphic user interface design, the simulator interface is intuitive and easy to use. It offers users many features, which will be presented further in this section. Figure 4.4 shows the different sections of the GUI, which we will now present in more detail.

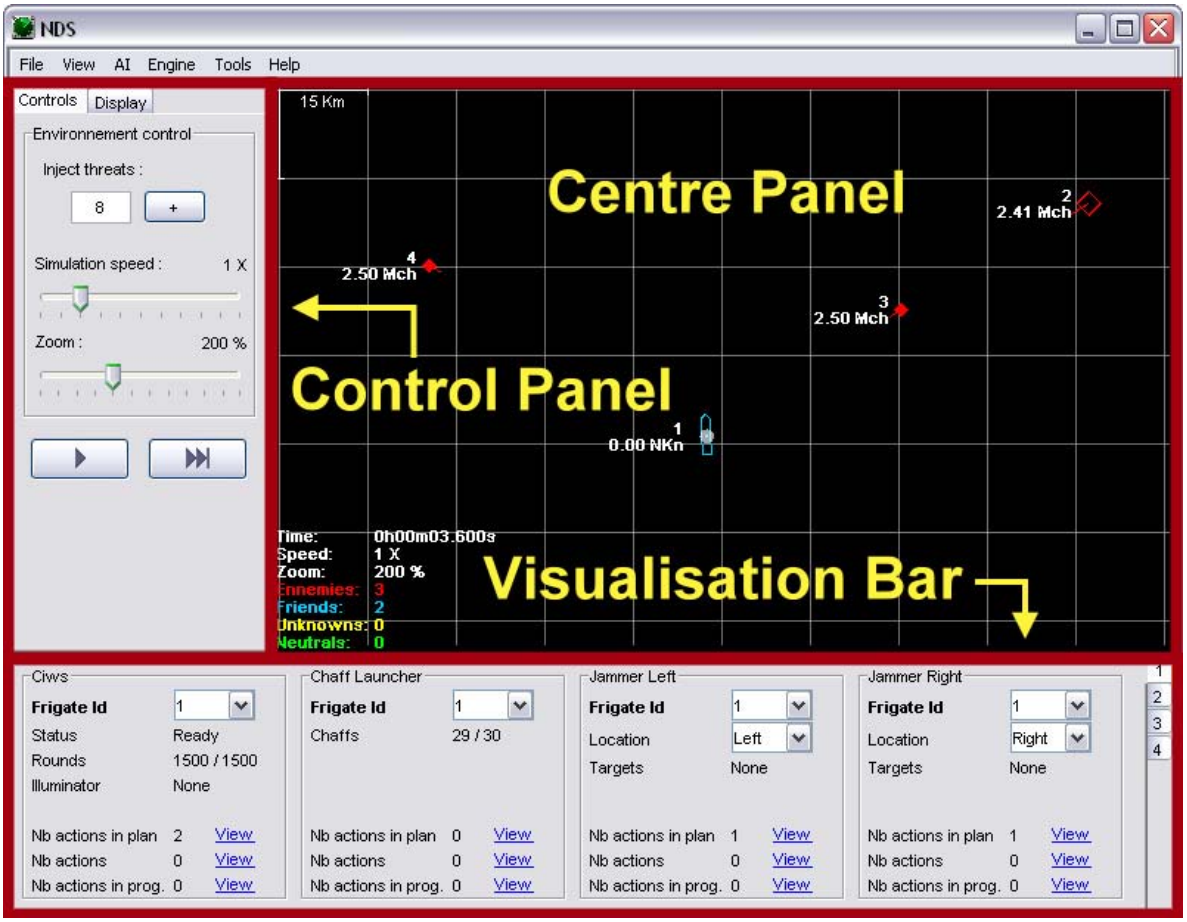

Figure 4.4: Components of the GUI.

Graphic User Interface

The centre panel of the simulator allows users to follow the development of the current simulation, also called *scenario*. It uses symbols and colours to visually represent objects. Table 4.1 shows the different symbols used in the simulation. The colour code (Table 4.2) serves to represent the allegiance of the objects.

Symbol	*Object*
	Cargo vessel
	Frigate
	Airplane
	Missile
	Chaff cloud

Table 4.1: Symbols used in NDS.

Colour	*Allegiance*
Blue	Allies
Red	Enemies
Yellow	Unknown
Green	Neutral

Table 4.2: Colour codes used in NDS.

The following objects, for visibility concerns, derogate from this colour code. First, the cargo vessels are represented in white, to further accentuate the fact that they are units with no actual defensive capability. Second, the chaff clouds are represented as white circles with alpha blending, like the range of the ship systems, which we will see later.

On the left side of the GUI is a zone with two different panels. The first one is the simulation control panel, which contains elements such as speed and zoom controls. The simulation progress can be controlled entirely from this panel. Of course, the simulation progress can also be controlled from items in the engine menu, as from keyboard shortcuts.

In the simulation control panel, a user can:

- Inject new threats, generated at random position, as specified in Appendix A.3.
- Start and pause the simulation at any given time.
- Zoom in and out between 12.5% and 25,600%.
- Speed up or slow down the simulation between 1/4X and 256X.
- Advance the simulation by exactly one turn, which is equal to 80 milliseconds in simulation time.

The left zone also contains another panel, the simulation display panel. In this panel, the user controls how the centre panel will be displayed. Firstly, there are display options common to most objects and some other options applicable only to frigate objects. The elements of the first category allow displaying the unique ID of an object, as well as its speed and position. In the centre panel, gun and CIWS rounds and chaff clouds will not have their information displayed. The reason behind this is that there would be too much information packed in the same space and it would clutter the

display with no appreciable added value. On the other side, the elements specific to frigates allow the user to display simultaneously the range of any onboard system, from radar to CIWS range. Figure 4.5 shows the range/blind zone of the gun as well as the coverage zone of both jammer systems. We can see in the screenshot that the areas where both jammer systems overlap are shown clearer. In fact, this is the superposition of both zones, as alpha blending is used to allow the end user to clearly distinguish every different section of the defended area.

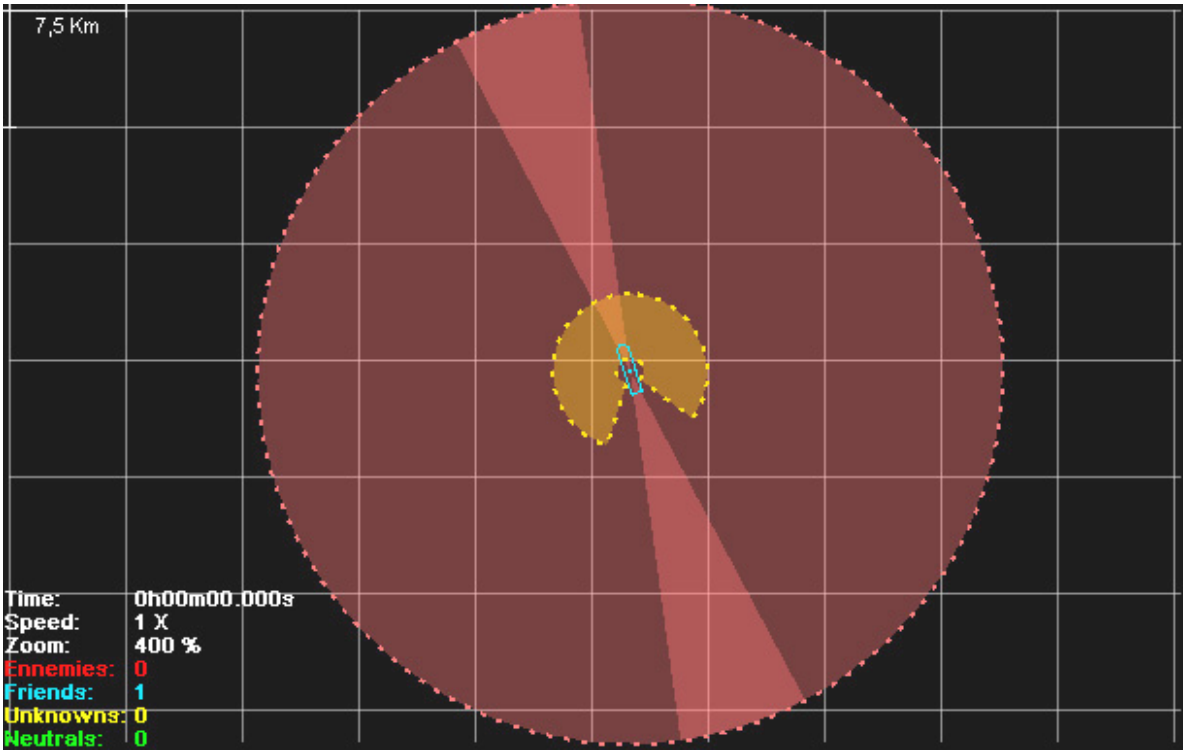

Figure 4.5: Example of system ranges.

The bottom panel of the GUI contains the visualisation bars. There are no real limits on the number of bars that can be present. Each bar is customisable and contains four visualisation items. A right click on any item slot lets the user change what is shown in this slot. As of now, the implemented visualisation items are separated in three types, as depicted in Table 4.3. The first class contains the items pertaining to the simulator core. These items are also present in the *Auto21* and *Supply chains* simulators. The second class of items is the object visualisation items, which present information relative to specific objects and object types. The last kind of visualisation item is specific to NDS and shows the different resource modules of the frigates.

It is easy for a developer to create new visualisation types. Moreover, the settings chosen and the displayed visualisation bars are saved when changed and reloaded when

the simulator is launched again. Figure 4.6 shows a visualisation bar with four visualisation items (in this case a *Frigate View Item*, *CIWS View Item*, *Jammer View Item* and a *Missile View Item* items).

System	*Objects*	*Frigate modules*
Memory Consumption	Airplane	CIWS
System Information	Missile	Gun
	Frigate	Missile Launcher
	Cargo vessel	Chaff Launcher
		Jammer
		STIR

Table 4.3: Available visualisation items in NDS.

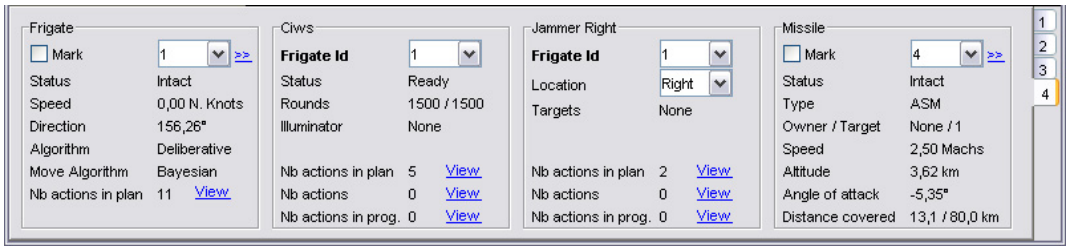

Figure 4.6: Visualisation items.

Functionalities

The panels and the menu in NDS offer a wide range of functionalities. A first functionality of the simulator, facilitated by the discrete time mechanism of the simulator, is the insertion of new objects in real-time. At any given time in the simulation, a user can add new object, without even the need to pause the simulation. At the next round, the object starts acting and moving as usual. This permits the easy creation of very complex and customised scenarios, even at runtime.

Also included is the possibility to choose the behaviour of the agents present in the system. The AI menu lets the user select the planning and movement algorithms as well as coordination modes. This enables the visual demonstration of any algorithm implemented up to now.

At any time in a simulation progress, the user can interrupt the engine and save the current situation in a file. If loaded back, the simulation will start with the same

situation (i.e., the position of objects, their speed, direction, etc.) that was saved. The plans and actions in progress are also saved in this snapshot of the situation. This mechanism is also used in batch tests, when comparing algorithms, to ensure that every algorithm is pitted against the same situation before going to the next scenario. This way, even if running only a small number of tests, we are sure that the results are not biased toward a particular algorithm because it got an easier initial situation.

Another possibility of the simulator is to record a particular scenario and replay it. Everything shown on the centre panel is recorded, including objects added dynamically, and played back on request. While in the replay mode, the user can still control the replay speed and zoom, as well as what is shown in the visualisation bars.

Finally, included for developers only, is the *debug screen*. This screen, available from various places in the GUI, allows the developer to see the exact content of some specific objects. It lets the programmer see the values of the members of this instance (even private ones) as well as the referenced objects and their content. An example of this *debug screen* is shown in Figure 4.7, where we see the details of a planned CIWS fire action.

In this screen, we can see the planned time of execution (at 55.11113 seconds), the hard deadline (at 60.135635 seconds), the list of preconditions that must be met before firing, etc.

Automated Tests

NDS possesses a tool to generate and run a large quantity of tests. This tool, the *Simulation Manager*, can either be accessed within the GUI, or set to start without loading the interface for faster batch tests. Figure 4.8 shows a screenshot of the *Simulation Manager* window used to start tests for scenarios without coordination (single frigate scenarios).

When starting the simulator, the file `tests.cfg` will be read if it is present. In this file there is a flag used to enable or disable automated testing. If automated testing is enabled, the remainder of the configuration file is used to set the testing environment, as will be described later. If this flag is disabled, the configuration file is left unused and the simulator GUI is launched as usual.

We will now describe the parameters that permit tailoring the situations to be tested. The algorithms used for planning and movement will be explained in Chapter

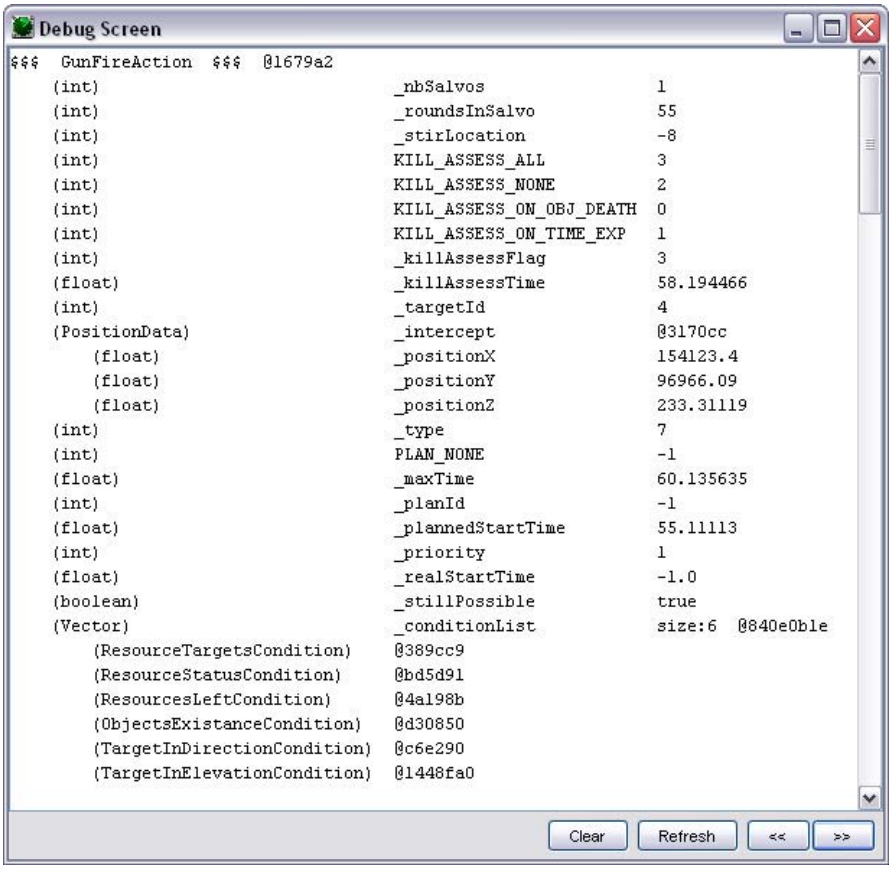

Figure 4.7: Debug screen.

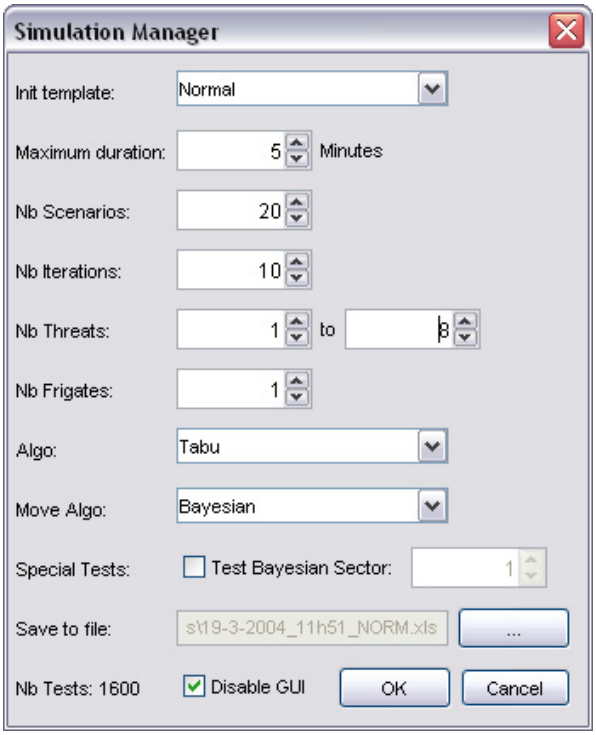

Figure 4.8: Simulation manager.

5, while the parameters useful for coordination mechanisms will be detailed in Section 6.2.

- *Maximum duration:* This controls the maximum duration of any single scenario. If the scenario is not over (There are still ASMs or airplanes with ASMs left in the simulation) when the maximum time specified is elapsed, the scenario is ended.
- *Number of scenarios:* This is the number of different scenarios that will be executed for each combination of parameters (planning algorithm, movement algorithm, coordination mechanism, threat number, etc.).
- *Number of iterations:* This is the number of iterations that will be done for each scenario. This means that, for each combination of parameters, the number of tests to be made will be (*Number of Scenarios*·*Number of Iterations*).
- *Minimum/maximum number of threats:* This is the number of threats present in a scenario. If these two numbers are different, all possible values between the two numbers will be used when generating different combinations.
- *Planning algorithm:* This is the algorithm used for the planning in the tests. The user can chose either 1) to use a single algorithm or 2) to test with every algorithm.

In the case where every algorithm is tested, one combination will be generated for each algorithm available. Planning algorithms are discussed in Section 5.3.

- *Movement algorithm:* This is the algorithm used for movements in the tests. The user can chose either 1) to use a single algorithm or 2) to test with every algorithm. In the case where every algorithm is tested, one combination will be generated for each algorithm available. Movement algorithms are detailed in Section 5.2.

- *Coordination mechanism:* This is the mechanism used for multiagent coordination in the tests. The user can chose either 1) to use a single mechanism or 2) to test with every mechanism. In the case where every mechanism is tested, one combination will be generated for each mechanism available. The different coordination mechanisms are presented in Section 6.1.1.

- *Formation:* This is the ship formations to test, defining the relative position of each frigate in the fleet. The formations are defined in Appendix A.2. This parameter is used only when a coordination mechanism is tested. A combination will be generated for each formation to test.

- *Distance:* The distance between the ships in a coordination formation. Usually, it represents the distance to the centre (cargo ship) along one axis. Obviously, this parameter is used only when a coordination mechanism is tested.

- *Communication preparation delay:* It represents the time to correctly prepare a message with security measures and the correct headers. This is invariant and independent of the size of the messages. This parameter is used only when a coordination mechanism is tested.

- *Bandwidth:* This is the bandwidth of the communication channel, and it is fixed for the length of the simulation. Thus, the bandwidth can be reduced to represent background noise or degraded communication conditions. This parameter is used when any coordination mechanism is tested.

- *Communication waiting time:* This represents the time any agent has to deliberate and return an answer. When waiting for a reply, an agent will wait a specific time defined by: *Time to send the initial message* + *Communication waiting time* + *Estimated time to receive the reply*. Of course, this parameter is used only when a coordination mechanism is tested.

- *Frigate per threat:* This is the number of frigates that will engage each incoming threat. The values of this parameter range from one frigate per threat to every frigate for each threat. This parameter is used only when coordinating with the Contract Net protocol.

- *Allocation algorithm:* Some coordination mechanisms (*Central coordination* and *∼Brown coordination*) compute a matrix of success probability, which contains the evaluation of probability[7] to destroy each threat, for each frigate. When this matrix is obtained, two allocation algorithms can be used: a complete state lookup and a greedy algorithm.

- *Maximum ship weight deviation:* The ∼Brown coordination uses different ship weighting related to each ship's importance. Once obtained, the weights are normalised in such a way that the maximum weight is 1 and the minimum is (1−*Ship weight deviation*).

- *Fleet engagement priority evaluation:* In the *∼Brown* method, the priority of each threat according to the fleet is evaluated from the received probabilities of success (P_S) for each frigate. These fleet priorities will later be used in the evaluation of the individual priority. There are three different fleet engagement priority evaluations: the mean of P_S, the highest P_S and the multiplication of P_S.

- *Capability matrix evaluation:* In the *∼Brown* mechanism, a capability matrix is computed by each ship at a certain moment. Many different evaluation methods can be tested.

- *Backup:* This parameter represents whether or not ships will demand backup in case they cannot engage a threat with a sufficient probability of success. This is used only in the *Zone Defence* coordination mechanism.

- *Threshold:* Used in the *Zone Defence* coordination method, this represents the probability of success threshold under which a ship will seek assistance in the engagement of a threat.

- *Number of frigates:* Used only in the *Zone defence coordination* mechanism, this is the number of frigates in the scenarios to test. It is used to evaluate the effects of more or less defending ships on an AAW scenario. In the other coordination protocols, the defined formation is used with exactly four frigates.

- *Bayesian sector:* This is the Bayesian sector we wish to test. Further details about Bayesian sectors are available in Blodgett et al. [2002] and Plamondon et al. [2003]. For now, suffice to say that this restricts the random appearance of threats in a specific azimuthal range (based on the ship positioned in the centre of the simulation area). This is a special test and is used only to generate the results of the *Bayesian movement* approach (see Section 23).

[7] More details on these evaluation methods are found in 6.1.2

- *Output:* This is the file where the outputs are saved. Typically, they are saved in Excel format (.xls), though results can also be saved in comma separated values format (.csv).

4.3 Summary

In this chapter, we presented the Naval Defence Simulator (NDS), developed at DAMAS for the purpose of simulating a naval anti-air warfare environment. We have seen that in the case of modern AAW, as in project NEREUS, it is not practical to develop prototypes for new concepts on real ships, given the cost of modern weaponry and the number of people involved.

NDS is a stable, industrial grade simulated environment to develop and validate methods to increase the survivability of frigates in anti-air warfare scenarios. NDS permits a large quantity of tests, including some very complex scenarios. With low costs compared to real-life demonstrations, this allows us to develop, implement, validate and compare a broad range of concepts.

Chapter 5

Planning in Project NEREUS

Initially, project NEREUS was aimed at managing the resources on a single typical frigate. Thus, most research was directed at planning in real-time systems. Many approaches were investigated and some will be presented briefly in this chapter. More details can be found in Plamondon [2003] and Soucy [2003], which inspired some sections of this chapter. More work on the project, which is not presented here, can be found in Blodgett et al. [2001].

5.1 Agents in Project NEREUS

A first step in defining agents is to describe them in term of their Performance, Environment, Actuators, Sensors (PEAS). The PEAS (initially presented as Percepts, Actions, Goals, Environment (PAGE)), was introduced by Russell and Norvig [1995].

In order to describe our agents, we will define their PEAS, which is shown in Table 5.1. Note that the resources mentioned in Table 5.1 are those available on a typical frigate, and are described in more detail in Appendix A.

A second step in developing agents is to describe the environment in which they will evolve. Considering the attributes of environments presented in Section 2.1.1, we can describe the environment in project NEREUS. Since the agent only has a partial view (its radar range), we can deem the environment partially observable. It could be argued that the environment is in fact fully observable, since ASMs are always launched inside the radar's range, because they have smaller autonomy than the frigate's radar detection

	NEREUS Agent
Performance	The performance is evaluated by the survivability of the ship and the utilisation of resources. The survivability is the most important performance measure of these two.
Actions	Use of every onboard resource, including hardkill and softkill weapon systems, and the steering and propulsion systems.
Environment	The environment is an unpredictable, naval environment. It might be situated on the coastal or in deep-sea.
Sensors	Everything detected by radars and sonars, including friends and foes, airplanes, ships, missiles, chaff clouds, etc.

Table 5.1: PEAS of a NEREUS agent.

radius. However, the agent lacks important information pertaining to the choice of actions: it does not know *when* a detected airplane will launch its ASM. Furthermore, the NEREUS environment is clearly stochastic, as one can refer to Appendix A and see that weapons have a *kill probability*. Finally, the environment is also sequential, dynamic and continuous, since a simulation is a scenario that stretches on a continuous timeline, with the threat appearing at unpredictable times.

The third step is to integrate the developed agents in the environment. To know how the agents blend in the simulation in terms of agent management, please refer to Section 4.2.1.

That leaves us with the most important part of agency: the deliberation process. The decision-support system (DSS) agent, to achieve an efficient defence, will 1) propose a rotation movement, which will bring the threats in the most effective defence sectors, and 2) elaborate a defence plan. This plan is composed of actions, consisting of resources used against threats, which the frigate should execute to ensure its defence. The movement actions choice and the construction of the defence plan are not necessarily done sequentially; they could be interleaved or done within the same planning level.

5.2 Movement

The relative position of a frigate facing incoming threats has an influence on its defence potential, and consequently, it is important to investigate this aspect to determine a good positioning method. Determining a good position allows making an optimal use

of the resources, while ensuring threats are not in the blind zones of the ship.

Analysing the problem of a threat targeting a frigate, we can propose a general formula expressing the survival rate. The survival probability, being the complement of the destruction probability, is expressed by the following equation (the probabilities are in the range of [0, 1]).

$$P(survive_i) = 1 - P(kill_i|hit_i) \cdot P(hit_i|lock_i) \cdot P(lock_i) \tag{5.1}$$

For a scenario with n threats, the survival probability is expressed in the following way:

$$P(survive) = \prod_{i=1}^{n} P(survive_i) \tag{5.2}$$

Two approaches were developed in the pursuit of a positioning solution. The first one, the Bayesian approach, aims at reducing the probability that a threat hits the frigate. The second one, the Radar Cross Section Reduction (RCSR)[1] movement aims at lowering the probability that a threat gets a lock on the frigate. So, the Bayesian approach will try to increase the survivability by lowering $P(hit_i|lock_i)$ (by destroying said threat) and the RCSR approach will try to lower $P(lock_i)$ (by reducing the Radar Cross Section) in order to increase survivability. Even though it is not in the scope of our research, we could imagine a third way to reduce the probability of destruction, by finding a way to lower $P(kill_i|hit_i)$.

5.2.1 Increasing Resource Efficiency

The first step required to use this approach, is to determine the sectors surrounding the frigate and their relative efficiency. These sectors are defined and explained in Appendix A.1.5.

Concerning frigate positioning, a specific scenario is represented by a *threat-sector* combination, which contains the information pertaining to the location of the threats (i.e., which threat is in which sector). Starting from the initial scenario state, we want to generate all the threat-sector possible combinations made by any rotation in the range $[-180°, 180°]$. This can be done with the following method, taken from Morissette and Chaib-draa [2004].

Knowing the current distance, direction and speed of the threat, we can compute the maximum time available for rotation. Knowing this time and the frigate rotation speed,

[1] The Radar Cross Section is the apparent size of an object, caused by the extent to which an object reflects radar pulses.

we can find the maximum possible rotation. Then we find every sector in which the threat could be found by making rotations in the range [-*maximum rotation*, *maximum rotation*]. We thus obtain, for each threat, the set of sectors that can be reached within the time available for rotation. We simulate a frigate rotation, for each sector, which would bring the threat to the beginning of this sector[2]. For example, in Figure 5.1, to bring the threat to the beginning of sector 1, we simulate a frigate rotation of 15°, which will give the situation depicted in Figure 5.2. For each threat, there is a maximum of 12 simulated rotations, generating 12 threat-sector combinations, since there are 12 different sectors. Thus, for n threats, we have a maximum of $12n$ simulated rotations. This algorithm will generate every possible threat-sector combination, and is in $O(n)$. The proof can be found in Morissette et al. [2004].

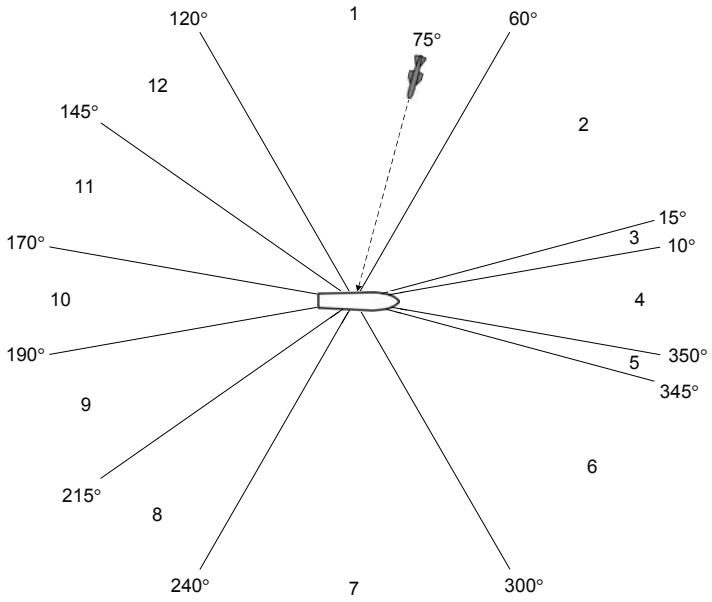

Figure 5.1: Initial situation.

When all the threat-sector combinations are found, we evaluate them according to one of the two implemented heuristics. To develop those heuristics, a learning module was set up, which evaluated the effectiveness in regards to two different metrics: the survival rate of the frigate and the percentage of threats destroyed. For the first metric, a mean of the survival rate was made for each sector, for 1 to 8 threats. For the second metric, what was evaluated was the potential number of threats destroyed for each sector, also for 1 to 8 threats.

Using the results of the learning module, the two following heuristics (the Bayesian

[2]The beginning of a sector is the angle we obtain when we enter the sector by making a counter-clockwise revolution.

Figure 5.2: After a 15° rotation.

and the Potential Number of Hits (PNH) approaches) were implemented, making it possible to evaluate the effectiveness of each position.

Bayesian Approach

The first heuristic is a *naive Bayes classifier* that calculates the survival probability of the frigate according to the sectors where the threats are located (Plamondon et al. [2003]). This method is known as *naive* because it is based on the simplifying assumption that the attributes' values are conditionally independent given the target value.

The best position will be the one that maximises Equation 5.3, which comes from Equation 5.2 where $P(survive_i) = P(survive_{S_i}|S_i)$. In Equation 5.3, S_i is the sector in which threat i is situated and $P(survive_{S_i})$ is the survival probability if a threat is incoming in sector S_i. Thus, every threat-sector combination generated earlier will then be evaluated using the values from the learning module to determine the $P(survive_{S_i}|S_i)$.

$$P(survive)_{NB} = \prod_{i=1}^{n} P(survive_{S_i}|S_i) \tag{5.3}$$

If we consider Equation 5.3 giving the survival probability according to the *naive Bayes* heuristic ($P(survive)_{NB}$), we find that the survival rate of the frigate is given by the

product of the survival rate for each incoming threat, which is dependent on the sector containing the threat. Intuitively, this means that the frigate will try to place threats where it has the most efficiency for survival, thereby optimising the use of available resources. If n threats are incoming in the same sector, this heuristic considers that the n^{th} threat has the same potential chance to be destroyed as the first or second. Obviously, while this might be close to truth for a few threats, the lack of available resources renders this invalid for more than two or three threats. In fact, even the best sector (sector 1-7) will be able to engage only a limited number of threats due to STIR availability (or unavailability in this case), which is the most constrained resource.

Potential Number of Hits

The second heuristic evaluates the Potential Number of Hits (PNH) on the frigate, and takes into account the number of threats found in each sector (Morissette et al. [2004]). The survival probability according to the PNH heuristic ($P(survive)_{PNH}$) is calculated as follows:

$$P(survive)_{PNH} = \sum_{i=1}^{12} n_i H(i|n_i) \tag{5.4}$$

where n_i represents the number of threats found in the sector i, and $H(i|n_i)$ the percentage of threats that would reach the frigate when n_i threats are in the sector i. The value of $H(i|n_i)$ is given by the learning module detailed earlier. Thus, the PNH represents an evaluation of the number of threats that will hit the frigate. Since this heuristic considers each sector and the number of threats it contains rather than each threat individually, the dependency between threats is better represented than with the naive Bayes classifier.

Figure 5.3 compares the efficiency of the two heuristics and a solution without any movement. These comparisons were made by simulating 15,000 different scenarios for each movement with from 1 to 8 threats, for a total of 360,000 tests.

We see that the PNH gives better results in every case, which is significant since the high number of tests run will result in a very low statistical error. Therefore, we can safely affirm that the reasoning about the dependency of threats was right. However, we should also add that the PNH is still just a heuristic and is imperfect. In this heuristic, we suppose that the sectors are independent, which is obviously not the case since our resource systems span more than one sector. A learning module still closer to reality would consider the number of threats in each sector, while also considering the dependencies with the threats in nearby sectors when learning the probabilities to destroy the threats.

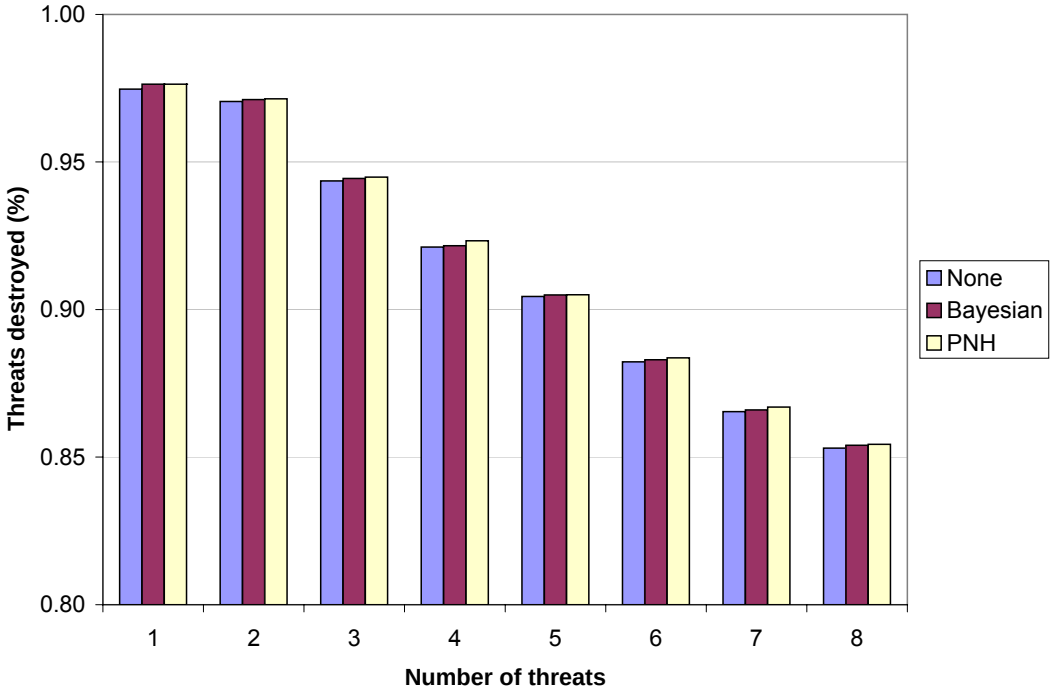

Figure 5.3: Efficiency of position-choosing heuristics (Morissette et al. [2004]).

5.2.2 Reducing Detection

As said earlier, a second way to increase the overall survivability is to decrease the probability that a threat gets a lock on the frigate, which is equivalent to reducing $P(lock_i)$ in Equation 5.1. The Radar Cross Section Reduction (RCSR) movement was specially developed with this intent.

Radar Cross Section Reduction Movement

The Radar Cross Section Reduction (RCSR) movement aims at reducing the frigate's radar cross-section exposed to incoming threats. This is important since the capability of threats to lock onto the frigate is directly related to the radar cross-section of the frigate they perceive (Liang and Liem [1992], Liang [1995]). Thus, an appropriate frigate position makes it harder for threats to lock and keep a lock on the frigate.

To implement an algorithm that minimises the surface exposed to all threats, we must consider 1) the angle at which the missile is incoming onto the frigate and 2) the estimated time available to turn the ship.

To evaluate the exposed surface, we used a simplified frigate structure, as illustrated in Figures 5.4 and 5.5, which shows a frigate compared to the simplified structure used.

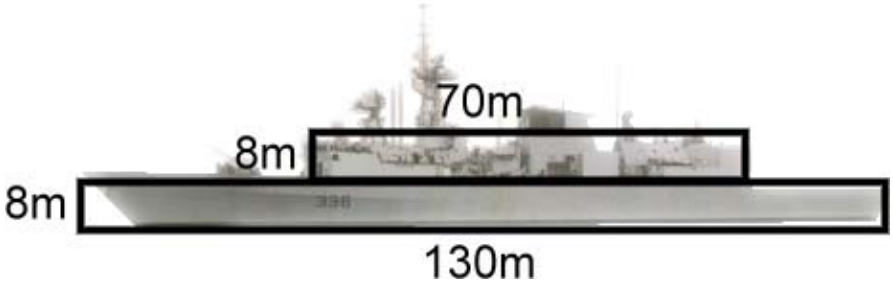

Figure 5.4: Frigate's side view.

Figure 5.5: Frigate's front/rear view.

When the frigate is on the side, the exposed surface is maximised, and is approximately 1,600 m^2. This is to the easiest case for the threat to lock onto the frigate. On the other side, if the threat is oriented toward the front (or rear), the exposed surface is minimal at approximately 240 m^2, which is only 15 % of the prior 1,600 m^2.

The idea is to minimise the total surface exposed to threats. Sadly, project NEREUS is not yet in a phase where uncertainty concerning the locking phase is introduced, so no results are yet available about this particular movement. However, the current implemented algorithm hints that this method will greatly help increase the overall survivability of the frigates.

5.2.3 Combining Both Movements

Bayesian movement aims at diminishing $P(hit_i|lock_i)$, so that it is less likely that a threat reaches the frigate, while RCSR movement aims at reducing $P(lock_i)$, so that

it is more likely that a threat never gets a clear lock on the frigate. Unfortunately, the current layout of the systems on the frigate makes the two movements select contradicting positions. In fact, the best sector for the Bayesian movement (sector 1-7), which minimises $P(hit_i|lock_i)$ happens to be the worst sector from the standpoint of the RCSR movement, since $P(lock_i)$ is maximised. We can perceive this problem in Figure 5.6 and 5.7. We see in these figures that the two positioning algorithms suggest two completely different positions for the same initial situation.

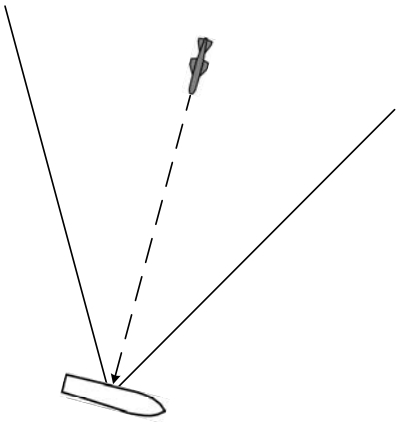

Figure 5.6: Optimal Bayesian positioning.

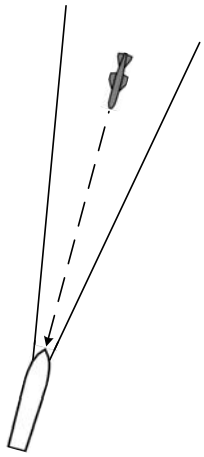

Figure 5.7: Optimal RCSR positioning.

However, these two phases are usually differentiated in time, as the RCSR movement is generally made prior to the Bayesian movement, as shown on figure 5.8. It is a simple problem to effectively use these two movements if we can restrain them to their respective coverage area. However, to know if a threat is indeed locked on a frigate, an agent has to reason using uncertainties. But once a threat is locked, it will almost certainly remain locked on the frigate until it hits or is destroyed, which renders the RCSR movement useless.

Thus, we suggest a metalevel agent that evaluates the Bayesian and RCSR movements and suggests which hybrid movement is the best, given a particular situation. This metalevel agent could use rules for prioritising the use of specific weapon systems (hardkill and softkill) or high-level strategies (screening, displacement to help the softkill systems). Still, at any single time, this can be reduced to minimising Equations 5.1 (and 5.2 in the case of multi-threat scenarios). However, the problem is that we want to minimise this equation over the whole time interval between the detection of the first threat, to the disappearance of the last one. Obviously, this is not a simple problem in itself.

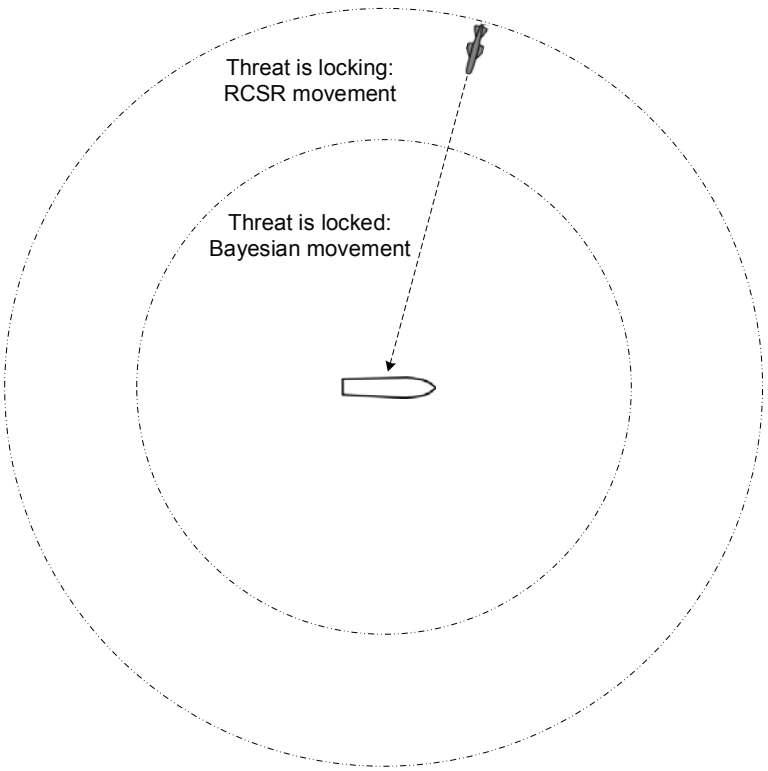

Figure 5.8: Repartition of Bayesian and RCSR movements in time.

5.3 Tabu Search for Resource Management

The work presented in this section was originally done and discussed by Soucy [2003].

Tabu is a word in *Tongan* (From Tonga Islands) that signifies *a prohibition, excluding something from use, approach, or mention because of its sacred and inviolable nature*. The Tabu search algorithm thus highlights alternatives to be avoided. In a more significant way, Tabu search introduces two fundamental principles: an adaptive memory and a judicious exploration. The adaptive storage capacity appears by the establishment of a search procedure based on a restricted solution space. The emphasis on a judicious exploration makes it possible to sanction certain changes to avoid the re-evaluation of already constructed solutions.

Two significant components of the Tabu search are the strategies of intensification and diversification. The intensification stands on the possibility to modify the decision rules to encourage the changes or solutions that proved to be advantageous in the past. The especially good solutions, said *elite solutions*, will be preserved so that their immediate neighbours can be examined. Elite solutions are often determined

by a threshold, quantified on certain attributes, which will make it possible to directly identify these solutions at search time. On the other hand, diversification will encourage the search progression toward still unexplored areas, which will make it possible to produce solutions significantly differing from already visited solutions.

Tabu search principle can be described as follows:

1. Choose a possible change to improve the solution.
2. Eliminate possible loops[3] by omitting (or penalising) certain changes or solutions, which are thereafter tagged *tabu*.

At first, Tabu search will create an initial solution, which will start as being the current best solution. This initial solution can be chosen randomly or constructed deliberately. However, its choice will directly influence the final solution and its quality; the worse the initial solution, the longer it takes to attain a good solution. Afterward, any change carried out on the solution will be marked as *active* so that it is penalised to avoid making it again during a definite iteration count.

Then, the algorithm seeks to obtain a new solution, which is contained in the set of answers in the vicinity of the current solution. This neighbouring set is composed of solutions that are obtained by applying a change not marked as *active* to the current best solution. If this new solution is better than the current solution, it is adopted as being the new best found solution.

This process is iterated until stopped by a static stop condition (reached a certain quality for example), or by a metalevel process that considers the real-time deadlines (See Section 3.1.1 for more on deadlines).

More details and the complete Tabu search algorithm can be found in Soucy [2003].

5.3.1 Tabu Search in Project NEREUS

In a real-time system such as NEREUS, anytime approaches, like the Tabu search, are not only indicated, but also required.

[3] By loop we mean the fact of going back to an already visited solution after carrying out various operations.

The first step to implement this approach is to decide how the initial solution will be constructed. A solution is a plan, represented as a tree, where each node is an action[4].

The initial solution is created in two steps. At first, a list of all possible actions is generated with the *GenCue* algorithm, adapted from Chalmers and da Ponte [1995] in Paquet [2001b]. The second step is to create a plan selecting actions and adding contingencies such as the destruction of an incoming threat or the failure of a resource to destroy said threat. Usually, a *resource engagement*[5] action is followed by an active *perception* action. By that, we mean that the agent will check on the environment to see the value of a proposition over the environment. In our case, the proposition would be "*is this threat destroyed?*" and corresponds to a Kill Assessment (KA)[6]. In the algorithm, a resource engagement and KA pair is considered as one unit (an engagement) when considering actions to change; they are added or removed together. For more details on how this solution is constructed, refer to Plamondon [2003] and Soucy [2003].

Following the formalism presented in Figure 3.4, the Figure 5.9 represents an initial plan to engage an incoming ASM. The quality of this solution, which contains only a SAM engagement, is 87.5 %. We will now see how the Tabu search is used to increase the quality of this solution.

Once an initial solution is obtained, the Tabu search iterative process is started and the initial solution is modified by removing and adding engagements in order to improve the quality of the plan. These engagements are taken from the list generated initially by the *GenCue* algorithm. Obviously, removing engagements will degrade the solution. However, this will free up limited resources such as STIRs and will allow the addition of new engagements. Adding a new action to the plan is subject to the same constraints as in the creation of the initial solution. When an action is added or deleted, it is set to *active* for a number of iterations, predetermined at design time, for which it cannot be used in changes. This means an engagement currently in the plan cannot be removed, while a recently deleted engagement cannot be put back until a certain number of iterations have passed. This flag is the actual enforcement of the intensification and diversification policies mentioned earlier; the algorithm will avoid the solutions evaluated in the past by disadvantaging the changes already carried out on the solution. Since this algorithm does not preserve the solutions already evaluated, it is called *memory less*. During execution, the algorithm will oscillate between saturated

[4] See section 3.2.2 for more details on plan representation.

[5] A resource engagement is the use of a specific resource against a threat, at a determined time.

[6] A KA action is an evaluation of the environment to verify whether or not a threat has been destroyed.

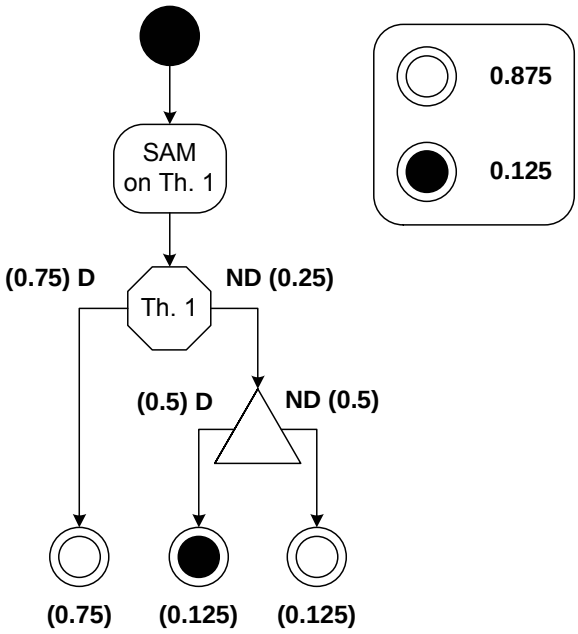

Figure 5.9: Example of an initial plan.

plans[7], withdrawing engagements only when it is not impossible to add new ones.

Once a new plan is constructed, it needs to be evaluated and compared to the initial solution. Going back to the simple example presented earlier, Figure 5.10 shows a new constructed plan with a CIWS action added. Evaluating the survival probability if the newly constructed plan is carried out, we see that the quality of this plan is 90 %, which is an absolute improvement of 2.5 % compared to the initial plan (Figure 5.9).

5.3.2 Results

Table 5.2 and Figure 5.11 (Morissette and Chaib-draa [2004]) show the *absolute* improvement on the quality of the initial solution, depending on the initial number of threats. As shown in Figure 5.11, the improvement in the quality of the solution becomes very small, almost nil, around 50 iterations.

An interesting result is that, as shown in Figures 5.12, 5.13 and 5.14 (from Soucy [2003]), the plan generated by Tabu search will tend to use more gun engagements and less SAMs against threats. This can be explained by the fact that since the gun is used at a shorter range and is faster than SAMs, a gun engagement requires less time from

[7] A saturated plan is a plan where no more engagements can be added.

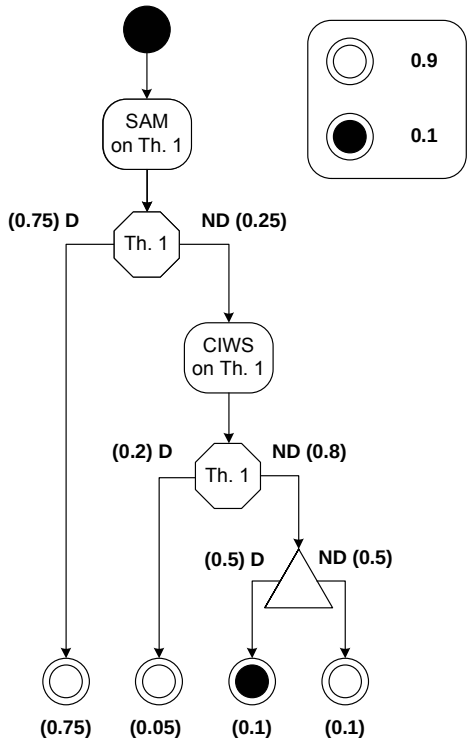

Figure 5.10: Example of a plan after a Tabu search iteration.

Number of Threats	*Absolute Improvement (%)*			
	1 Iteration	10 Iterations	25 Iterations	50 Iterations
1	1.215	1.215	1.215	1.215
2	0.859	1.336	1.379	1.379
3	1.128	1.739	1.798	1.842
4	1.289	2.101	2.182	2.218
5	1.671	2.760	2.862	2.905
6	1.888	3.146	3.263	3.335
7	1.964	3.416	3.529	3.592
8	1.936	3.565	3.694	3.741

Table 5.2: Improvement of initial plan, depending on the number of iterations (Morissette and Chaib-draa [2004]).

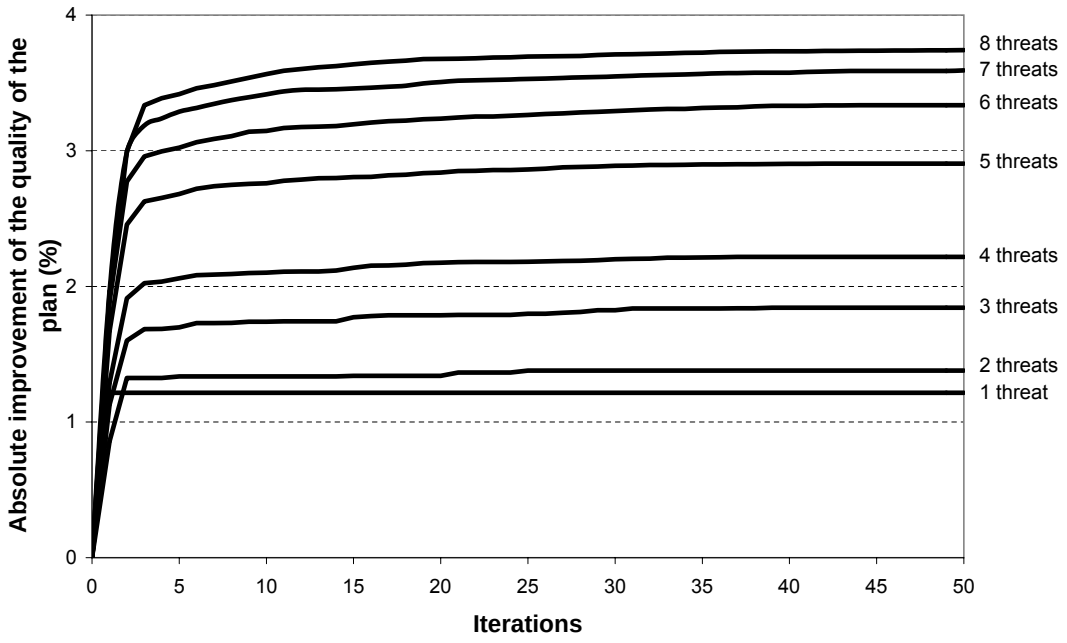

Figure 5.11: Improvement of initial plan, depending on the number of iterations (Morissette and Chaib-draa [2004]).

the STIRs, which can then be used to add other engagements. On the other hand, the fact we see no sensible difference for the CIWS use is coherent with this explanation; freeing the CIWS does not allow planning more actions.

Finally, Figure 5.15 shows comparative results between the *reactive* algorithm (which corresponds roughly to human response), *deliberative* algorithm (the algorithm constructing the initial solution) and the deliberative algorithm with *Tabu* improvement.

5.4 Discussion

We have seen that positioning the frigate could certainly improve the survivability of the ship. We have also demonstrated that there is more than one way to improve the survivability in the system, and that each type of movement aims at optimising one of the elements of the equation. However, we also pointed out that there is still work to be done in this domain.

Furthermore, we have seen how some anytime algorithms can be used in the context of project NEREUS to supplement the Resource Management (RM) problem in a typical

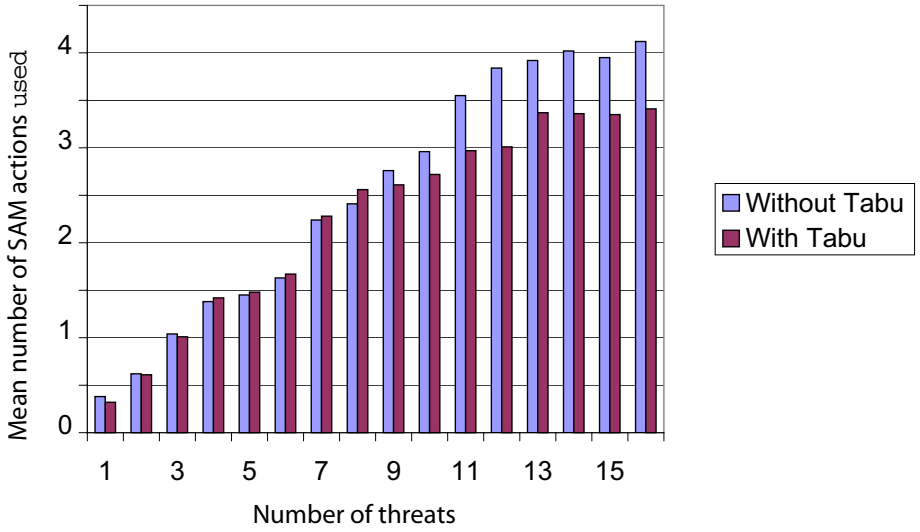

Figure 5.12: SAMs utilisation (Soucy [2003]).

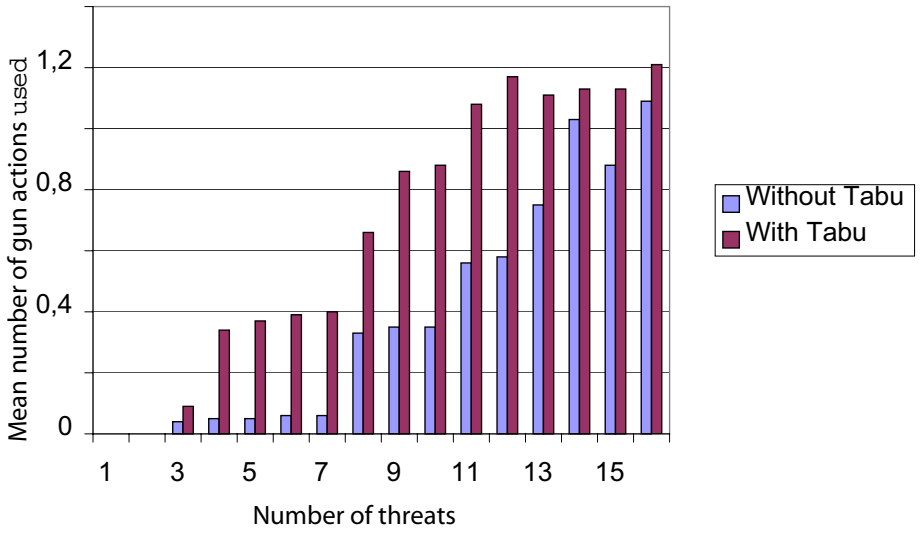

Figure 5.13: Gun utilisation (Soucy [2003]).

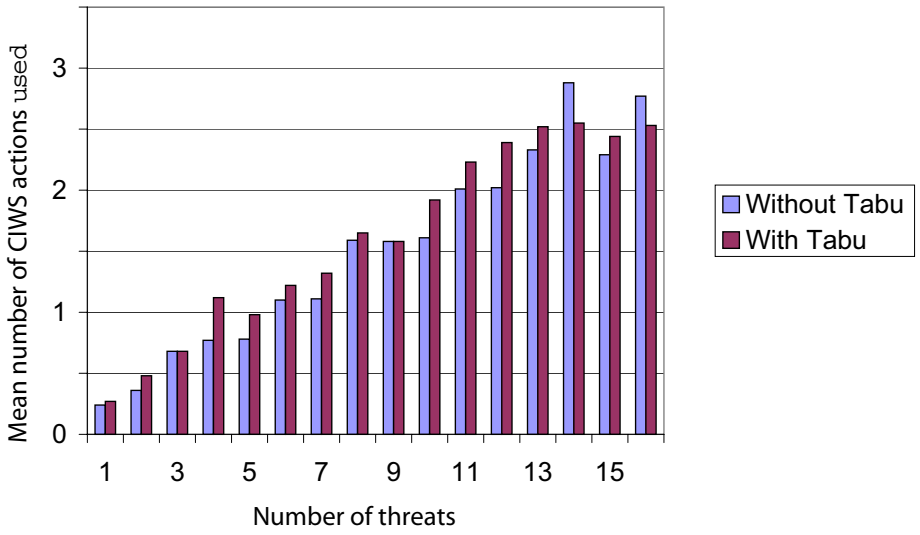

Figure 5.14: CIWS utilisation (Soucy [2003]).

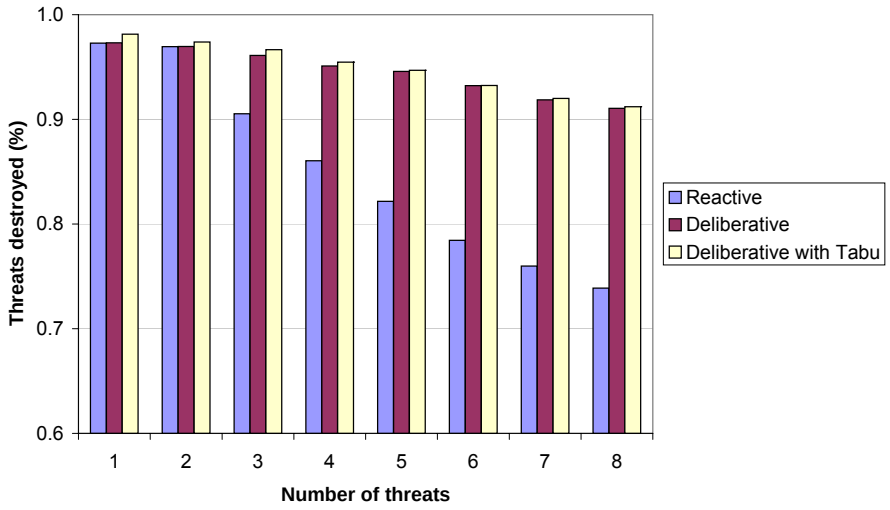

Figure 5.15: Comparing three planning algorithms (Morissette and Chaib-draa [2004]).

frigate as a decision-support system (DSS). Of course, Tabu search is not the only anytime algorithm that could be implemented in such a system.

Not much has been done yet on the integration of movements and planning. When should we integrate movement in the deliberation process? Before or after the construction of the plan? *During* the planning phase? Indeed, there is research still to be done to find a good way to integrate movement in a unified deliberation process. However, our intuition is that movement could be added as a basic action in an algorithm such as Tabu search, augmenting the number of possible cues available for switching. Two facts suggest that this could be a good way to go:

1. The increase in available options for the Tabu search algorithm could allow better quality solutions, as more available options also increases the chances to find a good solution.

2. The Tabu algorithm approaches a near optimal solution after about 50 iterations, taking only 39.24 milliseconds to do so. Furthermore, the time taken by the algorithm increases linearly with the number of threats involved in the planning. Considering that a threat can appear in ranges from 5 km to 80 km (by the specifications of our project), this leaves from 5.8 to 94.1 seconds before the first impact. Usually, this is sufficient for the Tabu search to obtain a very good solution. Bearing this in mind, we could certainly increase the number of engagements with which the Tabu search has to contend, even though it would slightly reduce the speed with which the algorithm converges to a good solution. In all cases, we would still get a solution of equivalent or better quality, since there *is* enough time to plan.

Chapter 6

Multi-Platform Coordination in Project NEREUS

As we have seen in Section 2.2.2, coordination is the process by which agents avoid superfluous actions, by managing the interdependencies to minimise the conflicting actions and goals. In most systems, this coordination must be carried out in an environment constrained on time, available bandwidth, etc.

In the case of the defence of several frigates, multiagent coordination is a very complex problem of capital importance. The environment imposes strong real-time constraints, which is mostly due to the threats becoming smarter and the AAW situations happening more and more on the littoral rather than in open sea. In a standard AAW situation, operators have few seconds available, in which they must identify threats, choose and apply defence plans. When in a multi-frigate system, it is also necessary to coordinate defence actions between the ships. As the reaction time is usually very short, it is often not possible for the operators to coordinate their actions with the other members of the group. This can result in 1) redundancy in the engagements, using more resources than necessary, 2) inefficient defence and 3) an increase in the cost of the global defence solution. Another impact of the lack of coordination is the negative interactions that can take place when certain resources are used in parallel, which creates a degradation of the global solution. This prompts for an increasing need for cooperation between frigates. Indeed, good coordination mechanisms for the optimal use of the resources of a group of frigates become essential during a military deployment.

Thus, in NEREUS, the coordination is cooperative and is intended to organise the individual actions toward a common goal, which is the efficient defence of the complete

fleet. This problem is very complex, since we focus on a problem close to real-life situations, where we have strict deadlines (a limited amount of time to coordinate) and communications are not free.

6.1 Coordination Mechanisms

Before going on with the coordination mechanisms, which are processes and methods by which agents coordinate their actions, it would be useful to describe the MAS environment, as seen in Section 2.2.

Table 6.1 reproduces the Table 2.1 (which specifies the attributes of MAS) with our specific values for the multiagent system environment in project NEREUS.

	Attribute	*Range*
Agents	Number	Usually 4
	Uniformity	Homogeneous
	Goals	Complementary
	Architecture	Mostly deliberative
	Abilities	Somewhat advanced
Interactions	Frequency	Low to Medium
	Persistence	Middle-term
	Level	Small but meaningful
	Pattern	Decentralised or Hierarchical
	Variability	Changeable
	Purpose	Cooperative
Environment	Predictability	Stochastic
	Accessibility	Slightly limited
	Dynamics	Fixed for a scenario
	Diversity	Limited
	Resource availability	Restricted

Table 6.1: Characteristics of a multiagent system.

Agents: In the multi-frigate context of our simulator, each frigate is considered as a sophisticated, "autonomous" agent. In fact, these agents have a limited autonomy because 1) they are members of a team (i.e., fleet), 2) they are expected to be fully cooperative and 3) they have to respect military doctrines and rules of engagement. Thus, all agents need to coordinate themselves to achieve an acceptable

solution. At first, it is important to note that we are only describing the agents for a frigate decision-support system (DSS). The simulator also contains basic agents for airplanes and ASMs. However, since they are not studied in project NEREUS, we will make abstractions of these simple agents while describing the system. The number of ships in a typical task group (or fleet) is 4 frigates and an important unit (typically a cargo vessel in our case). The frigates are responsible for their own defence and the defence of the cargo vessel, which has no defence systems. However, we also tested for situations with more or less frigates, to see the impact on the survivability of the fleet[1]. Considering only the frigate agents, they are clearly uniform. However, we plan to include diversity in the available resources in the future of the project. This would allow simulating ships from different classes and nationalities. The goal of each frigate is to maximise the survivability of the fleet as a whole. To simulate this attitude, we played with the relative ship importance; when there is a ship of greater importance, the frigate will try to defend it, even to the detriment of its own survival. On a final note, the abilities of the agents are diversified and relatively advanced; they can use all onboard systems to create a great number of different solutions.

Interactions: The interactions in our simulator (NDS) happen in the form of asynchronous messages. Depending of the mechanisms used, there can be no message exchanged (such as in *Zone defence* with no need for backup) up to a number of messages on the order of the number of threats (as in Contract Net)[2]. The effects of these messages persist some moments since they are used in the plan creation. However, it might also happen that they become useless because replanning is required. The messages transmitted are reduced to the strict minimum, but they require some knowledge to be constructed before passing and analysed when received. We implemented both centralised and decentralised mechanisms. Whether a mechanism is centralised or not has an important impact on agent interactions. The types of interactions are variable and will be further explained in Section 6.1.1. Obviously, the agents we develop are cooperative, as they all are from the same fleet and nationality. It is not yet in the scope of this project to develop mechanisms for ships with different nationalities and not completely cooperative.

Environment: We have said earlier that our environment is stochastic[3]. In our case, this means that the environment is partially foreseeable, as we can determine the probability that specific events occur. Furthermore, since we focus on Resource Management (RM), we do not consider the Situation and Threat Assessment

[1] A fleet is comprised of frigates (defending units) and cargo vessels. Furthermore, a "ship" can refer to either a frigate or a cargo vessel.

[2] These mechanisms are presented in Sections 6.1.3 to 6.1.6

[3] See Section 5.1

(STA) for a fleet and the inter-agent coordination required to obtain a global view of the system. Thus, we consider that every agent have the same *combined* view. However, the environment is still partially globally observable, meaning that even when the different sensory inputs from agents are put together, the view is not necessarily completely observable. In our simulator, the dynamics (such as the communication environment, etc.) are fixed for any specific situation, but can be changed over the course of many scenarios. In the NDS environment, the diversity is intentionally limited, to focus on specific important aspects. Finally, the resources available to any agent are limited by the number and specification of weapon systems. Usually, the most constraining aspect in our simulations is the available time, as we rarely run out of any physical resource.

With this being known, every single agent has to choose *when* to coordinate and *what* to do. This particular problem is complex, as has been expressed by many authors (Durfee and Lesser [1991], Jennings [1996], Cohen et al. [1997], Tambe [1997], Lizotte [1996]).

In NEREUS, our principal concern is not only the quality of the solution obtained by coordination but also the required time to obtain this final solution and the total use of communications. In fact, these two aspects are directly connected as the use of many communications combined with low bandwidth output can cause the creation of a solution to be significantly delayed.

6.1.1 General Coordination Mechanism

In the light of those elements, the first step to successfully coordinate agents consists of developing efficient coordination mechanisms[4]. A first approach can be considered: the *complete plan coordination* method, which consists of:

1. Detecting, identifying and prioritising threats.
2. Creating individual plans.
    - Creating normal plans.
    - Creating backup plans.

[4]Note that in the approaches we present, even though the steps seem to be sequential, there can be a certain amount of concurrency between them. For example, carrying out step 4 may introduce new interactions that would need to be dealt with (step 3).

3. Managing conflicts and interactions.
    - Ruling out conflicts between actions.
    - Accentuating positive interactions.
    - Avoiding negative interactions.
4. Effectuating the determined plan.

However, this coordination method has a problem: it takes much time to rule out the conflicts emerging from the individual planning, since many interactions are required to do so.

To resolve this problem, we consider a second approach: distributing the task among agents before starting the planning process. Thus, in the *threat distribution coordination* method, each threat will be allocated to agents, then each agent will only plan on threats it has been allocated. This task repartition problem is usually referred to as *task sharing* in the literature and is a divide-and-conquer approach to multiagent coordination[5]. Task sharing is usually composed of four different steps (Huhns and Stephens [1999]):

1. Task decomposition.
2. Task allocation.
3. Task accomplishment.
4. Result synthesis.

Taking these four steps of task sharing, it is clear that our major problem is the distribution of tasks among the agents. Indeed, the *decomposition* step is obvious: a task is the destruction of one threat. Furthermore, the *task accomplishment* has already been worked on, as seen in Chapter 5. The *results synthesis* is trivial; every agent can easily determine whether any threat is destroyed or not.

Therefore, using the idea of task sharing, we define the *threat distribution coordination* method in the following way:

[5]In might be interesting to note that the *task sharing* comes from the Cooperative Distributed Problem Solving (CDPS) (Smith and Davis [1980]), which contains three steps: 1) problem decomposition, 2) sub-problem solution and 3) solution synthesis. This is directly related to the fact that the MAS domain derives from Distributed Artificial Intelligence (DAI).

1. Detecting, identifying and prioritising threats.
2. Distributing threat among frigates.
    - Using *Zone Defence coordination*.
    - Using *Centralised coordination*.
    - Using *Contract Net coordination*.
    - Using *∼Brown coordination*.
    - Using *SWAT coordination*[6].
3. Creating individual plans.
    - Creating normal plans.
    - Creating backup plans.
4. Managing interactions.
    - Accentuating positive interactions.
    - Avoiding negative interactions.
5. Effectuating the determined plan.

The *threat distribution* alleviates the burden of the agents, as it significantly reduces the complexity of the planning process. Indeed, it is easier to plan for a few threats than for the complete list of threats. Another advantage of this approach is that we eliminate conflicting actions where two agents engage the same threat while another is left unimpeded. Furthermore, since we are not in a system with free communications, it takes several milliseconds for a message to reach its recipient. Thus, reducing the conflicts greatly accelerates the coordination process since much less messages will be exchanged.

The main difference between the *complete plan coordination* method and *threat distribution coordination* is mostly *when* this planning is done and upon *what* the planning is done. In the *complete plan coordination* method, a complete plan is generated considering all threats, while only a plan for the assigned threats is constructed in the *threat distribution coordination*.

In the case of the *complete plan coordination*, the coordination happens at Step 3, *after* the creation of individual plans. In the *threat distribution coordination*, coordination occurs at Step 2, *before* the creation of individual plans.

[6]This coordination mechanisms is proposed as an idea for future researches in Section 7.1.

Inspired by mechanisms commonly used to distribute tasks among agents (market mechanisms, Contract Net, multiagent planning and organisational structure), we developed coordination mechanisms for the *threat distribution* among frigates, which we will present in the following sections.

- *Zone defence:* Further described in Section 6.1.3, this method functions with an organisational structure (As seen in Section 2.2.2). The physical space around the fleet is separated into subspaces, each one being supervised and defended by one frigate. Upon one threat entering a *zone*, the ship in charge of this zone evaluates whether it can engage it, and asks for support from other ships if it cannot.
- *Central coordination:* Section 6.1.4 presents the *central coordination*, which is related to market mechanisms. In this method, every agent computes its probability of success (P_S)[7] against each threat and communicates them to a central coordinator. Then, the coordinator decides which agent will engage which threat. In this method, the coordinator is chosen beforehand and is usually the best-defended ship.
- *Contract Net:* Described in Section 6.1.5, this method has been presented first by Smith [1980] and is somewhat similar to the *Central Coordinator* method. In Contract Net, a coordinator does the following for each threat: 1) starts an auction to which each agent replies with its P_S for this threat, 2) chooses the agent that will engage this threat, and 3) communicates the allocation to the "winning" agent.
- *∼Brown:* Presented in Section 6.1.6, this method is adapted from Brown and Lane [2000] and Brown et al. [2001]. The idea is to compute the relative danger of each threat for the fleet. This is done while considering the relative importance of each ship, the confidence that a threat is targeting a ship and the P_S list (LP_S) of each ship. The threats are then allocated with an optimisation algorithm (such as a greedy algorithm).

6.1.2 Probability of Success

Before going on, we need to define what the probability of success (P_S) is and how it can be computed. The P_S of a frigate for a threat is a value that represents the probability, evaluated by the frigate, to destroy this threat. A P_S list (LP_S) for a specific frigate

[7] P_S are detailed in Section 6.1.2.

contains its P_S evaluation for each threat. Finally, a P_S matrix (MP_S) contains the LP_S of every frigate.

Obviously, it is possible to compute the exact P_S of a frigate against a threat only when the complete plan has been constructed. Since we need to determine the P_S *before* planning, it is imperative that we use heuristics to determine probabilities of success.

The current P_S evaluation implemented was inspired by Brown et al. [2001] and is based on the closest point of approach (CPA) of a threat relative to a frigate. The CPA, illustrated in Figure 6.1, is the shortest possible distance between the trajectory described by a threat and the evaluation point. In this example, the ships are stationary.

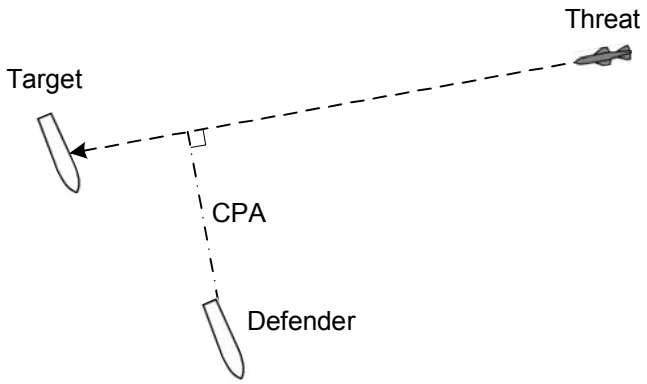

Figure 6.1: closest point of approach.

The CPA-based method to evaluate the P_S is given by the following formula:

$$P_S = P_K \cdot \left(1 - \frac{CPA}{R_{max}}\right) \tag{6.1}$$

where P_K is the *probability of kill* for this specific threat and R_{max} is the *maximum distance* at which a threat can be engaged (in our case, this is the range of the SAM, i.e. 20 km). The probability of kill P_K is relative to the prior number of threat engagements contracted. To determine the P_K, we did empiric testing and compiled the results. Then, we extrapolated the curve shown in Figure 6.2, which shows the estimated value of P_K, in function of the number of threats already targeting the frigate. We do not believe that this curve reflects the actual survivability as a function of the number of threats. Thus, Equation 6.2 does not reflect correctly the reality, as there is probably another inflexion point on the curve, with an asymptote at $y = 0$. However, the curve is accurate for any number of threats below eight, since the results and the expected value do not diverge by more than 0.75 % in any case.

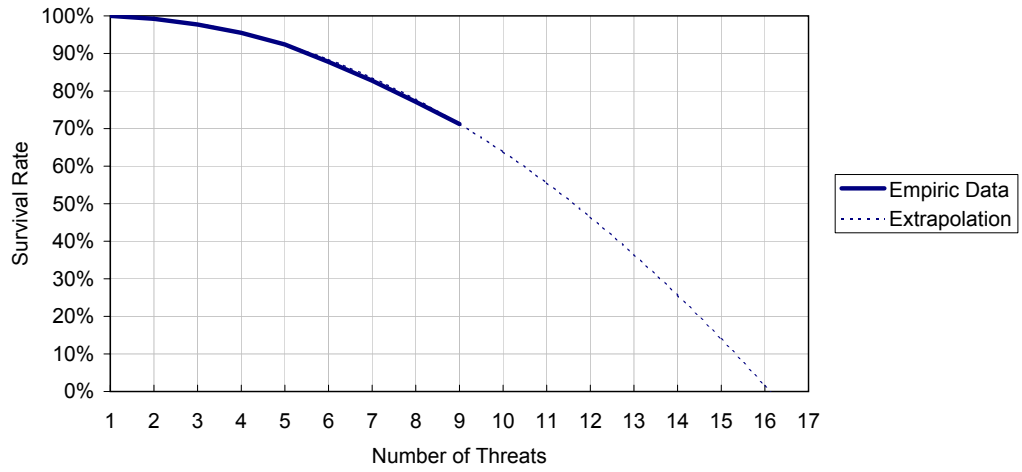

Figure 6.2: Extrapolation of P_K values.

The exact curve equation is:

$$P_K = (-0.00424414 \cdot nb\_threats^2) - (0.0020234375 \cdot nb\_threats) + 1 \qquad (6.2)$$

6.1.3 Zone Defence

The first coordination mechanism we look into is the *zone defence* coordination. The zone defence is a type of *organisational structure* coordination. In organisational structures, the notion of role is a key concept. Each role has associated responsibilities and preferences, which can be defined at design time or dynamically. The roles can be defined upon many heuristics, physical proximity being one of them. The principle of zone defence is that every agent is responsible for defending its "territory", as long as it has a good enough probability of doing so.

The zone defence mechanism defines an agent role as the defence of a particular azimuthal sector around the fleet. These sectors are determined in the following way:

1. The *centre* of the fleet is determined. In our case, it is the protected unit (a cargo ship) that is situated in the centre of the fleet.
2. Determine the azimuth of every ship from the centre unit.
3. Determine the boundaries, which divide exactly in two each pair of neighbouring ship azimuths.

This zone division is represented in Figure 6.3, which shows a simple formation for five frigates and a cargo vessel. When the boundaries are determined, each agent knows that it is responsible for engaging any threat detected in the zone delimited by its clockwise and counterclockwise boundaries. These boundaries are determined and maintained dynamically. Thus, if a ship is destroyed, they are resolved again to determine the new zones of defence. Figure 6.3 illustrates an initial situation where the frigates are uniformly positioned around the cargo ship in the centre. Figure 6.4 represents the sector redivision if frigate F_D is destroyed.

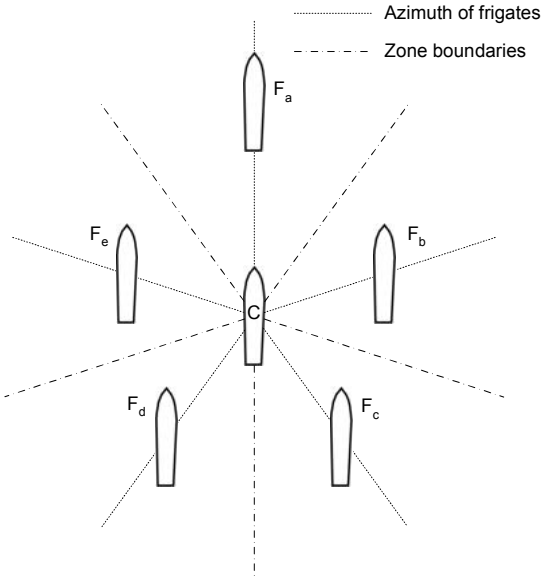

Figure 6.3: Initial zone division.

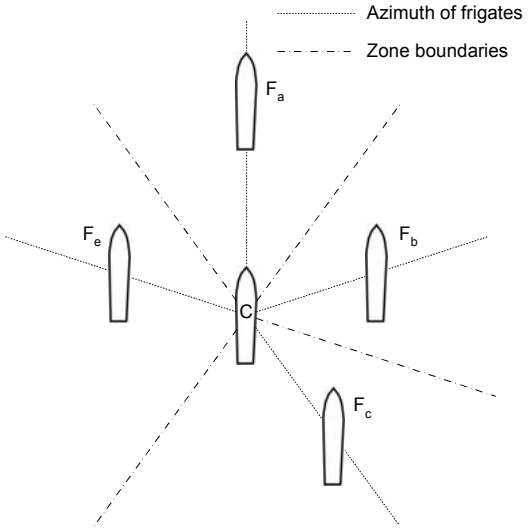

Figure 6.4: Dynamic zone redivision.

Another important concept of the *zone defence* is the defence threshold. If an agent, estimating a P_S, realises that this P_S is under the threshold, it will seek assistance in the engagement of this particular threat. At first, it demands help from the neighbour closest to the threat. Then, if the neighbour does not reply or refuses to engage the threat, the agent will seek assistance with its other neighbour. In the case of another refusal, it will add the threat to its plan, even though its estimated P_S value for this threat is below the threshold. Since we developed completely cooperative agents, an agent will refuse helping its neighbour if and only if its personal P_S evaluation for this threat is worse than the P_S evaluation of the asking agent.

Figure 6.5 presents the communication protocol for the *zone defence* mechanism. As with every other protocol presented in this chapter, we will use the formalism of Agent UML introduced by Odell et al. [2001].

This coordination mechanism has an advantage over the other mechanisms we will

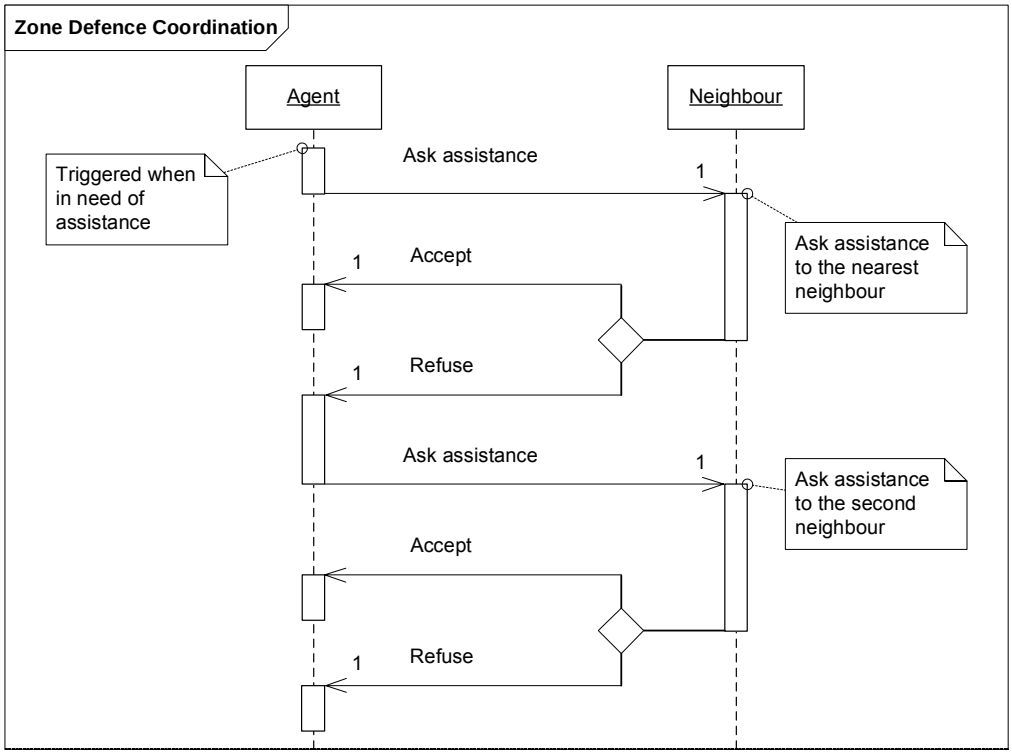

Figure 6.5: Zone defence protocol.

present: it uses almost no communications. Therefore, in the case where the communication channels have a very low output or are completely disabled, this mechanism stands out from the others.

6.1.4 Central Coordination

The *central coordination* mechanism is based on communications (according to the taxonomy presented in Figure 2.2), with a centralised coordinator. The concept of the central coordination is that a central agent is responsible to collect the information, and decide on a task distribution according to this information. In this case, the information transmitted is the LP_S of every agent.

The central coordination process is described in the following way:

1. The fleet chooses a coordinator.
2. When one or more threats are detected, every ship computes its LP_S and sends it to the coordinator.

3. The central coordinator constructs a capability matrix (MP_S), which is a matrix of LP_S for each frigate.
4. The coordinator decides how to assign the incoming threats to frigates, using the capability matrix and an optimisation algorithm.
5. The coordinator sends notification messages to chosen ships.

Note that the choice of an appropriate coordinator is done prior to detecting any threat. In our case, the coordinator is the frigate with the highest ranking, which is a simple value of the relative importance of each ship. The process of choosing the coordinator is facilitated by the assumption that every agent has the same common view, as we explained in Section 6.1. This, combined with the fact that our agents are fully co-operative, makes it possible for any agent to determine which one is the coordinator without the need for communication or negotiation.

Optionally, one or more backup coordinators can be chosen in the first step. These backup coordinators will receive the same information as the central coordinator, and will take the role of coordinator if the central coordinator is destroyed or becomes unable to accomplish its tasks.

The task assignation by the coordinator can be done with any optimisation algorithm, such as a greedy algorithm or even a complete lookup of the solution set[8]. The greedy algorithm is a fast heuristic, while the complete solution lookup takes more time, since it looks over every possible solution to choose the best one. Usually, the central coordinator allocates only one threat per frigate. However, if there are more threats than frigates, the coordinator has two choices. The first is to allocate one threat per frigate as usual and then start the process over by demanding another P_S evaluation for the remaining threats. The second choice is to allocate more than one threat per frigate, using a heuristic.

A disadvantage of the central coordination mechanism is that it is centralised. This is dangerous, since a single point of failure can make the whole coordination process abort. Indeed, if the coordinator becomes unresponsive for any reason (damaged communication system, the ship is sunk, etc.), the coordination process must be started over with a new coordinator, at the expense of several important seconds. Of course, the use of backup coordinators can alleviate this problem, but will also significantly increase the use of communication channels, which can become overloaded.

Figure 6.6 presents the *central coordination* mechanism protocol formalised with

[8]These allocation algorithms will be discussed further in Section 6.2.7.

AUML. In this figure, we see the messages exchanged between the coordinator and the agents (participants). We also see the iterative process (Assign Threats) which we described earlier. In this process, the coordinator asks the participants to send their LP_S. When it has received every P_S list, it then assigns the threats to the agents. The process is iterated as long as there are threats left unassigned.

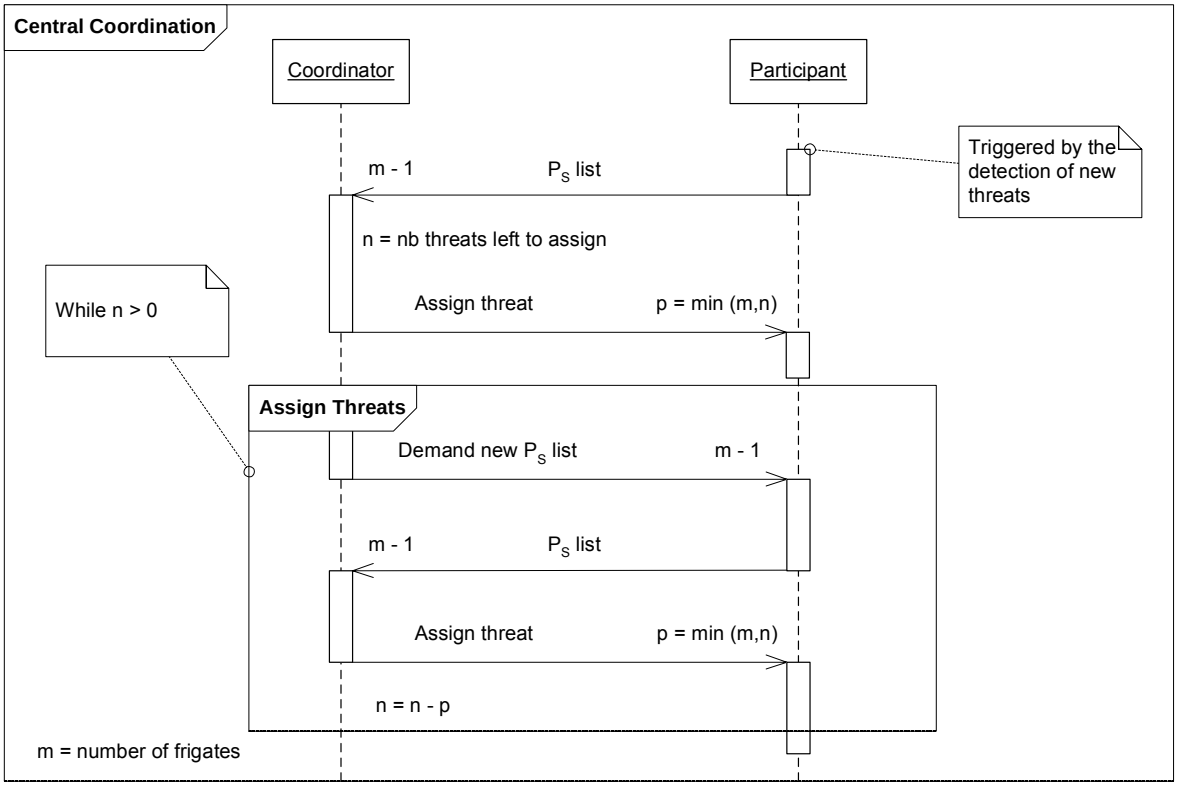

Figure 6.6: Central coordination protocol.

6.1.5 Contract Net

The Contract Net mechanism, explained earlier in Section 2.2.2, is similar to the central coordination mechanism, as it relies on a central coordinator and the use of communications. The difference between the two mechanisms is the number of threats assigned at one time. In the central coordination, we want to assign all threats at once, while we will allocate one threat at a time in the Contract Net mechanism. The following describes the Contract Net process:

1. The fleet chooses a coordinator.

2. For each threat detected, one at a time:

   (a) The coordinator asks every ship to send an estimated P_S value for this threat.

   (b) Each ship returns its estimated P_S value for this specific threat, considering already assigned threats.

   (c) The central coordinator chooses the best frigate to engage this threat and informs the agent.

The protocol described in Section 2.2.2 can then be adapted to our domain, as presented in Figure 6.7. In this figure, there are three types of messages exchanged between the coordinator and the other agents (participants), corresponding to the three steps presented. It is also interesting to compare the iterative process in this protocol to the one in the central coordination. We find that the only difference is the number of threats we allocate at a time.

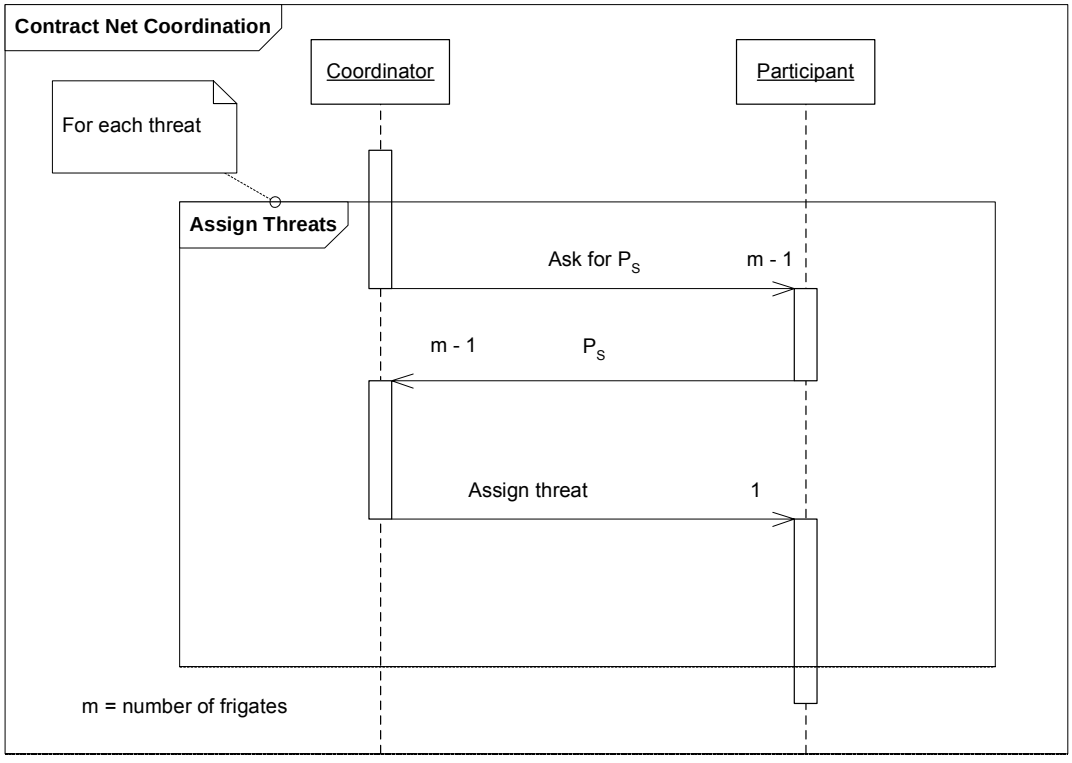

Figure 6.7: Contract Net protocol for the NEREUS problem.

The problems associated to the central coordination mechanism also apply to the Contract Net protocol, since both mechanisms are centralised coordination methods. However, the communication channels are more solicited in the Contract Net protocol than in the central coordination protocol, since more messages are sent. Thus, if the

communications are unsafe or unstable, the chances increase that they become elements of failure. Therefore, the central coordination will give more "reactive" (i.e., faster) responses, while the Contract Net approach will give better results but will also take longer. In addition, as the communication bandwidth decreases, the expected quality and timeliness of the coordinated solution obtained with this mechanism will decrease faster than with the central coordination method. Therefore, Contract Net is applicable only when there is sufficient time to coordinate. A more detailed comparison of these two mechanisms is provided in Section 6.3.

6.1.6 ∼Brown (Similarly Brown)

Another mechanism based on communications is the mechanism proposed by Brown (Brown and Lane [2000] and Brown et al. [2001]). This method can be used in a centralised or decentralised way.

This mechanism closely resembles the central coordination with added parameters. The main difference between the central coordination and the ∼*Brown* mechanism is that the threats are ordered by a priority evaluation before being distributed. This priority is based on three factors: the certainty that a threat is aimed at a specific ship, the relative importance of each ship and the fleet engagement capability for each threat.

Figure 6.8 illustrates the ∼Brown coordination protocol. This figure illustrates the broadcasts between the agents. As in the central coordination, we see that the *Assign Threats* process is iterated as long as there are still unassigned threats.

To transform the original centralised mechanism into a decentralised mechanism, the following assumptions must hold: 1) the agents must be entirely cooperative, 2) the agents must be homogeneous, 3) the protocol must be used in the same way for every agent and 4) the agents must be aware that the three preceding assumptions hold. These assumptions are required to make sure that every agent, receiving the same information, will evaluate the situation in the same way. Firstly, if the agents are not entirely cooperative (e.g., if they are from different nationalities), there is the possibility that one agent might defect, which is unacceptable. Furthermore, if the agents are not homogeneous, or if any part of the protocol uses information specific to a ship (such as ship's own ranking), the allocation might be evaluated differently among different ships. The fourth assumption is self-explanatory.

Thus, if these assumptions hold, a simple way to use the mechanism in a decentralised fashion is to broadcast all the information to every agent. Thus, since each

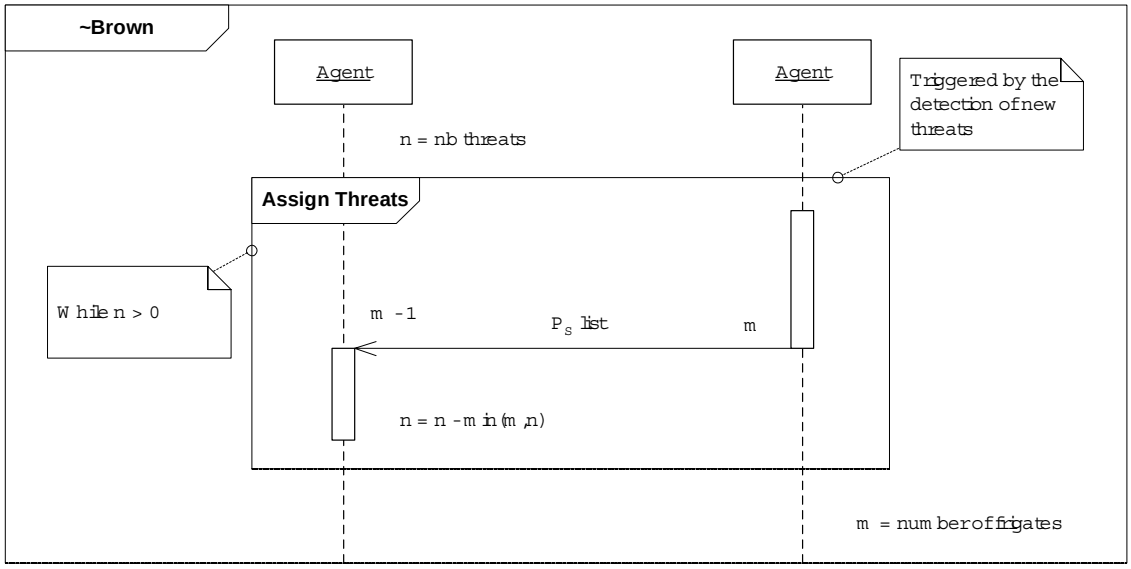

Figure 6.8: ∼Brown coordination protocol.

agent receives the same information and reasons the same way, each agent can deliberate and come up with an allocation solution. Moreover, this solution does not need to be sent to other agents since each one will find the same solution. Therefore, once a solution is obtained, an agent only needs to act on its assigned threats, as it is sure that the other agents will take care of their assigned threats[9].

The following process describes the steps of the decentralised version of this mechanism:

1. Before any threat is detected, each agent determines the relative weight of each ship and puts in a list of weights (W). The generic formula to compute the weights is:

   $$weight = rank \cdot x + y$$

   where we need to find x and y. The rank, as explained earlier, is a simple value of the relative importance of each ship. Thus, a ship with a ranking of 10 is more important than a ship with a ranking of 5. Since any agent knows the ranking of every ship, it also knows what the lowest and highest ranks are. In addition, an adjustable parameter is known: the *maximum weight deviation* (Dev_{max}), which is the desired difference between the highest and lowest ranking frigates' weights. Knowing this, we have that:

   $$1 - Dev_{max} = lowest \cdot x + y$$

[9]Obviously, in real-life systems, a verification method would be required to ensure nothing is overlooked due to system error.

$$1 = highest \cdot x + y$$

Thus,

$$x = \frac{Dev_{max}}{highest - lowest}$$

$$y = 1 - \frac{highest \cdot Dev_{max}}{highest - lowest}$$

2. When threats are detected, a matrix of threats' targeting probability (T) is created. A threat's targeting probability is easy to determine since, as we presented in Appendix A.1.1, the radar gives an exact evaluation of where a threat is heading. Thus, we know instantly which ship (frigate or cargo vessel) is targeted by each threat. Therefore, the *threats' targeting probability* matrix contains only Boolean values.

3. Each agent determines its P_S list and broadcasts it to the other agents.

4. A threat-weight matrix ($T \cdot W$) is calculated, which is a multiplication of the weight list with the targeting matrix.

5. Once each LP_S is received, the fleet engagement capability (P_F)[10] for each threat is determined. The fleet engagement capability (P_F) for a threat is computed by multiplying every P_S for this threat[11].

6. Each agent computes the fleet engagement priority matrix, which is $\frac{T \cdot W}{P_F}$, for each threat.

7. Each agent constructs a capability matrix which is the multiplication of the matrix composed of the LP_S of every frigate by the *fleet engagement priority* matrix ($\frac{T \cdot W}{P_F} \cdot MP_S$). Thus, this matrix represents how the threats should be engaged. It considers the relative weight of each ship, the possible targets of a each threat and the different capabilities of each frigate against the incoming threats.

8. Using an allocation algorithm, as in the central coordination mechanism, each agent determines the assignation of threats to ships.

A simple example shown in Figure 6.9 illustrates a simple situation. In this example, we consider there is uncertainty on the actual target of the threat. Thus, in this situation, we believe with a certainty of 80% that threat Th_1 is aimed at frigate F_A. Figure 6.10 illustrates the calculations required to determine the fleet engagement priority list for this specific example. Looking at the weights list, we see that the Dev_{max}

[10] referred to as *force performance* by Brown

[11] We will present methods to evaluate the P_F in Section 6.2.9

is 0.5. In this example, uncertainties in the T matrix are also shown. Once the MP_S is constructed (with every LP_S received), the P_F list can be computed for each threat. Then, we can use the results of T, W, P_F and MP_S to compute the final capability matrix. In this particular example, the complete lookup (i.e., best answer) would give the following assignations: (F_A-Th_1, F_B-Th_2, F_C-Th_3).

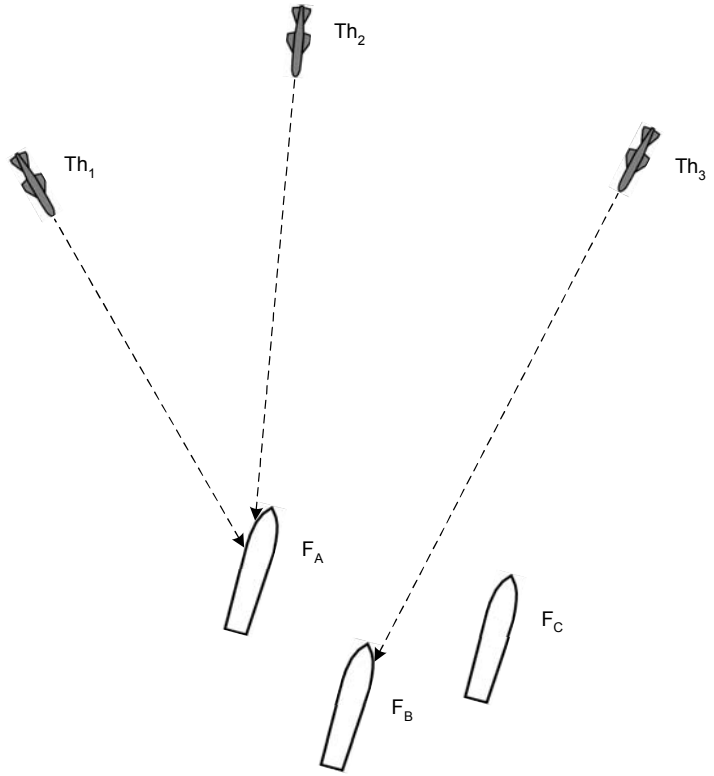

Figure 6.9: A simple AAW scenario.

6.2 Results of Preliminary Experiments

To evaluate the task distribution mechanisms developed, we ran two different types of tests. The first set serves to analyse the adjustment of coordination parameters, while the second compares the different coordination mechanisms. The tests were conducted on a dual Xeon 2.6 GHz, with 4 GB of RAM.

For the first set, we ran 1,000 tests for each value of the parameter to test. The average time required to run a coordination test was 900 milliseconds.

There are five parameters, common to each coordination mechanism, which we

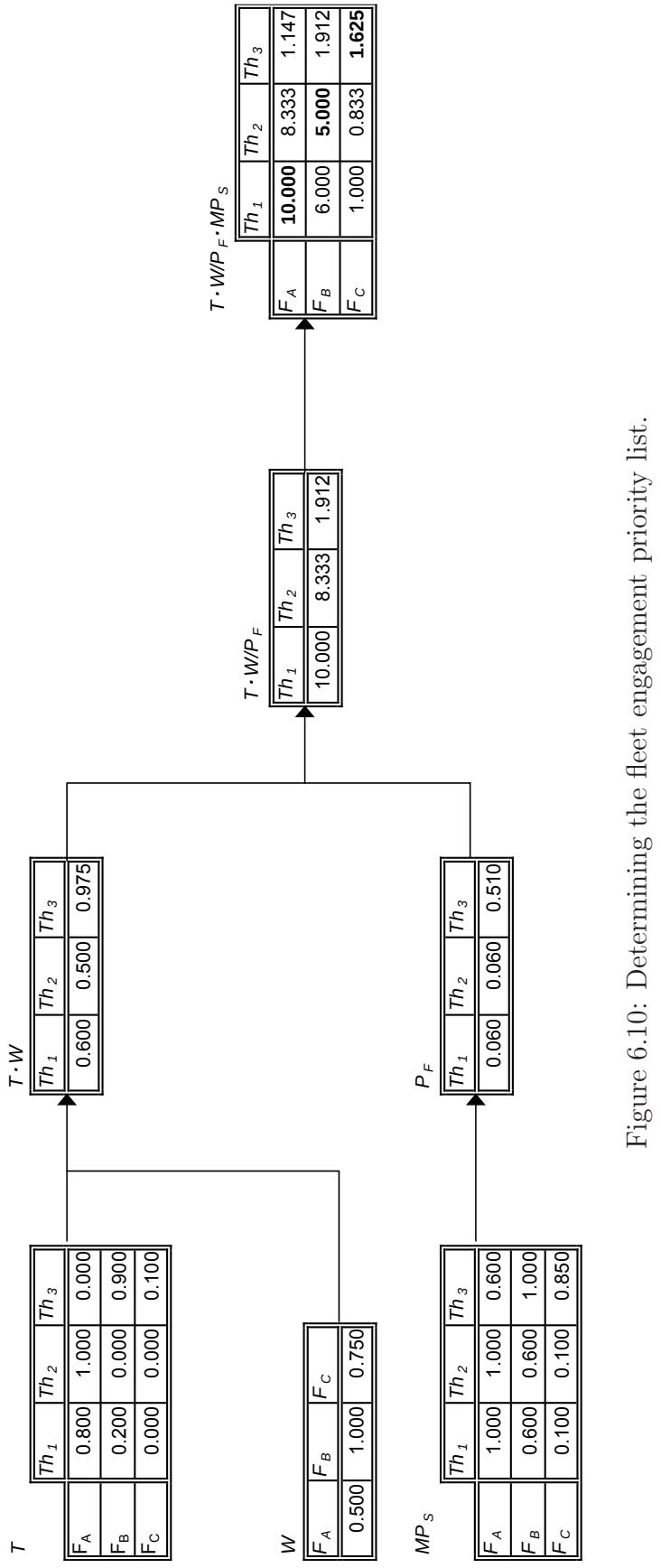

Figure 6.10: Determining the fleet engagement priority list.

tested: the *ships formation*, the *inter-ship distance*, the *communication preparation time*, the *bandwidth* and the *communication waiting time*.

The default coordination mechanism used for testing is the central coordination. When testing individual parameter values, we used the default settings presented in Table 6.2 for the other parameters, unless otherwise specified.

Coordination Mechanisms	*Parameters*	*Default Values*
All	Ship formation	Layout 1
	Inter-ship distance	500 m
	Comm. preparation delay	500 milliseconds
	Bandwidth	1,024 k/s
	Comm. waiting time	1,500 milliseconds
Contract Net	Frigates per threat	1
Central & ∼Brown	Allocation algorithm	Complete lookup
∼Brown	Max. weight deviation	0.5
	Fleet priority eval.	$T \cdot W / P_F$
	P_F	Multiplication
	Capability matrix	$\frac{T \cdot W}{P_F} \cdot MP_S$
Zone Defence	Backup	Active
	Threshold	0.95
	Nb frigates	4

Table 6.2: Default values for coordination parameters.

Thus, if we are testing different ships formations, the other parameters will be:

- *Coordination mechanisms:* Central coordination.
- *Inter-ship distance:* 500 m.
- *Communication preparation delay:* 500 milliseconds.
- *Bandwidth:* 1,024 k/s.
- *Communication waiting time:* 1,500 milliseconds.
- *Allocation algorithm:* Complete lookup.

We will now detail each parameter, give results for different values evaluated and discuss these results. In the presentation of the results, we use the term *scenario*. A

scenario is a typical naval battle; it starts when threats are detected and ends when each threat either has been destroyed or has hit the ship. When we specify the number of hits per scenario, it represents the average number of hits (on any ship) during a scenario. These results are separated by threat numbers. On the other side, the *average* number of hits per scenario represents the average number of hits per scenario, but over the whole set of scenarios (i.e. not separated by threat numbers). Also note that for the purpose of validation, the number of hits represents the number of ASMs that reach their target. Thus, when a threat reaches a ship, we evaluate whether this ship should be destroyed or not for the purpose of the "ships destroyed" metric. However, we continue the simulation as if the frigate was undamaged to have a better idea of the overall defence solution, not stopping after the first few hits.

In the following sections, the statistical error can be estimated, knowing that the scenarios are following a normal distribution. Thus, since we ran 1,000 tests for each parameter value of the tests separated by threat numbers, the statistical error is between 3.5% and 5%, with a confidence of 90%. The mean statistical error of these results is 4.02%. On the other side, for the results where the threats number are amalgamated, the number of tests for each parameter value is 8,000. Thus, the statistical error drops to an average of 1.34% (with a confidence of 90%). However, despite our best efforts, it is possible that the results are also tainted by systematic errors, yet undetected.

6.2.1 Ships Formation

The *ships formation* is the way the ships are positioned relatively to each other. The different formations presented here are defined in Appendix A.2.

Figure 6.11 presents the complete results for the different formations. In this figure, we can see that layout 7 is the best, almost with every number of threats. Figure 6.12, which shows the mean number of hits for each layout, also presents layout 7 as the best formation. It might be interesting to point out that the layout 1 is one of the commonly used formations in AAW, while being the most inefficient formation tested. It is also interesting to compare the similar layouts: (1 and 3), (2 and 4) and (5 and 6). It seems that each time, the formation with three ships aligned gives slightly better results than the formation where they are placed in a triangular disposition. Another interesting observation that can be made is that in the *reverse* formations: (1 and 2) and (3 and 4), it seems that the configuration where the three grouped ships point toward the centre is more efficient.

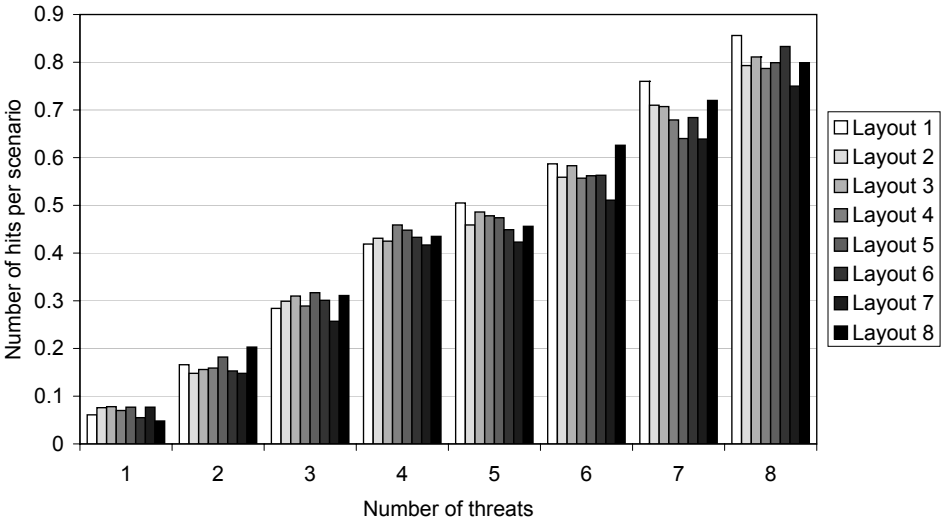

Figure 6.11: Number of hits per scenario, considering different formations.

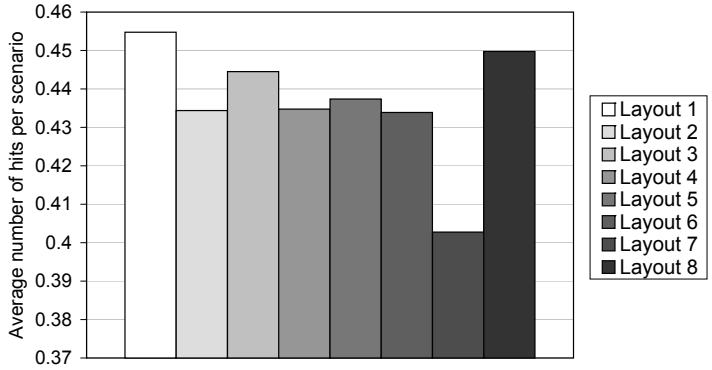

Figure 6.12: Average number of hits per scenario, considering different formations.

6.2.2 Inter-ship Distance

Since the effective range of the various weapon systems varies greatly, it is interesting to consider the standard distance between ships. Analysing the data presented in Figure 6.13, which is the number of hits per scenario considering the inter-ship distance, it is clear that any distance up to 1,000 metres is roughly equivalent, while the efficiency decreases rapidly past 1,500 metres. Furthermore, this difference becomes clearly marked beyond three threats in a scenario. This can be explained by the fact that the farther apart the ships are, the longer the STIR is used to guide a SAM intercepting a threat aimed at another ship. While a STIR is occupied, no other gun or SAMs can be used on this side of the ship. In Figure 6.13, there is also a gap in the number of hits between 2,000 and 2,500 metres. This is related to the fact that the range of the CIWS, which is 2,500 metres. Figure 6.14 makes it clear that on the average, it is best to be at a distance of 1,000 metres. Indeed, this inter-ship distance leaves enough space to manoeuver and position the frigates, which would not be possible with an inter-ship distance of 50 metres.

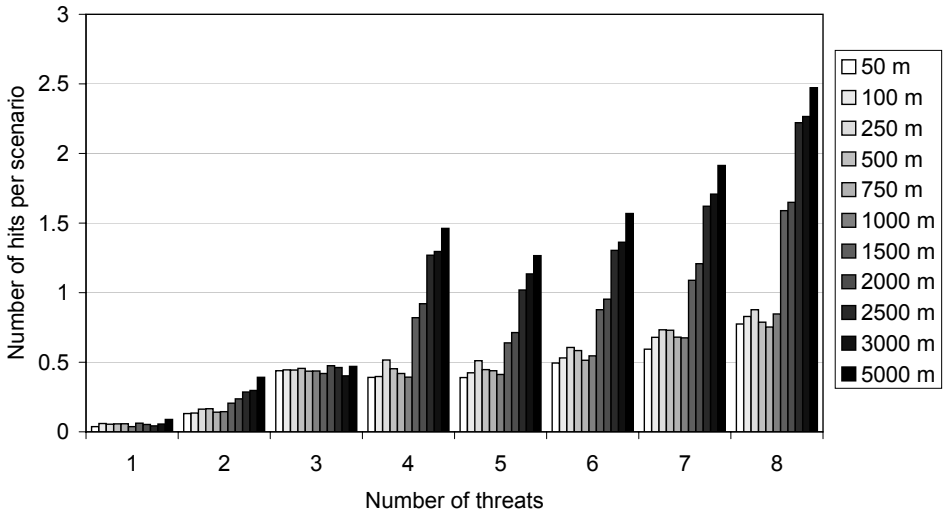

Figure 6.13: Number of hits per scenario, considering different inter-ship distances.

6.2.3 Communication Preparation Delay

The *communication preparation delay* is a time added to delay the communications. It represents the time to correctly prepare a message with security measures and the correct headers. This is invariable and independent of the size of the messages. At first, we believed that increasing this delay would result in inefficient planning, since

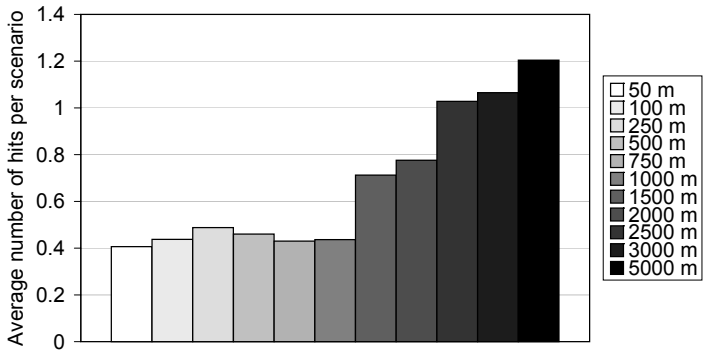

Figure 6.14: Average number of hits per scenario, considering different inter-ship distances.

the agents could not obtain the same timeliness. However, the results in Figure 6.15, which presents the number of hits as a function of the communication preparation delays, shows that the solutions obtained are mostly equivalent, as no parameter value seems to be better in every situation. A good way to see if solutions are equivalent is to look at the number of SAMs planned. Indeed, the engagement of a threat with a SAM requires a certain, invariable time. If the delay induced by communication time is too great, then the agents will not have enough time to use actions such as SAMs. In Figure 6.16, we see that there is almost the same number of SAMs planned for every parameter value. This means that in most scenarios, the agents have enough time to communicate before a soft deadline is met. However, we used the central coordination, which is not the most communication intensive mechanism, to evaluate the impact of this parameter. Thus, having a high communication preparation delay in Contract Net coordination could significantly reduce the efficiency as the number of threats increases. This is due to the fact that the Contract Net mechanism considers threats one at a time, and therefore uses many communications. Since a communication preparation delay is added to each communication, the total added delay is proportional to the number of messages.

Furthermore, it is clear that the results shown in Figures 6.15 and 6.16 are also due to the fact that after the coordination has been done, each frigate only intercepts a limited number of threats (usually 1 or 2). Thus, it is possible to intercept the incoming ASMs a little later without important degradation in the plan quality, as the STIRs will not be used after the incoming threats are destroyed.

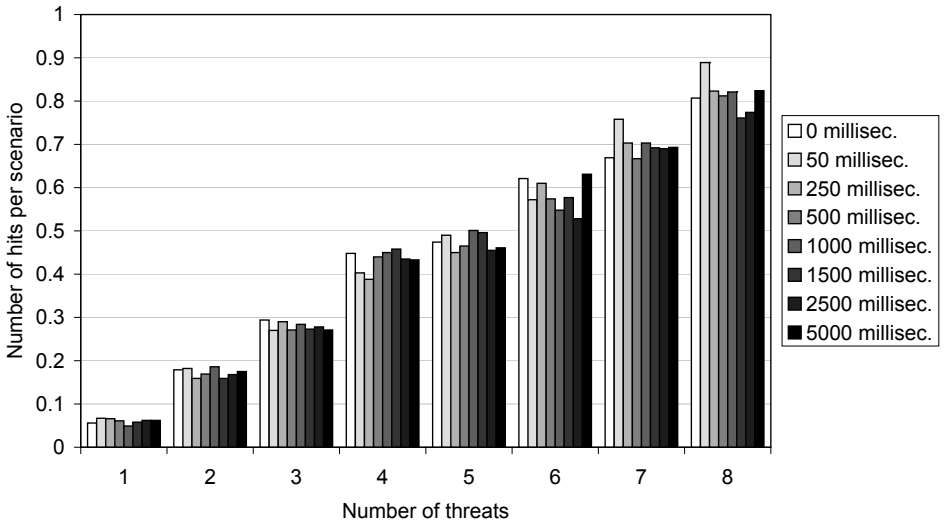

Figure 6.15: Number of hits per scenario, considering different communication preparation delays.

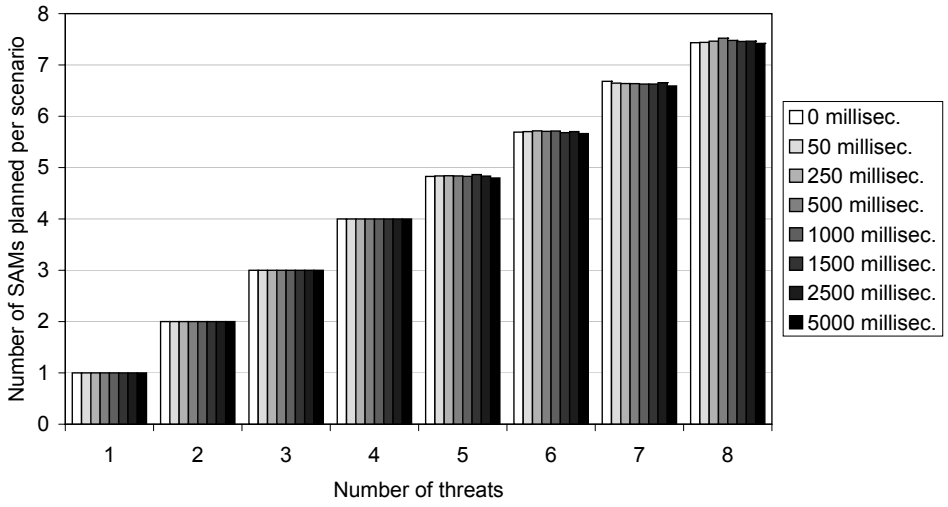

Figure 6.16: Number of SAMs planned per scenario, considering different communication preparation delays.

6.2.4 Bandwidth

In our approach, the bandwidth of the system's communication channel is fixed for the length of the simulation. Before starting simulations, the bandwidth can be reduced to represent background noise or degraded communication conditions. Figure 6.17 shows the average number of hits per scenario, for tests considering different bandwidth values from 1 k/s to 8192 k/s. In this figure, we see that even at 1 k/s, which is low by modern standards, the number of hits per scenario is much the same. Figure 6.18, representing the average weight of a message, shows that a standard message is around 1040 bytes in size. The size does not vary much depending on the number of threats, since the message core is relatively small, with the rest being the security and transport headers. This explains why the results are not very conclusive: at the lowest bandwidth tested (1 k/s), it takes only 1 second to send a message. As we have discussed earlier, the central coordination mechanism introduces a delay of a few seconds while retaining almost the same quality.

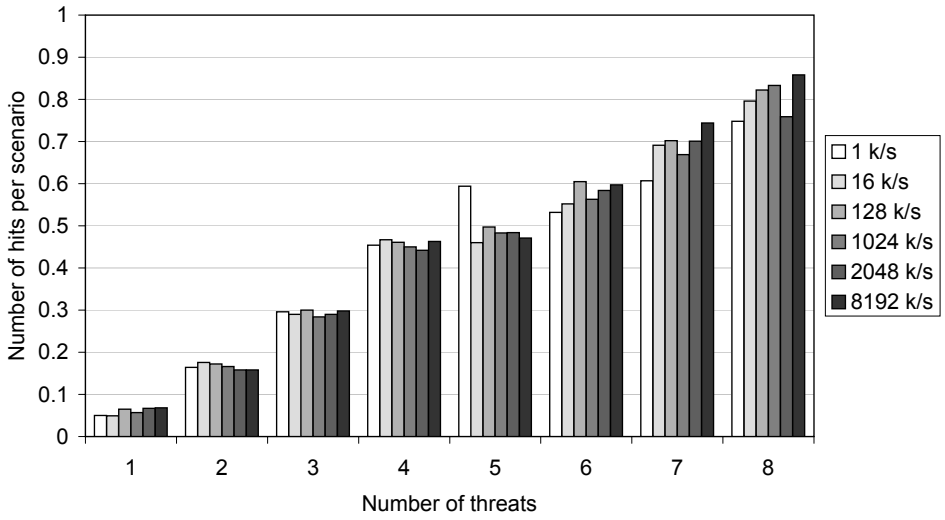

Figure 6.17: Number of hits per scenario, considering different bandwidth values.

6.2.5 Communication Waiting Time

The *communication waiting time* parameter sets the available time for an agent to deliberate and return an answer. Thus, when waiting for a reply, an agent will wait a specific time defined by: *Time to send the initial message + Communication waiting time + Estimated time to receive the reply.*

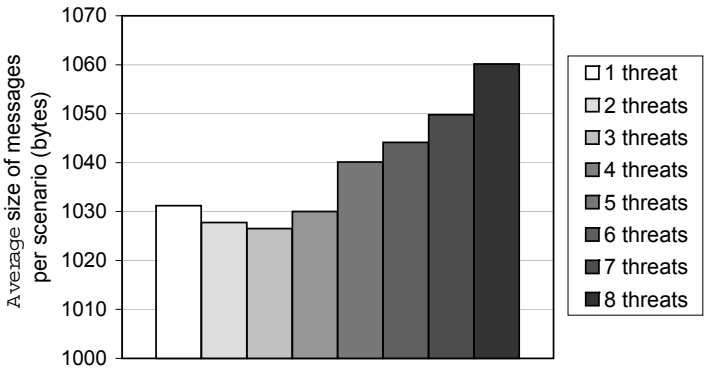

Figure 6.18: Average size of messages per scenario.

We see in Figure 6.19, presenting the number of hits considering communication waiting times, that the waiting time does not have much impact on the observed results. The same discussion as provided for the *communication preparation delay* applies here. The fact that the values show similar results can be explained in the following way. At first, we know that computing the P_S value is really fast, since it is a very simple heuristic. Thus, the time required for an agent to determine its P_S list and send it to the coordinator is less than 50 milliseconds. Therefore, when the agents have 50 milliseconds or more, they have the required time to provide their answers and the coordination mechanism is used as usual. Furthermore, even when the agents are allowed 0 milliseconds to deliberate, the results are not conclusive. This is due to the reply evaluation heuristic used. This heuristic has the tendency to slightly overestimate the size of the reply messages. When the reply size is overestimated, this gives a little more time for the agents to deliberate and reply, which is enough for the simple task of computing the P_S in the central coordination mechanism. On the other hand, should the time to transmit the reply be underestimated, replies might not be given in time, resulting in a loss of efficiency.

6.2.6 Number of Frigates per Threat

This parameter controls the number of frigates that will engage each incoming threat. The minimum number of frigate to intercept a threat is one, while the maximum is the number of frigates in the scenario. This parameter was tested with the Contract Net protocol (which makes it trivial to assign more than one frigate for each threat), with a constant number of four frigates per scenario.

Figure 6.20 presents the average number of hits per scenario, given the number of threats and the number of frigates assigned to engage each threat. In this figure, we

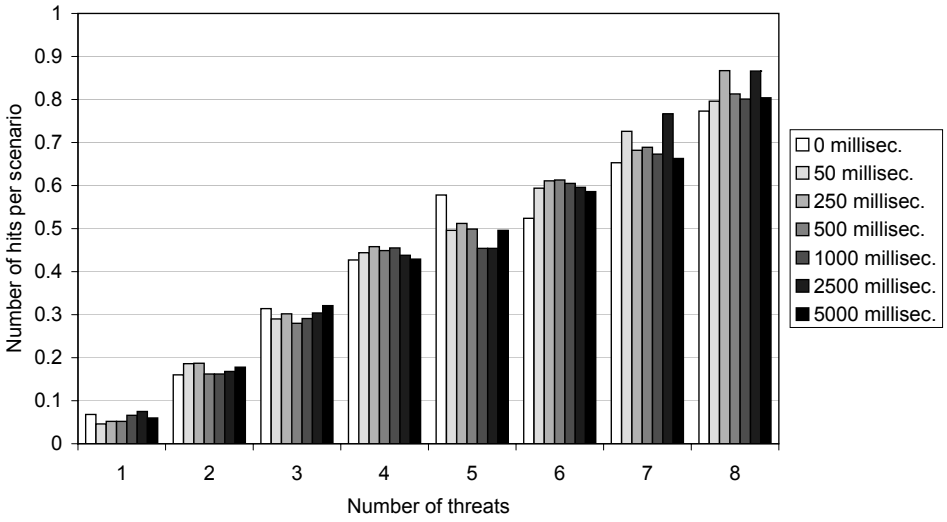

Figure 6.19: Number of hits per scenario, considering different communication waiting times.

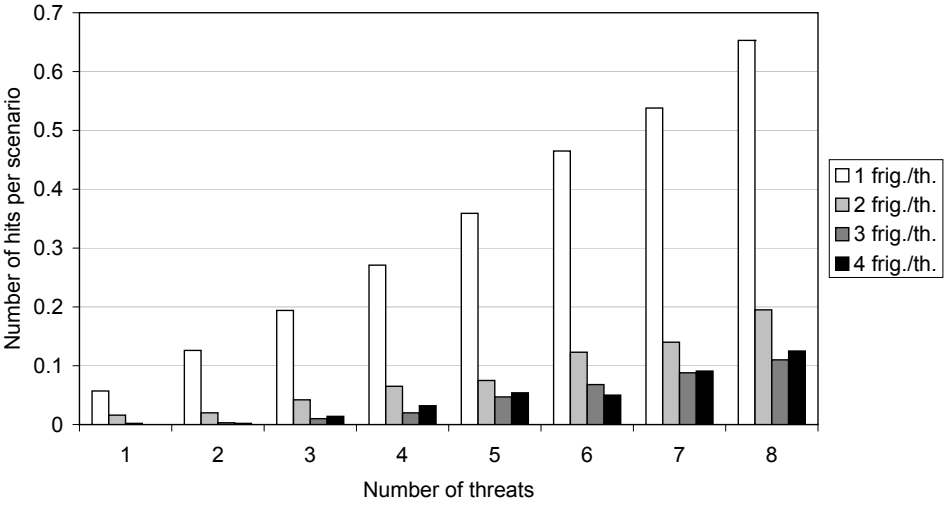

Figure 6.20: Number of hits per scenario, considering different numbers of frigates per threat.

see that increasing the number of frigates engaging the threats results in fewer hits per scenario. On the other side, Figure 6.21 presents the number of SAMs planned for each frigates/threat ratio. As expected, the number of planned SAMs increases with the ratio of frigates per threat ratio. We see interesting in Figure 6.22, which presents the efficiency, given the number of frigates per threat. This efficiency is the percentage of threats destroyed, divided by the number of SAMs used to destroy them $((1 - \%\ of\ hits)/nb\ SAMs\ used)$. In this figure, we see that the most efficient ratio is one frigate/threat. However, in our case, the survivability is far more important than the total of resources used. Thus, if we transform the efficiency calculus by adding an α parameter to stress the fact that it is bad to be hit by threats, the efficiency becomes: $((1 - \alpha \cdot \%\ of\ hits)/nb\ SAMs\ used)$. Figure 6.23 shows the efficiency calculated with an α of 10. In this case, we see that the best ratio becomes two frigates per threat.

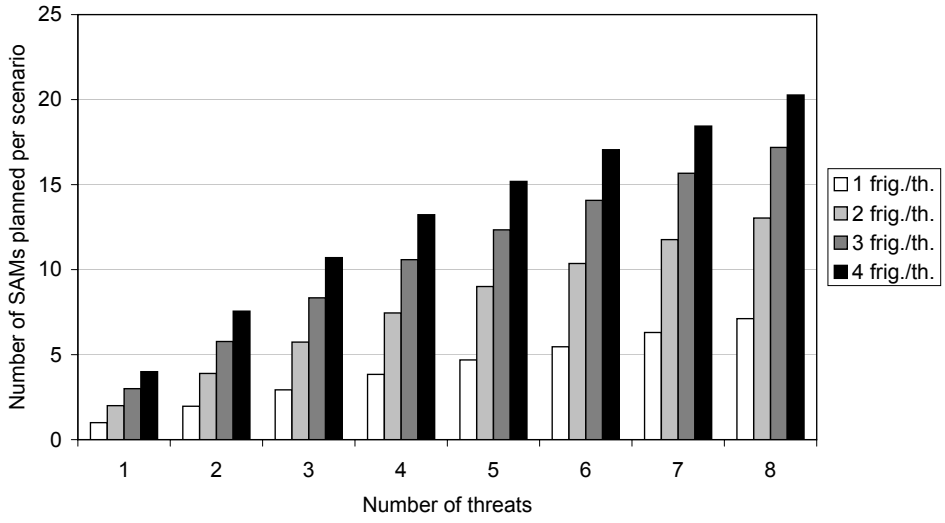

Figure 6.21: Number of SAMs planned per scenario, considering different numbers of frigates per threat.

6.2.7 Allocation Algorithm

The allocation algorithm serves in the *central coordination* and the ∼*Brown* coordination mechanisms. It is used to assign the threats once a capability matrix has been constructed (respectively steps 3 and 7 of central and ∼Brown coordination mechanisms). The *complete solution lookup* generates every possible solution and choose the best one, while the *greedy algorithm* uses a greedy approach to determine the assignation. We consider that both algorithms will assign at most one threat to each frigate. If there are more threats than frigates, some threats will be left unassigned in the first

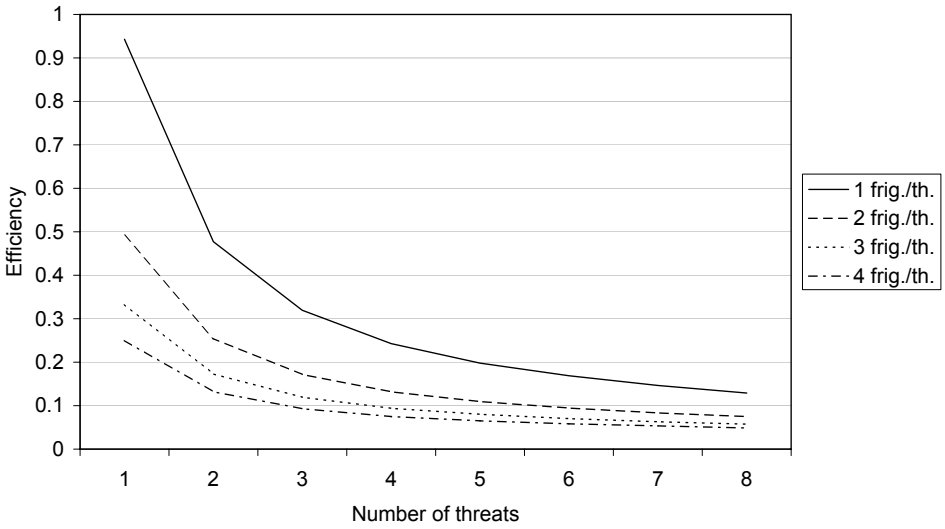

Figure 6.22: Efficiency, considering different numbers of frigates per threat.

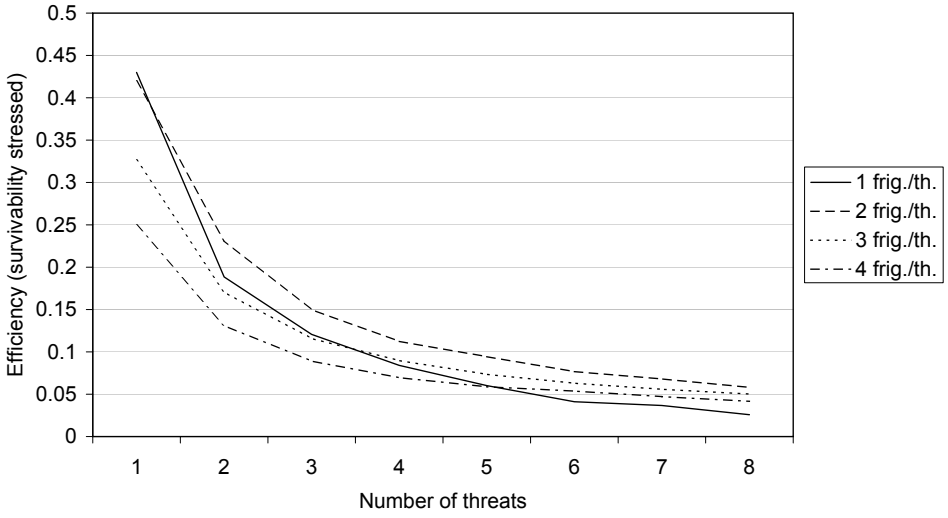

Figure 6.23: Efficiency with survivability stressed, considering different number of frigates per threat.

	Th_1	Th_2	Th_3
F_A	0.6	0.75	0.85
F_B	0.4	0.5	0.8
F_C	0.7	0.65	0.3

Table 6.3: Capability matrix for a simple situation.

round of allocation and will have to be allocated in a second round (as discussed in Sections 6.1.4 and 6.1.6). On the other hand, if there are more frigates than threats, the algorithm will choose only the best frigates to engage the threats. The situation illustrated in Table 6.3 represents a simple capability matrix[12], where we have three threats (Th_1, Th_1, Th_1) and three frigates (F_A, F_B, F_C). The values of this matrix represent the estimated LP_S of each frigate. Considering this situation, the *complete solution lookup* would give the following assignations: (F_A-Th_2, F_B-Th_3, F_C-Th_1), while the greedy heuristic would give (F_A-Th_3, F_B-Th_2, F_C-Th_1).

Looking at the results in Figures 6.24 and 6.25, we see that there are no marked differences between both allocation algorithms. After a deeper analysis of individual results, we determined that only in rare cases do the two algorithms give different answers. Thus, for the average number of hits, the choice of algorithm makes no real difference. However, the greedy algorithm is quicker, since it does not need to explore the complete solution set. Therefore, in the long run, the use of a greedy heuristic seems appropriate to assign the threats to individual frigates.

6.2.8 Maximum Weight Deviation

Used in the ∼*Brown* coordination mechanism, the ship weighting is related to each ship importance. Once obtained, the weights are normalised in such a way that the maximum weight is 1 and the minimum is $(1 - Dev_{max})$[13].

Increasing the value of this parameter will emphasise the importance of ships with higher ranking. If the maximum weight deviation is set to 1, the ships with the lowest ranking will be left completely undefended. Figure 6.26 shows the number of hits for different weight deviations. On the other hand, setting the maximum deviation to 0 will give equal importance to every ship, effectively lowering the defence of the more important ships.

[12]The capability matrix was introduced in the central coordination and ∼Brown mechanisms sections. Capability matrix are further detailed in Section 6.2.10

[13]Presented in Section 6.1.6

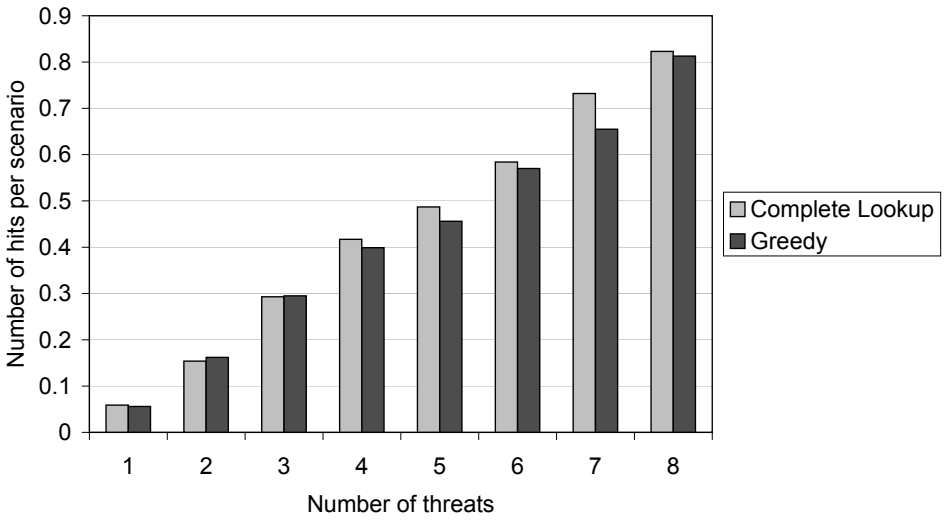

Figure 6.24: Number of hits per scenario for the central coordination, given the allocation algorithms.

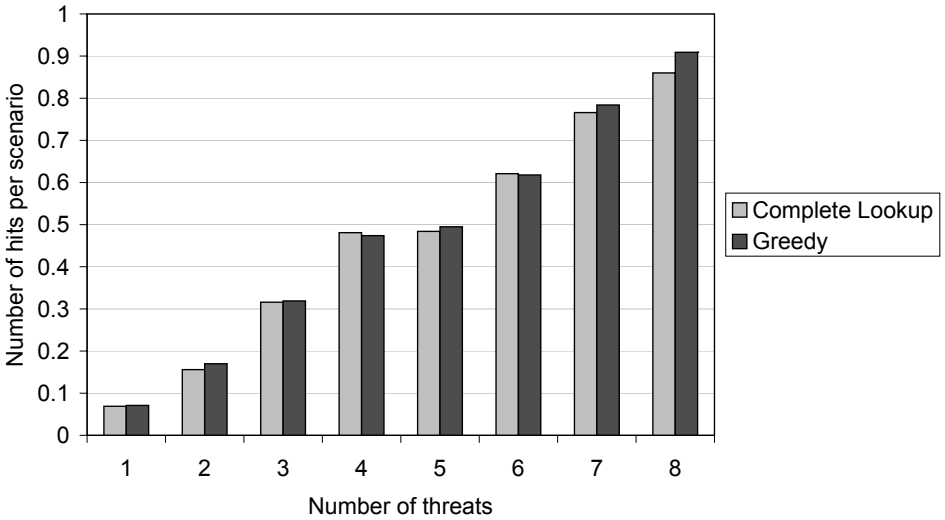

Figure 6.25: Number of hits per scenario for the ∼Brown coordination, given the allocation algorithms.

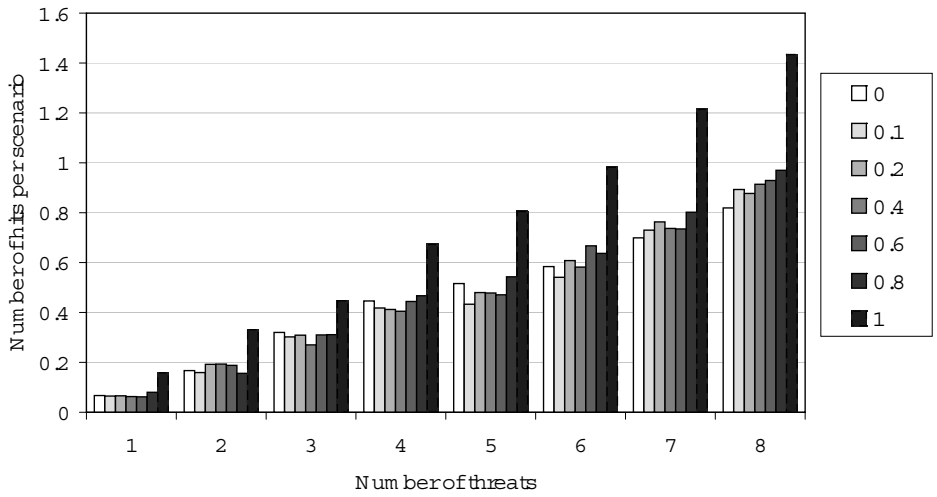

Figure 6.26: Number of hits per scenario, considering different maximum weight deviations.

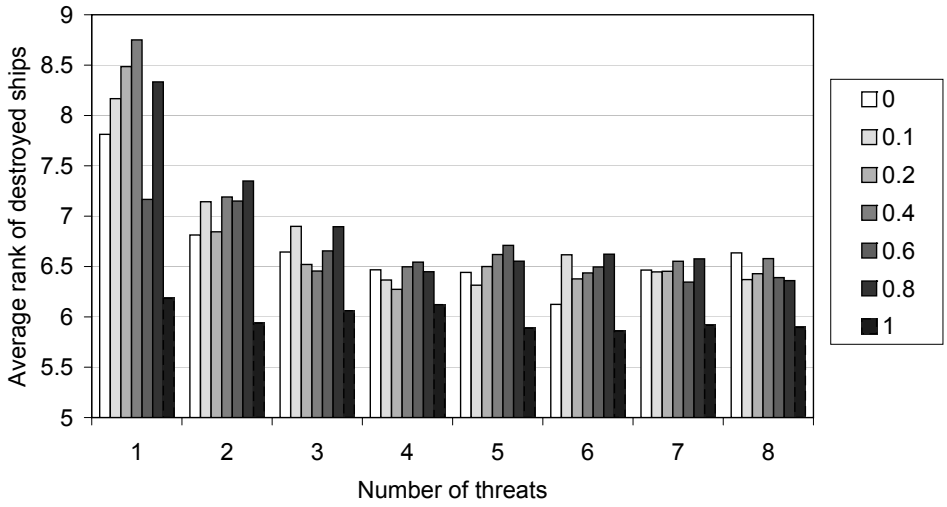

Figure 6.27: Average rank of destroyed ships, considering different maximum weight deviations.

In Figure 6.27, we see that the average ranking of destroyed ships is almost constant for every value, except when the maximum deviation is 1. In this case, there are more ships hit (as shown in Figure 6.26), but they are mostly ships of lower importance, since we really put the emphasis on protecting the highly valued unit.

6.2.9 Fleet Engagement Priority Evaluation

Used in the ∼*Brown* method, the priority of each threat, at fleet level, is evaluated from the received probabilities of success (P_S) for each frigate. There are three different fleet engagement priority (P_F) evaluation methods: the mean of P_S, the highest P_S and the multiplication of P_S. Tables 6.4, 6.5 and 6.6 use the example of Figure 6.10 and shows results of the three different evaluation methods.

Th_1	Th_2	Th_3
1.00	1.00	1.00

Table 6.4: P_F (Highest).

Th_1	Th_2	Th_3
0.57	0.57	0.82

Table 6.5: P_F (Mean).

Th_1	Th_2	Th_3
0.06	0.06	0.51

Table 6.6: P_F (Mult.).

Results for these evaluation methods are shown in Figures 6.28 and 6.29. We discover that the three evaluations give very similar results. Thus, considering the statistical error on those results, a definitive statement cannot be issued on the efficiency of these evaluation methods; we can consider that the three will give similar results.

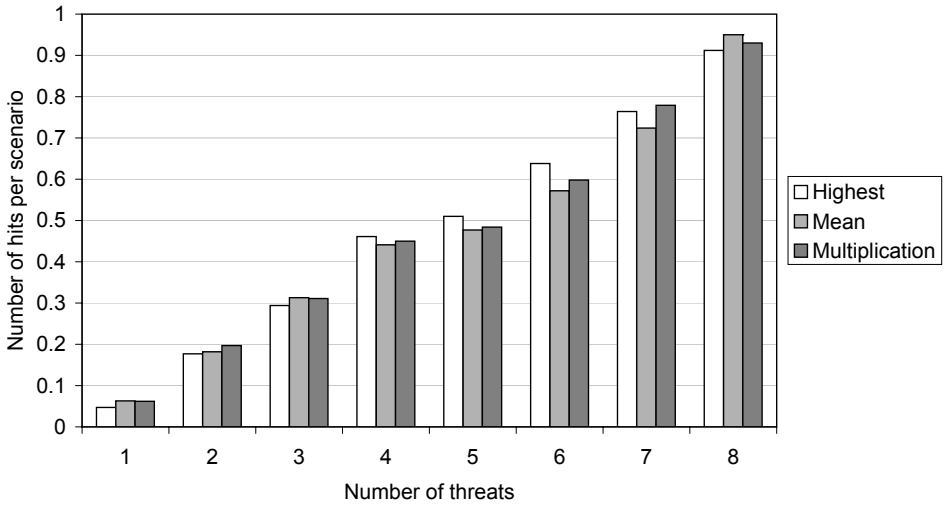

Figure 6.28: Number of hits per scenario, considering different fleet engagement priority evaluations.

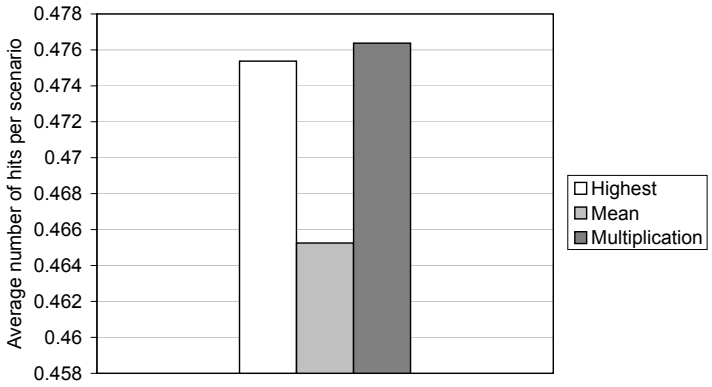

Figure 6.29: Average number of hits per scenario, considering different fleet engagement priority evaluations.

6.2.10 Capability Matrix Evaluation

Also used in the ∼*Brown* mechanism, the capability matrix is evaluated by each ship (step 7 of the ∼Brown process), before using the allocation algorithm to determine how to assign the threats. What we call the *fleet engagement priority* matrix is presented by Brown and Lane [2000] as the "prioritised force level threat table" and is evaluated in the following way (also shown in Figure 6.10):

$$\frac{T \cdot W}{P_F}$$

Once this matrix is obtained, the multiplication of $\frac{T \cdot W}{P_F}$ by the P_S matrix (MP_S) will give what we call the *capability matrix*.

Therefore, the standard capability matrix is $\frac{T \cdot W}{P_F} \cdot MP_S$. However, we tried four other different ways to determine the capability matrix, which are shown in Figures 6.30 and 6.31. These present respectively the average number of hits per scenario and the average rank of destroyed ships. Note that the MP_S *alone* method is the one used to build the capability matrix of the central coordination mechanism.

The results in Figures 6.30 and 6.31 demonstrate that a deeper analysis would be required to determine the best capability matrix evaluation, since there is significant statistical error on these results (an average of ±0.063 and ±0.288 for Figures 6.30 and 6.31 respectively).

Even so, we can examine the comparative results of $\frac{T \cdot W}{P_F} \cdot MP_S$ and $T \cdot W \cdot P_F \cdot MP_S$ in Figure 6.31. When we *multiply* the $T \cdot W$ by P_F, we give greater importance to threats that the fleet is more confident of being able to intercept. On the contrary,

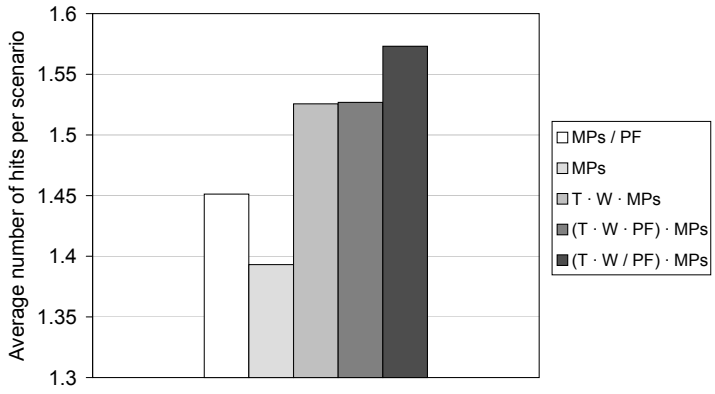

Figure 6.30: Average number of hits per scenario, considering different capability matrix evaluations.

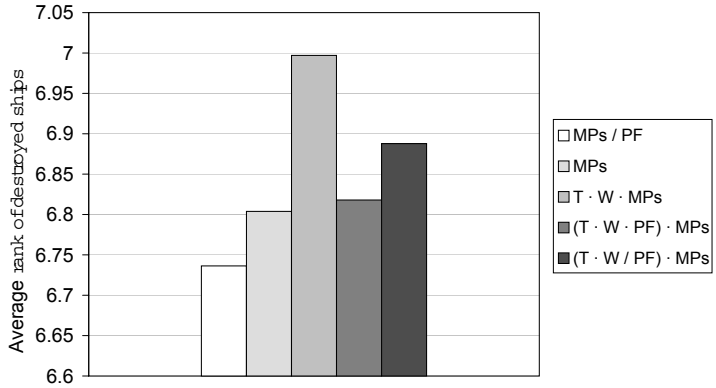

Figure 6.31: Average rank of destroyed ships, considering different capability matrix evaluations.

dividing $T \cdot W$ by P_F tends to prioritise threats harder to intercept. Thus, these results hint that it could be better to invest time and resources on threats that have more chances of being intercepted.

6.2.11 Threshold

Used in the *zone defence* coordination method, this is a threshold (on the estimated P_S) under which a ship will seek assistance in the engagement of a threat[14].

As we see in Figure 6.32, the best value for the threshold parameter, *given good communications*, is 0.98. However, as shown in Figure 6.33, the number of messages sent with this value is rather high, though it is still low compared to the other mechanisms. When a threshold of 0.98 is used, a frigate will usually engage only one threat before asking for help. If we consider two threats coming at the frigate (CPA = 0), we see that the P_S is equal to P_K (see equation 6.1). Looking at equation 6.2, we see that the P_S for the second threat is 0.97898, thus below the 0.98 threshold.

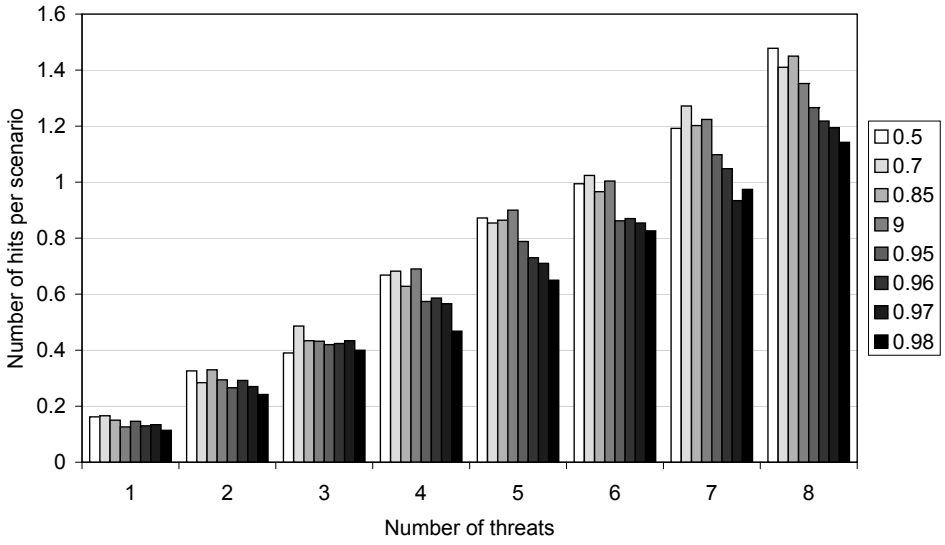

Figure 6.32: Number of hits per scenario, considering different threshold values.

[14]This process of seeking assistance is explained in Section 6.1.3.

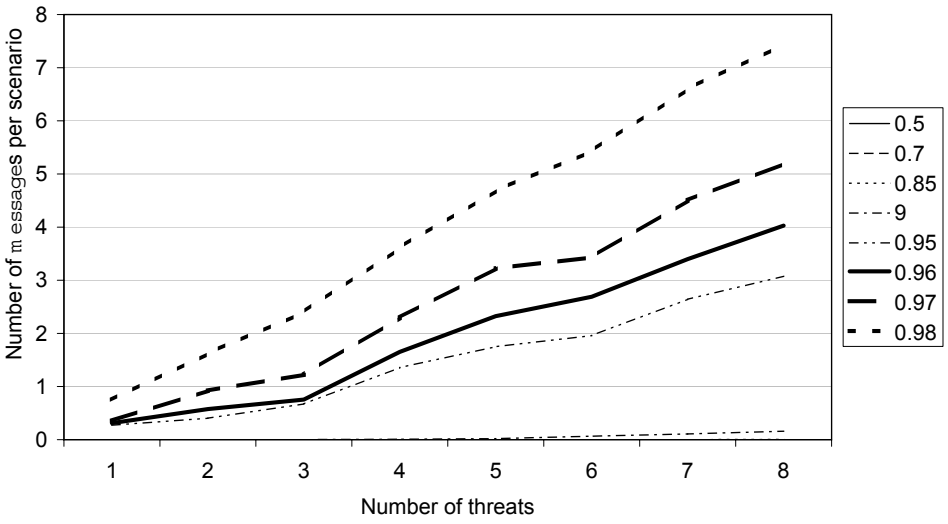

Figure 6.33: Number of messages per scenario, considering different threshold values.

6.2.12 Number of Frigates

Used only in the *Zone defence coordination* mechanism, this is the number of frigates in the scenarios to test. It is used to evaluate the effects of having more or less defending ships in an AAW scenario.

Figure 6.34 illustrates the number of hits for different numbers of frigates in a scenario. The results, which might seem a little counter-intuitive, can be explained easily. We already know that in the zone defence mechanism, the threats are assigned by their relative bearing from the point of view of the cargo vessel. It is important to note that it is the bearing *at the time of detection*. Thus, it is possible that an incoming ASM is not targeting the ship defending the sector it is in at the time of detection. In these cases, the threat will be engaged by the ship defending the sector where the threat appeared, not the ship that is targeted. In Figure 6.35, showing a simple scenario, the targeted frigate is F_A, while it is F_B that will engage the threat. We discussed earlier[15] that when threats are not engaged by the frigate they target, the overall fleet defence efficiency is decreased. When there are more frigates, the defence zones are smaller and the chances that a threat is not engaged by its target increases. Therefore, this explains the results of Figure 6.34 that show that when there are more frigates, the efficiency decreases.

[15]See Section 6.2.10

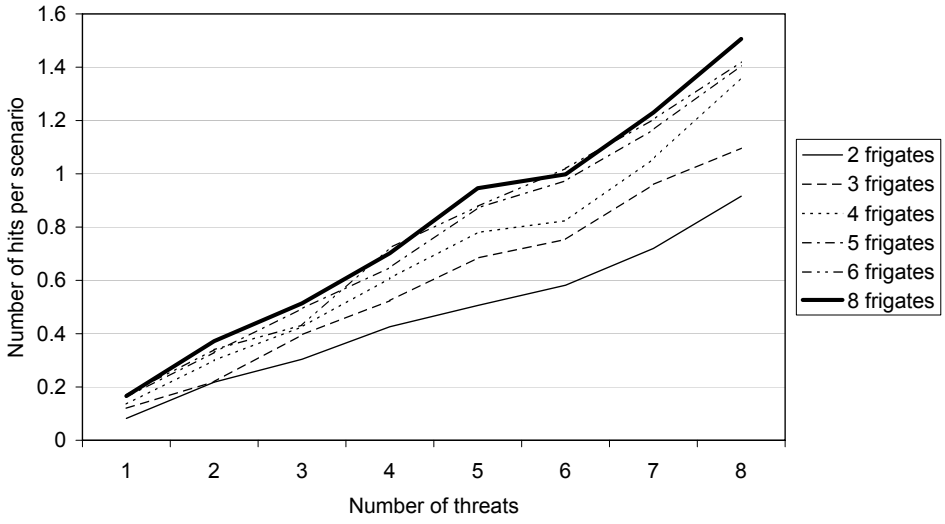

Figure 6.34: Number of hits per scenario, considering different numbers of frigates in a fleet.

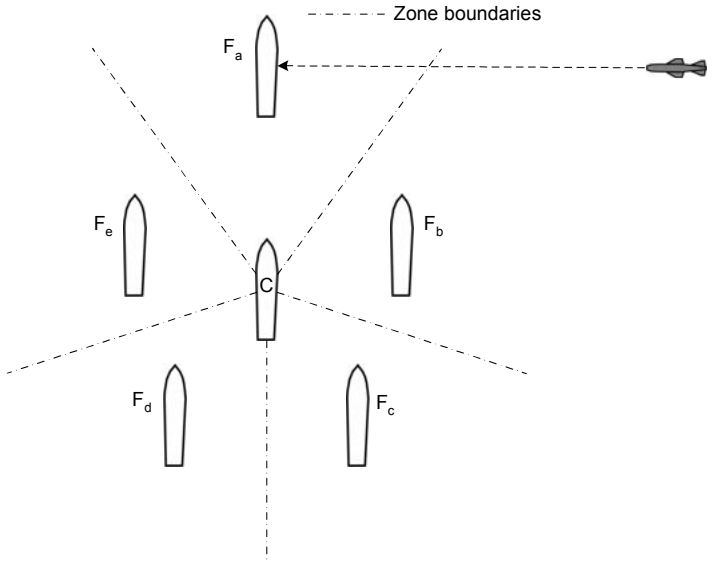

Figure 6.35: Effects of many frigates in a fleet.

6.3 Comparison

We have seen how different parameter values can influence the coordination mechanisms. Thus, the fine-tuning of these parameters is crucial to develop a good mechanism. However, we can analyse the different protocols independently of the results to determine their relative strengths and weaknesses.

The following are attributes of coordination mechanisms:

- *Type of coordination method:* As we presented in Section 2.2.2, the coordination mechanisms can be separated into three types: communication-based mechanisms, social laws mechanisms and mechanisms based on learning. Knowing the type of each one is useful to determine which factors are more important to this mechanism.
- *Communications:* As we have discussed earlier, the communications are important in many coordination mechanisms. Thus, knowing the number and importance of communication is important to know which mechanisms will be more sensible to degradation in the communication environment.
- *Centralised:* Some mechanisms use centralised information and decision-making. In MAS, it usually is believed that a centralised method is less robust than an equal but decentralised method. Indeed, in centralised mechanisms, the failure of a single agent (the coordinator) can make the process abort, or at least significantly reduce the quality of the solution. Furthermore, a centralised mechanism implies a certain hierarchy and authority structure. While this structure is present in most military contexts, there are systems where having an authoritative agent might be unwelcome.
- *Ship importance:* We have seen that in some cases some ships are more important than others. A ship with higher importance could be a commanding ship, an escorted supply ship, a coalition ship, etc. Some mechanisms deal with the relative importance of ships, while others do not take into account these ranking methods.
- *Other agent models:* Does a mechanism need a good model of the other agents to be able to efficiently coordinate with them? Acquiring and maintaining a model of other agents is an arduous process in the best case, usually requiring some training situations. Please note that needing a good model of others is different from believing that every agent is homogeneous, as presented in Section 6.1.6.

- *Backup plan:* In a stochastic environment such as project NEREUS, it is important to be able to implement and use backup plans, in case an agent is unable to take care of its assigned tasks. Thus, this attribute represents the possibility of integrating such backup plans in the mechanism.

- *Completeness:* In project NEREUS, it is unacceptable to let threats reach ships unimpeded. Therefore, the completeness of a solution (whether every task is distributed) is important.

The evaluation of these factors for each coordination mechanism is presented in Table 6.7, which compares the different coordination mechanisms. In this table, m is the number of frigates, n is the number of threats and p is the number of times a contingency arises. In the case of the *zone defence*, p is the number of times the P_S valuation is below the threshold; for the SWAT mechanism, p is the number of times an agent detects that the plan will fail if it does not communicate information with the other agents. The SWAT mechanism has not been implemented but is proposed as an idea for future researches. It will be discussed in Section 7.1.

6.3.1 Metrics

The metrics used to evaluate the different coordination mechanisms are: *the total number of communications*, the *survival rate* and the *efficiency*, which is the survival rate considering the resources used. As discussed in Section 6.2.6, we can add an α parameter to the efficiency evaluation to stress the importance of survivability versus the used resources. We looked at two different efficiency measures: the efficiency according to the SAMs launched and according to the total size of messages sent.

In the following figures, we compare the results for different coordination mechanisms. We used many different values for the various parameters described earlier, and we averaged the results to get a good idea of the performances of the mechanisms. However, since the *frigate per threat* parameter had too great an impact on the results of the Contract Net mechanism, we also included the results for the Contract Net protocol with only one frigate per threat, which is noted "Contract Net*"in the figures. In "Contract Net", scenarios with 1 to 4 *frigates/threat* were tested. The relative differences between "Contract Net*" and "Contract Net" provide an idea of the performances that could be obtained if we adapted the other mechanism to permit assigning a threat to more than one frigate. Figure 6.36 presents the number of SAMs used in the different mechanisms. We see that most mechanisms will fire about the same number of SAMs,

	Zone defence	Central coordination	Contract Net	∼Brown	SWAT
Coord. type	Social laws	Communications	Communications	Communications	Learning
Number of comm.	$4p$	$m + n\%(m+1)$ $+2(m-1)\cdot\lceil\frac{n}{m}-1\rceil$	$n \cdot (2m-1)$	$m(m-1)\lceil\frac{n}{m}\rceil$	$(m-1)p$
Comm. importance	Useful	Very Important	Very Important	Very Important	Useful
Centralised		✓	✓		
Ship Importance				✓	✓
Agent models					✓
Backup plans	✓	✓	✓	✓	✓
Completeness	✓	✓	✓	✓	✓

Table 6.7: Comparing coordination mechanisms.

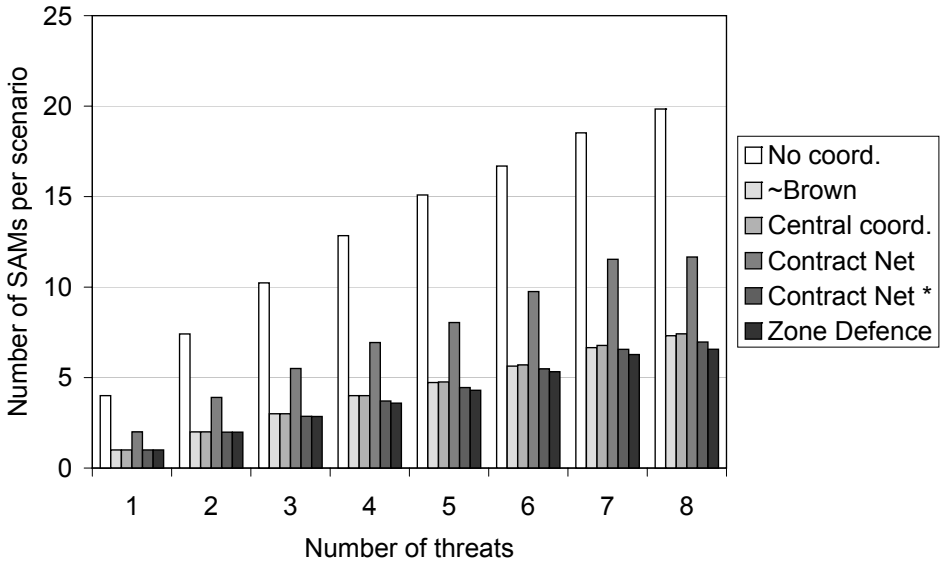

Figure 6.36: Number of SAM planned per scenario, considering different coordination mechanisms.

except for the Contract Net used with more than one frigate per threat. The uncoordinated scenarios will also use more SAMs, which is normal since in this case, every frigate considers it must defend the fleet alone.

Figure 6.37 illustrates the size of communications, depending of the number of threats and mechanisms used. Since the messages are roughly the same size, the graph is also representative of the *number* of communication that occurs. It is also interesting to see that the results are coherent with the analysis provided in Table 6.7. Also note that there are no communications in uncoordinated scenarios.

Figure 6.38 shows the number of hits per scenario, comparing the coordination mechanisms. We see that the number of hits is greater in the *zone defence* mechanism. This can be explained mostly by the discussion in Section 6.2.12, which states that the division in sectors around the frigate generates many situations where the plan is sub-optimal. However, we see that this mechanism also uses far less communications than the other ones. Therefore, approaches based on social laws are still to be considered, but there is room to find another approach that would give better results. We also see that using no coordination gives a low number of hits per scenario. This is normal since each frigate engages every threat. However, as seen in Figure 6.36, there are many resources used when not coordinating.

Figure 6.39 presents the *efficiency* $((1 - \%\ of\ hits)/nb\ resources\ used)$ according to the utilisation of communications, while Figure 6.40 presents the efficiency of the

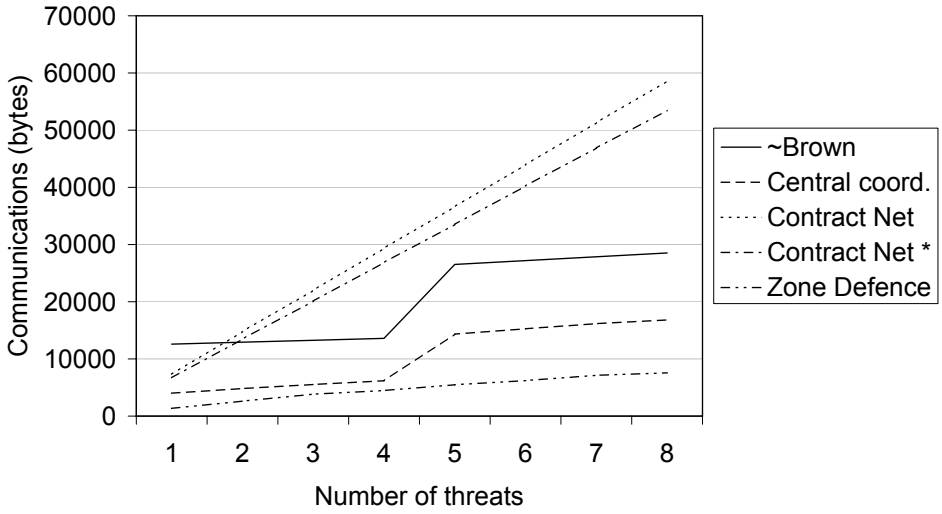

Figure 6.37: Total size of communications per scenario, considering different coordination mechanisms.

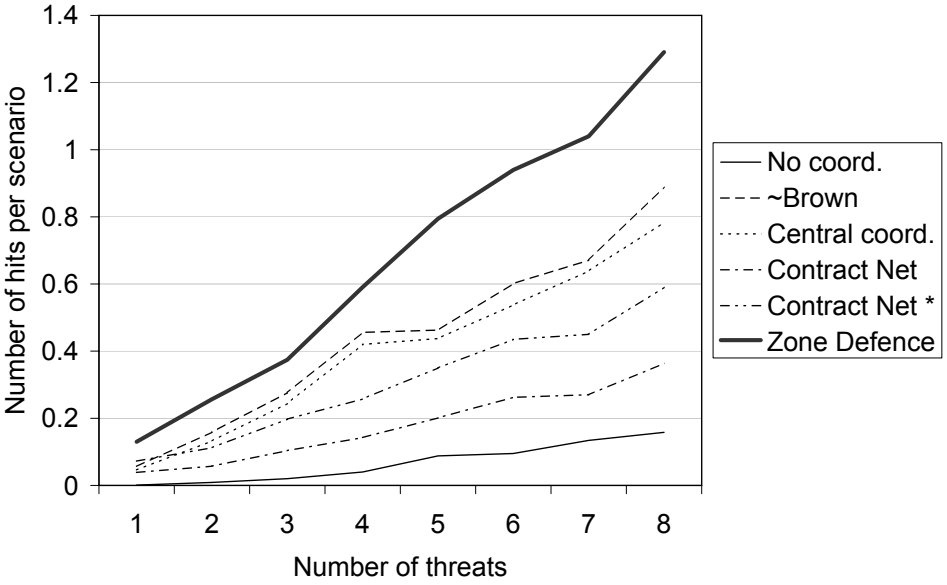

Figure 6.38: Number of hits per scenario, considering different coordination mechanisms.

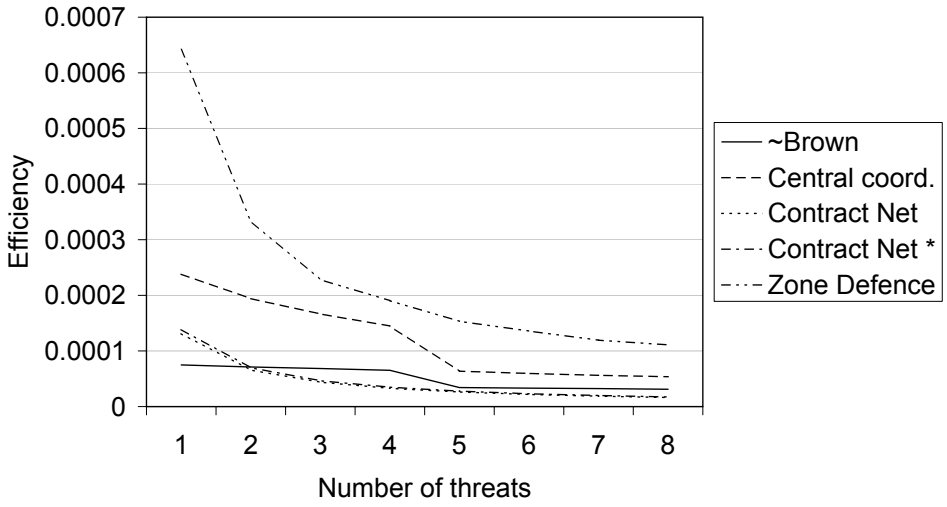

Figure 6.39: Efficiency relative to communications use, considering different coordination mechanisms.

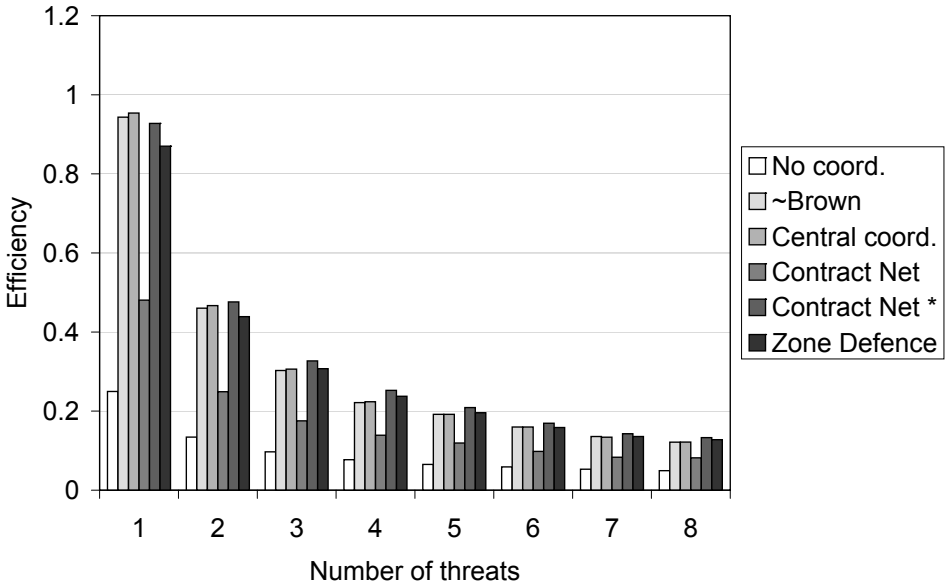

Figure 6.40: Efficiency relative to SAM use, considering different coordination mechanisms.

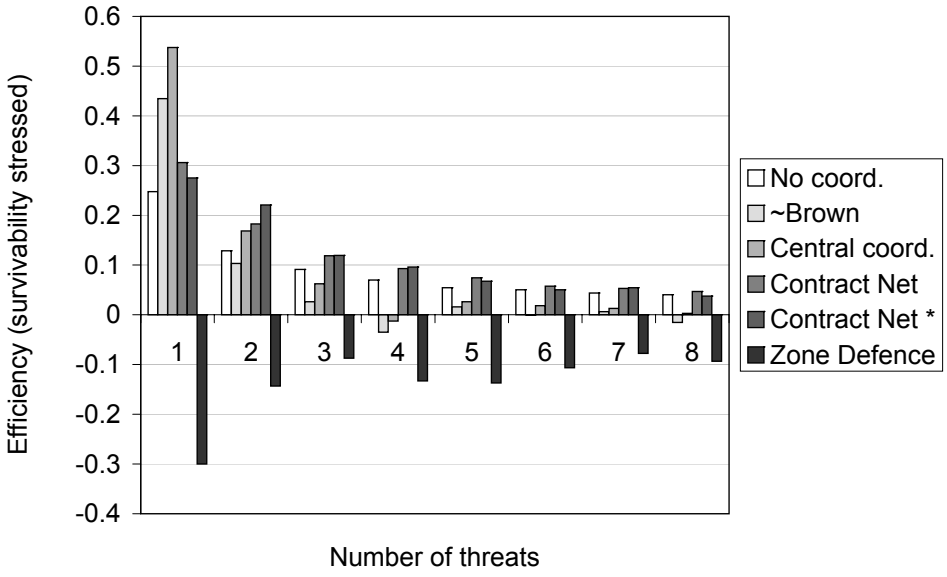

Figure 6.41: Efficiency (with survivability stressed) relative to SAM use, considering different coordination mechanisms.

mechanisms, according to the utilisation of SAMs. We see that in this case, the zone defence has a good efficiency, which can be explained by the fact that it does not use much communications. However, we should introduce an α parameter in the calculus of efficiency, as presented in Section 6.2.6. Indeed, these results are not biased enough toward the importance of survivability. Figure 6.41, which presents the efficiency (according to SAM use) with an α of 10 shows that overall, the Contract Net gives very good results. Thus, applying an α factor would allow correctly putting the emphasis on the survivability, which is our primary concern.

6.4 Discussion

In this chapter, we have seen different coordination mechanisms to address the specific problem of task distribution. We now have fewer conflicts by using task sharing. However, we have deliberately left open the question of *managing interactions*. What is the best way to avoid negative interactions while promoting positive interactions? A method based on social rules is a possible way to solve this problem. In fact, social laws coordination mechanisms can obtain very good results without as much communications as in communication-based approaches. Moreover, the navy already uses standard operating procedures, doctrines and rules of engagement. These are all social laws, with more or less importance via their firmness and sanctions that apply for breaking them.

Other open aspects are the evaluation of P_S. We presented in this chapter a heuristic to evaluate these probabilities of success. However, it is a very simple heuristic, and we have reasons to believe that is quite inaccurate. First, the evaluation of the first parameter, P_K, is an extrapolation that is probably too simple. Secondly, the parameter based on the CPA is overly simple, as it cannot appropriately model the complexity of the different systems on a typical frigate. A heuristic considering each system independently would probably be closer to the real probabilities of success and allow for better coordination.

Furthermore, there are still researches to be done on the use of backup plans. How could we introduce such safeguards in the planning? What would be the importance of backup plans? Incorporating safeguard actions in plans is time consuming and blocks resources that could have been used elsewhere. Is it a good idea to sacrifice actions in the constructed plan to keep some backup actions in case another agent fails? Another interest of backup is to plan actions to protect *ourselves*. In the case where a threat directed at us in engaged by agent-x, how far do we trust agent-x to be able to defend us? An agent may be cooperative but still fail in its tasks.

We showed in this chapter that the communications channels were not as used as we first expected them to be. Therefore, the communications seem not to be as problematic as we first imagined. However, communication-based mechanisms are especially hard to scale up. Will these mechanisms be as successful if we double the number of threats and frigates? On the other hand, it is usually easier to scale up mechanisms with social laws approaches; thus, we could develop new coordination mechanisms based on social laws.

Moreover, an interesting modification that could be done is on the calculus of the fleet priority. Currently, we consider the threat's target and the weight of this target. However, in real-life, there are various kinds of threats. Some missiles are far more sophisticated than others, and therefore more dangerous.

Finally, we have presented results where more than one frigate engages the same threat. The results were pointing that this behaviour is desired as it increases the survivability. Thus, still in a *task sharing* setting, we would simply assign the same task to more than one agent. However, there is work still to be done on the redundancy in engagements. There needs to be some way to prioritise the threats differently when constructing plans, so as to make sure that the threats are engaged uniformly (i.e., so that not every resource is pitted against the same portion of the threats, while other threats get engaged with only few actions).

Chapter 7

Conclusion and Future Work

In this thesis, planning and coordination with multiagent systems has been studied with the purpose of conceiving decision-support system (DSS) for resource management on Canadian frigates. We started by presenting the problem of managing resources in anti-air warfare. Then, we presented project NEREUS and its specifications.

We proceeded with the introduction of the theoretical basis for agents and multiagent systems. In multiagent systems, we detailed the interactions and defined multiagent coordination. We also presented some recent work on specific parts of the coordination problem.

Then, we presented different categories of real time systems. We illustrated the importance of contingencies and plan representations, and then detailed the importance of anytime algorithms in real-time systems.

We followed by presenting the simulator developed in the context of project NEREUS. This Naval Defence Simulator (NDS) is a naval environment to develop and validate methods to increase the survivability of frigates in anti-air warfare scenarios. Although it uses simplified frigate specifications, NDS is highly realistic and has been designed to be easily scaled up to new specifications.

We then presented past work realised at DAMAS on planning for efficient resource management in single frigate situations. We presented the *Bayesian movement*, developed by Plamondon [2003] and added another movement, the *radar cross section reduction movement* (RCSR movement), aimed at reducing the probability that a threat locks on the frigate. We also presented the *Tabu search* planning for resource management, accomplished by Soucy [2003].

Finally, we presented two general approaches to coordinate actions in anti-air warfare. We further detailed the *task sharing* approach and detailed four different mechanisms to distribute threat engagements among frigates: the *zone defence* coordination, *central coordination*, the *Contract Net* approach and the *∼Brown* coordination mechanism. Then, we analysed the impact of varying several parameters in these mechanisms. We concluded by comparing the developed mechanisms and by discussing some issues left open.

Parts of the work introduced in this thesis have been used in the following article: Plamondon et al. [2003].

7.0.1 Revision of Objectives

We will now recall the objectives defined in the introduction and see how they were addressed in this thesis.

- *Study strong temporal constraints and therefore real-time aspects, as they are reflected in our problem.* Chapter 3 has been dedicated to the study of real-time systems, real-time constraints and the implications of real-time in project NEREUS.
- *Develop a real-time naval combat simulator: the NDS.* In chapter 4, we presented the simulator that was developed in the context of project NEREUS. This simulator was designed to be used with the current specifications of project NEREUS and the new specifications for the future of the project.
- *Develop coordination mechanisms among several frigates.* Chapter 6 has been dedicated to the presentation and description of the coordination mechanisms developed in the context of project NEREUS. Four different task distribution mechanisms were presented: the *central coordination*, the *Contract Net* approach, the *∼Brown* coordination and the *zone defence* coordination. Of these mechanisms, the first three are based on communications, while the last one is based on social laws.
- *Integrate the developed coordination mechanisms in NDS.* We integrated the coordination mechanisms in NDS, as seen in Chapter 4. In chapters 4 and 6, we presented the different adjustable parameters we implemented while integrating these mechanisms in NDS.
- *Implement a tool to validate the conceived mechanisms with empiric testing.* In chapter 4, we presented the tools used to validate our mechanisms with empiric

testing. Different values for the coordination mechanism parameters were tested and presented in chapter 6. We also analysed and compared the mechanisms in Chapter 6.

7.1 Future Work

The development of decision-support systems for anti-air warfare is a complex task. Many good ideas still need to be explored for efficient resource management on Canadian frigates. We will now present some of the most interesting ideas for future work in project NEREUS.

7.1.1 Coordination Based on Learning

We presented three coordination types in Section 2.2.2. According to the taxonomy presented in Figure 2.2, we developed coordination mechanisms based on communication and social laws. However, coordination based on learning still needs to be explored. In real-life, coordination based on learning is often seen in small teams needing quick reactions, such as SWAT teams. Therefore, we believe that a coordination based on mutual learning should be explored in the future. Agents using such *SWAT coordination* mechanisms should predict the choices of other members of the team and select their actions to complement the team plan. In such mechanisms, the agents would learn their role in the team and the appropriate reactions according to various situations.

7.1.2 Metalevel Decision

We have seen different movement methods and coordination mechanisms. In the case of movement, there are some situations where RCSR movement is appropriate, while there are other scenarios where a Bayesian movement would be preferable. The problem is that we want to maximise the estimated survivability over the whole time interval between the detection of the first threat and the disappearance of the last one.

We have also seen that the coordination mechanisms proposed each have strengths and weaknesses. In the case where the communications are unavailable, the zone defence can be used with appreciable results. However, when there is enough time left and the

communications channels are fully functional, a Contract Net approach gives very good results. We see that different situations condition different responses.

Thus, we suggest the use of a metalevel decision agent to decide which movement method, planning algorithm and coordination mechanism to use, depending on the situation. This could be implemented using the meta-deliberation technique proposed by Dean and Boddy [1988].

7.1.3 Human in the Loop

An onboard decision-support system should be designed to be used by humans. It should present plans to operators before applying them so that operators can decide whether or not to put this plan into action. However, in the current project, the decided actions are started without consulting operators. Since the human needs time to evaluate a solution and decide to approve it, the deadlines would not be the same if the plans have to be approved by human operators. Furthermore, current planning algorithms do not take into account the possibility that actions can be refused by operators.

Moreover, having only one agent for each frigate does not allow the knowledge in multiagent systems at DAMAS to be used to its full extent. New agents could be designed that correspond to human operators and interact with the DSS agents. This would allow 1) exploring the interactions between human and agents in real-time systems, 2) improving current algorithms to take into account the choices of human operators, 3) modeling the complexity of negotiation and coordination between human operators in a hierarchical structure.

Appendix A

System Description

A.1 Typical Frigate

Much of the information about the exact nature of the specifications and capabilities of a Canadian frigate is classified. To avoid this issue and in order to maintain emphasis on the research interests and not be burdened the complexity and fidelity of the representation of hardkill and softkill systems, a considerably simplified model of the relevant Anti-Air Warfare (AAW) hardkill and weapons was used. This model is a simple, non-classified version of AAW hardkill and softkill for a typical frigate. Some of the details of the model are taken from Chaib-draa et al. [2001], while others were added at a later time.

The frigates in our researches share the following characteristics: 1) frigates are at rest (i.e. speed is null) and 2) frigate rotational speed is 3 °/seconds. For the purpose of this research, we assume the frigate turns on a single point.

A.1.1 Radar

Frigates possess a radar system, whose *range is 80 km.* For the purpose of this work, we assume everything in the range of the radar is detected, and correctly identified. This detection process is usually known as Situation and Threat Assessment (STA). The simulator was designed for it to be possible to introduce uncertainty in the detection and identification process. However, since the STA is not in the scope of our project, we will not bother with *how* the STA is done, but we will rather directly take the

normal outputs of this step as inputs from our sensors. It is also assumed that the information given by the radar is complete and sound (i.e., there is no uncertainty in object's position, bearing and angle of attack).

A.1.2 Illuminators

Closely linked to the hardkill systems, are two Separate Tracking and Illuminating Radar (STIR)s that are used to guide Surface-to-Air Missile (SAM)s to the threats, and to point the gun. This effectively provides two concurrent fire channels for the AAW hardkill weapons. The Close-in Weapon System (CIWS) has its own illuminator.

Separate Tracking and Illuminating Radar

The STIRs are radar used to guide hardkill weapon systems, more specifically, the SAMs and the gun. There are two distinct STIRs available, which each control only one SAM or the gun. Finally, a doctrine specifies that both STIRs will never be simultaneously assigned to the same threat.

- Search and lock time: 3 seconds.
- KA for SAM: 2.5 seconds.
- KA for Gun: 1 second.
- Blind zone of the first STIR: $\pm 120^\circ$ in azimuth, looking in the *forward* direction of the frigate.
- Blind zone of the second STIR: $\pm 120^\circ$ in azimuth, looking in the *backward* direction of the frigate.
- Range: 50 km.
- Once STIR lock is obtained, control can be instantly passed to the second STIR (only if not guiding an in-flight SAM).
- STIR must remain locked on target during total time-of-flight of SAM to target.

Close-in Weapon System Radar

The CIWS radar is used to control the CIWS firing at a threat.

- Search and lock time: 1 second.
- Blind zone: $\pm 15^\circ$ in azimuth, looking in the forward direction of the frigate; $70 - 90^\circ$ in polar angle (where 90° is vertical).
- Range: 5.5 km.
- KA: 1 second.

A.1.3 Hardkill Systems

The AAW hardkill weapons are directed to intercept a threat and actively destroy it through direct impact or explosive detonation in the proximity of the threat. Effectiveness of these weapons depends on a variety of factors: distance, type and speed of the threat, environment, etc. The AAW hardkill weapons for a typical frigate include SAMs, which are long-range interception missiles, an intermediate range gun, and a CIWS that is a short-range, rapid-fire gun.

Finally, neither a SAM nor the gun will engage or re-engage a threat until Kill Assessment (KA) of a prior engagement has been completed.

Surface-to-Air Missile

The SAMs, as the name implies, are missiles fired to intercept incoming air threats. The specifications are as follow:

- Inventory: 16 missiles (assume all are initially loaded).
- Blind zone: none.
- Minimum range: 2.2 km.
- Maximum range: 20 km.

- Speed: 306 m/s.
- Probability of kill: 75 %.
- A SAM can be fired only after a STIR has locked on the target.
- There is no delay between the time the fire order is issued and the missile is launched.
- SAMs travel on a straight-line trajectory.

Gun

The gun is a long-range canon, which can fire rounds up to a fire rate of 200 rounds/minute. The specifications are as follow:

- Total inventory: 750 rounds (5 loads).
- Load capacity: 150 rounds (10 magazines).
- Magazine capacity: 15 rounds.
- Gun capacity: Two magazines at a time.
- Blind zone: $\pm 35^\circ$ in azimuth, looking in the backward direction of the frigate.
- Minimum range: 0.9 km.
- Maximum range: 5 km.
- Speed: 850 m/s.
- Probability of kill: 4 %/round.
- Fire rate: 200 rounds/minute.
- Standard salvo division: 5 rounds/salvo (Consecutive salvos can be fired with no delay between them).
- Magazine reload time: 5 seconds to reload two magazines (30 rounds). Can be reloaded anytime there are 7 rounds or fewer remaining in the loaded magazines.
- Load reload time: 8 minutes to reload a complete load (150 rounds).
- The gun can be fired only after a STIR has locked on the target.

- The slew time to move the gun into fire position is 0 seconds.
- There is no delay between the time the fire order is issued and the gun starts shooting.

Close-in Weapon System

The CIWS is a short-range gun. It can fire high-velocity rounds up to a fire rate of 3,300 rounds/minute. The specifications are as follow:

- Inventory: 1,500 rounds.
- Blind zone: $\pm 15^\circ$ in azimuth, looking in the forward direction of the frigate; $70^\circ - 90^\circ$ in polar angle (where 90° is vertical).
- Minimum range: none.
- Maximum range: 2.5 km.
- Speed: 1,200 m/s.
- Probability of kill: 0.6 %/round.
- Fire rate: 55 rounds/s.
- The CIWS can be fired only after the CIWS radar has locked on the target.
- Slew time to move the CIWS into position to fire at the target is 0 second.
- There is no delay between the time the fire order is issued and the CIWS starts shooting.

A.1.4 Softkill Systems

The AAW softkill weapons use techniques to deceive or disorient a threat to cause its self-destruction, or at least lose its fix on its intended target. The softkill resources consist of chaff shells and jamming systems.

Jamming Systems

The jamming systems use electromagnetic emissions to confuse the threat's sensors, causing the threat either to lose its fix on its intended target, or to improperly assess the position of its target.

There are two steps when using a jamming system to disable an incoming threat:

1. *Break missile lock on the frigate:* This lasts 20 seconds and consists of the jamming system searching for and acquiring the threat, and then processing to cause the missile to break its radar lock on the frigate.
2. *Create a false target position on the missile's radar:* The jamming system is used to create a delayed offset from a normal radar reflection, and is interpreted by the missile's radar as the actual target position. Because of this offset, the range determined by the radar of the threat is greater than the actual range of the target. Once the jamming antenna has broken the lock of the missile, creating the false position takes 3 seconds.

The specifications of the jamming systems are as follow:

- Blind zone of the first antenna: $\pm 100^\circ$ in azimuth, looking at the left of the frigate.
- Blind zone of the second antenna: $\pm 100^\circ$ in azimuth, looking at the right of the frigate.
- Range: 24 km.
- Probability of success (jamming only): 40 %.
- Probability of success (jamming and chaff): 80 %.
- Each jamming system can deal with up to two threats at a time.
- Jamming responsibility for a threat can be passed from one jamming antenna to the other.

Chaff

The chaff system launches a shell that produces a burst at a designated position. The resultant chaff cloud consists of aluminised glass elements and is conceived to screen

the frigate or produce an alternate target on which a radar-guided threat can fix. It is designed to offer a large Radar Cross Section (RCS)[1] and therefore confuse the guidance system of the incoming threat. The specifications are as follow:

- Inventory: 30 shells.
- Range: 250 m.
- Duration: 10 minutes.
- Probability of success (chaff only): 30 %.
- Probability of success (jamming and chaff): 80 %.
- Shape: Sphere of 40 m radius.

To simplify the model, we did not consider gradual degradation of the chaff cloud, neither did we consider meteorological factors. The chaff is active to its full potential for a total of 10 minutes and then becomes completely inefficient.

A.1.5 Sectors

It is useful to summarise the information of the resources described in Sections A.1.2, A.1.3 and A.1.4 and their specifications. Tables A.1 and A.2 sum up the specifications of hardkill systems, while Table A.3 wraps the details of softkill systems.

	Inventory	*Coverage* (°)	P_{kill} (%)
SAM	16	0-360	75
Gun	750	0-145, 215-360	4/rnd
CIWS	1,500	15-360	0.6/rnd

	Range min/max (km)	*Speed* (m/s)	*Firerate* (rounds/min.)	*KA Duration* (seconds)
SAM	2.2/20	306	-	2.5
Gun	0.9/5	850	200	1
CIWS	-/2.5	1,200	3,300	1

Table A.1: Frigate hardkill weapons specifications.

[1]The Radar Cross Section is the apparent size of an object, caused by the extent to which an object reflects radar pulses.

	Coverage (°)	*Range* (km)	*Lock Duration* (s)
Front STIR	0-120, 240-360	50	3
Rear STIR	60-300	50	3
CIWS Illuminator	15-360	5.5	1

Table A.2: Frigate hardkill illuminators specifications.

	Range (km)	*Coverage* (°)
Left Jammer	24	0-190, 350-360
Right Jammer	24	0-10, 170-360
Chaff	0.25	0-360

Table A.3: Frigate softkill systems specifications.

If we take a look at the resources, we realise it is possible to divide the environment surrounding the frigate in sectors. Thus, we find there are only 12 distinct angular sectors, depicted in Figure A.1.

Table A.4 describes these twelve distinct sectors, showing the angular coverage of a sector and the differences in the weapon engagement capabilities of a sector compared to the "normal" state, which includes:

- One available STIR.
- Availability of the gun.
- Availability of the CIWS.
- Availability of a jamming antenna.
- Possibility to launch a chaff.

Sectors	*Coverage* (°)	*Differences from normal state*
1, 7	60-120, 240-300	One additional STIR
2, 6, 8, 12	15-60, 120-145, 215-240, 300-345	No difference
3, 5	10-15, 345-350	No CIWS
4	0-10, 350-360	No CIWS, but two jammers instead of one
9, 11	145-170, 190-215	No Gun
10	170-190	No Gun, but two jammers instead of one

Table A.4: Sectors for hardkill and softkill systems.

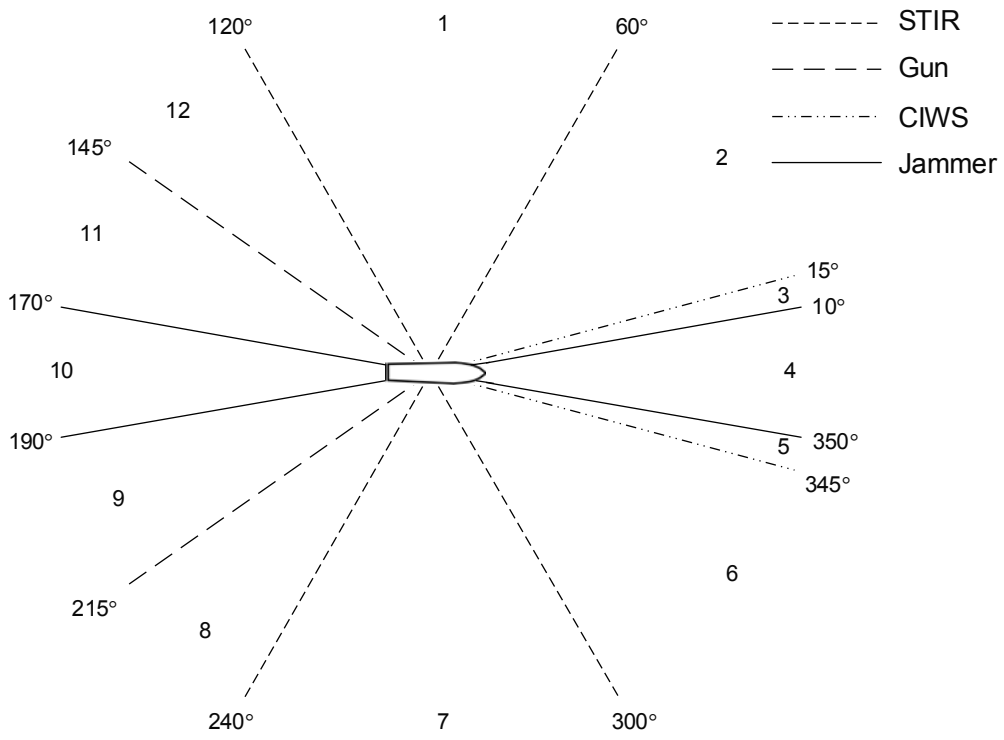

Figure A.1: Bayesian sectors.

A.2 Coordination Layouts

Obviously, the efficiency of the different mechanisms is conditioned by the layout in which the frigates are positioned relative to each other. In this research, 9 different formations were tested and compared. Shown in Figure A.2, layouts 1 to 8 were designed especially for scenarios with 4 frigates and 1 cargo vessel[2]. Layout 9 is an example of a generic formation for any other number of frigates. In the formations, the cargo vessel is at the centre, while the frigates are positioned uniformly around it. In Figure A.2, cargo ships are identified by the letter C to distinguish them from frigates. The d represents the typical inter-ship distance, while n (in layout 9 only) is the number of frigates. Both of these parameters can be decided before starting the tests. Finally, while the distances between ships are scaled, the ships have been enlarged to get a better view.

[2]Basically, a *cargo vessel* is a ship without any defence systems.

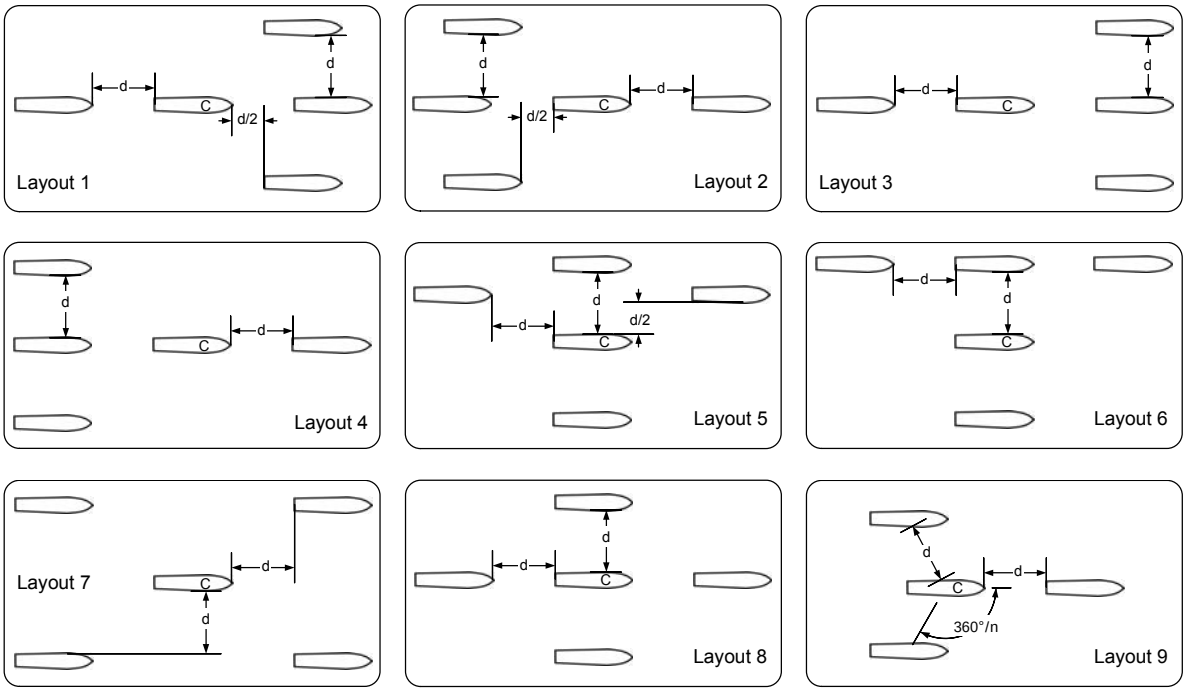

Figure A.2: Coordination layouts

A.3 Threats

ASM, also referenced without distinction as *Anti-Ship Missile* or *threat*, represent Anti-Ship Missile threats coming toward the frigate. At this stage of the project, only one type of threat is used. However, it has been planned to introduce diversity in incoming threats in the near future of the project. The threats have the following specifications:

- Threats travel in a straight line.
- Speed: 850 m/second.
- Threats are generated randomly in a range between 5 and 80 km.
- Threats are generated randomly at an altitude between 1 km and 20 km.
- Threats are generated randomly at any azimuthal angle around the frigates.
- Probability of kill if a threat reaches the the frigate: 50 %.

Appendix B

Intercepting Threats

When intercepting incoming threats, there are two major concerns; we need to know *when* and *where* to fire our weapons. The following sections will explain how we find the solution to these two problems. In this appendix, we will consider that the weapon used is ready to fire at the state we deliberate on. This assumption can be made because we can virtually bring *forward* the current state to a new state where the lock on the object has been obtained and the weapon is really ready to fire. This is possible because it is easy to extrapolate the future positions of the objects knowing their speed and current position, since we use rectilinear movement in our simulator.

B.1 Closest Point of Approach Equals Zero

The CPA of a threat is evaluated for a specific position. It is the shortest possible distance between the trajectory described by a threat and the evaluation point. When the CPA equals zero, meaning the threat is coming right at us, the matter of where to fire is easy. Because we use rectilinear movement, we only need to fire directly toward the threat. The important matter in this case is to know *when* to fire. We need to assume two facts in order to determine the fire time.

1. The distance we want to intercept the threat at is within the weapon *range of action*.
2. We have enough time to fire and intercept the threat at the determined distance.

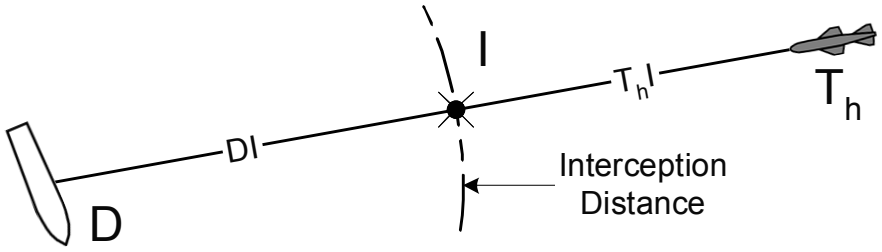

Figure B.1: Interception point (CPA = 0).

$D =$ Position of the defender
$I =$ The interception point
$T_h =$ Position of the threat
$DI =$ Distance between the defender and the interception point
$T_hI =$ Distance between the threat and the interception point
$S_d =$ Speed of the defender's weapon
$S_t =$ Speed of the threat
$\Delta w =$ Time to wait before firing

The Figure B.1 shows the variables used in the computation of Δw. DI and T_hI can be easily computed since we decide ourselves the interception distance. Using the details of Figure B.1, the time of interception can be expressed as follows:

$$\Delta w + \frac{DI}{s_d} = \frac{T_hI}{s_t} \tag{B.1}$$

In other words, the time to wait before firing is equal to the time taken by the threat to reach that point, minus the time taken by our weapon to reach the same point. Therefore, the function $FINDΔw$ is a method taking D, T_h and I in input and returning the Δw.

If assumption 2 is not valid, we will observe a negative Δw, meaning that this weapon cannot intercept the said target at this particular distance.

B.2 Closest Point of Approach Different Than Zero

The case where the CPA is equals to zero is simple enough. However, when the CPA is *not* zero, the calculus becomes a little more extensive. Intercepting at the closest point means intercepting at the minimum range of the current weapon, while interception at the farthest point means either we intercept at maximum range or the soonest possible.

In both case, we need to determine not just the time of fire, but also the direction of fire. We will see how this is obtained. Algorithms B.1 and B.2 find the distance to the interception point. Once we have the distance from the threat to the interception point, we just need to use the function *POS-BETWEEN* to find the interception point. *POS-BETWEEN* returns a position between two positions, given the ratio of this new position on the line between position A and B. In this case, the positions are the threat and threat's target positions, while the ratio is (*time to interception point*/*time to target*). When we have this interception point and the position of the defender, getting the direction and time of fire is trivial.

B.2.1 Finding the Closest Interception Point

We need to assume that in this case, the CPA is at least smaller than the maximum range of the weapon used. Algorithm B.1 shows how to find the distance from the threat to the closest interception point, roughly corresponding to intercepting at the latest possible time.

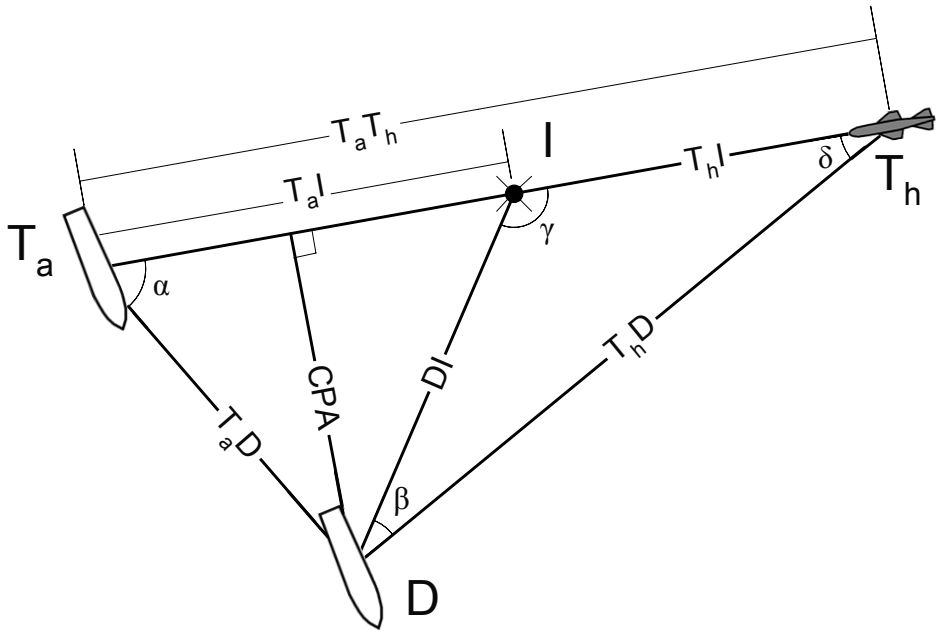

Figure B.2: Closest interception point (CPA $\neq 0$).

Figure B.2 shows the variables to consider in the calculus.

The following is known:

$T_a =$ Position of the target
$T_h =$ Position of the threat
$D =$ Position of the defender

$T_aT_h =$ Distance between the target and the threat
$T_aD =$ Distance between the target and the defender
$T_hD =$ Distance between the threat and the defender

$\alpha =$ Angle formed by $\overline{T_aT_h}$ and $\overline{T_aD}$
$\beta =$ Angle formed by $\overline{DI}$ and $\overline{T_hD}$
$\gamma =$ Angle formed by $\overline{DI}$ and $\overline{T_hI}$
$\delta =$ Angle formed by $\overline{T_hI}$ and $\overline{T_hD}$

$S_d =$ Speed of the defender's weapon
$S_t =$ Speed of the threat

$R_{min} =$ Minimum efficient range of the weapon used
$R_{max} =$ Maximum efficient range of the weapon used

$SafeDist =$ Range of a zone surrounding any ship in which we do not shoot
$CPA =$ Minimum possible distance between the defender and the threat

In addition, the following is unknown:

$I =$ The interception point

$T_aI =$ Distance between the target and the interception point
$T_hI =$ Distance between the threat and the interception point
$DI =$ Distance between the defender and the interception point

$\Delta t =$ Time elapsed between the current situation and the impact time
$\Delta w =$ Time to wait before firing

To simplify the algorithm, which is already complex, we will not show how to get what is known, nor what is passed in parameters. It is sufficient to know that we can easily obtain these variables.

Algorithm B.1 Dist-Closest() **returns** the distance to the closest interception point

$T_hI \leftarrow$ -1
if $CPA > R_{min}$ **then**
  if $CPA = T_aD$ **then**
    $T_hI \leftarrow T_aT_h - SafeDist$
  else
    $T_hI \leftarrow T_aT_h - \sqrt{T_aD^2 - CPA^2}$
else
  $\beta \leftarrow \gamma - \delta$
  if $\beta = 0$ **then**
    $T_hI \leftarrow T_aT_h - R_{min}$ {The target is also the defender}
  else
    $T_hI \leftarrow R_{min} \cdot \sin(\beta)/\sin(\delta)$
    $I \leftarrow$ POS-BETWEEN$(T_h, T_a, (T_hI/T_aT_h))$ {see Section B.2}
    $\Delta w \leftarrow$FIND$\Delta w(D, T_h, I)$ {see Section B.1}
    if $\Delta w < 0$ **or** $T_hI > T_aT_h$ **then**
      $\beta \leftarrow (180 - \gamma) - \delta$
      $T_hI \leftarrow R_{min} \cdot \sin(\beta)/\sin(\delta)$
      $I \leftarrow$ POS-BETWEEN$(T_h, T_a, (T_hI/T_aT_h))$
      $\Delta w \leftarrow$ FIND$\Delta w(D, T_h, I)$
      if $\Delta w < 0$ **or** $T_hI > T_aT_h$ **then**
        $T_hI \leftarrow$ -1
return T_hI

B.2.2 Finding the Farthest Interception Point

Algorithm B.2 shows how to find the distance from the threat to the farthest interception point. In this case, if we cannot intercept at the maximum range of the weapon, we intercept at the first possible moment. The variables used in this algorithm are those defined in Section B.2.1, and shown in Figure B.2.

B.3 Finding the Δt

When we want to intercept a threat at the first possible time, assuming that we shoot right now, we need to find *when* the interception will occur. Once again, we use the variables defined in the Section B.2.1.

Algorithm B.2 Dist-Farthest() **returns** the distance to the farthest interception point

$T_hI \leftarrow$ -1
$\beta \leftarrow \gamma - \delta$
if $\beta = 0$ **then**
  $T_hI \leftarrow T_aT_h - R_{max}$
else
  $T_hI \leftarrow R_{max} \cdot \sin(\beta)/\sin(\delta)$
  $I \leftarrow$ POS-BETWEEN$(T_h, T_a, (T_hI/T_aT_h))$
  $\Delta w \leftarrow$ FIND$\Delta w(D, T_h, I)$
  if $\Delta w < 0$ **or** $T_hI > T_aT_h$ **then**
    {We cannot intercept at R_{max}, so we intercept at the first possible time}
    FINDΔt {see Section B.3.}
    $T_hI \leftarrow \Delta t \cdot S_t$
return T_hI

It is then clear that:

$$T_hI = S_t \cdot \Delta t \tag{B.2}$$

$$DI = S_d \cdot \Delta t \tag{B.3}$$

$$T_aI = T_aT_h - T_hI \tag{B.4}$$

With the law of cosines:

$$DI^2 = T_aI^2 + T_aD^2 - 2 \cdot T_aI \cdot T_aD \cdot \cos\alpha = {S_d}^2 \cdot \Delta t^2$$

Substituting T_hI, DI and T_aI with result from Equation B.2, B.3 and B.4, we get:

$${S_d}^2 \cdot \Delta t^2 = (T_aT_h - S_t \cdot \Delta t)^2 + T_aD^2 - 2 \cdot (T_aT_h - S_t \cdot \Delta t) \cdot T_aD \cdot \cos\alpha$$

If we develop:

$$T_aT_h^2 + S_t^2 \cdot \Delta t^2 - 2 \cdot T_aT_h \cdot S_t \cdot \Delta t + T_aD^2 - 2 \cdot T_aT_h \cdot T_aD \cdot \cos\alpha + 2S_t \cdot T_aD \cdot \cos\alpha \cdot \Delta t - S_d^2 \cdot \Delta t^2 = 0$$

By grouping the terms on Δt:

$$(S_t^2 - S_d^2) \cdot \Delta t^2 + (2S_t \cdot T_aD \cdot \cos\alpha - 2 \cdot T_aT_h \cdot S_t) \cdot \Delta t + (T_aT_h^2 + T_aD^2 - 2 \cdot T_aT_h \cdot T_aD \cdot \cos\alpha) = 0$$

Since we have a quadratic equation where the only unknown variable is Δt, we can resolve it by finding the roots:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where:

$$\begin{aligned} a = &\quad (S_t^2 - S_d^2) \\ b = &\quad (2S_t \cdot T_a D \cdot \cos\alpha - 2 \cdot T_a T_h \cdot S_t) \\ c = &\quad (T_a T_h^2 + T_a D^2 - 2 \cdot T_a T_h \cdot T_a D \cdot \cos\alpha) \end{aligned}$$

At first, we can note that, when a equals 0, we have a special case and only need to solve the following equation:

$$(2S_t \cdot T_a D \cdot \cos\alpha - 2 \cdot T_a T_h \cdot S_t) \cdot \Delta t + (T_a T_h^2 + T_a D^2 - 2 \cdot T_a T_h \cdot T_a D \cdot \cos\alpha) = 0$$

Or, isolating Δt:

$$\Delta t = -\frac{(T_a T_h^2 + T_a D^2 - 2 \cdot T_a T_h \cdot T_a D \cdot \cos\alpha)}{(2S_t \cdot T_a D \cdot \cos\alpha - 2 \cdot T_a T_h \cdot S_t)}$$

On a final note, it is important to say that we should return the *lowest* Δt found from the two roots, since we want to find the *first* interception point. We could still use the second Δt if we find conflict in the plan using the first Δt.

Bibliography

Stanislaw Ambroszkiewicz, Olaf Matyja, and Wojciech Penczek. Team formation by self-interested mobile agents. In *DAI*, pages 1–15, 1998.

David W. Ash and Vlad G. Dabija. *Planning for Real Time Event Response Management*. Prentice-Hall, 2000.

Mihai Barbuceanu, Tom Gray, and Serge Mankovski. Coordinating with obligations. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 62–69, New York, 9–13, 1998. ACM Press. ISBN 0-89791-983-1.

Dale Blodgett, Sébastien Paquet, Pierrick Plamondon, Brahim Chaib-draa, and Peter Kropf. Coordinating plans for agents performing AAW hardkill and softkill for frigates. In *Proceedings of The 2001 AAAI Fall Symposium Series*, pages 1–8, North Falmouth, MA, July 2001.

Dale Blodgett, Pierrick Plamondon, Brahim Chaib-draa, Peter Kropf, and Eloi Bossé. A method to optimize ship maneuvers for the coordination of hardkill and softkill weapons within a frigate. In *7^{th} International Command and Control Research and Technology Symposium (7^{th}ICCRTS)*, Québec, QC, September 2002.

Alan H. Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988. ISBN 093461363X.

Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.

John R. Boyd. Organic design for command and control. Briefing slides, May 1987.

John R. Boyd. The essence of winning and losing. Briefing slides, January 1996.

Will Briggs and Diane Cook. Flexible social laws. In Chris Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 688–693, San Francisco, 1995. Morgan Kaufmann.

Chris Brown, Peter Fagan, Angela Hepplewhite, Bob Irving, Dick Lane, and Emma Squire. Real-time decision support for the anti-air warfare commander. In *6^{th} International Command and Control Research and Technology Symposium (6^{th}ICCRTS)*, U.S. Naval Academy, Annapolis, Maryland, June 2001.

Chris Brown and Dick Lane. Anti-air warfare co-ordination - an algorithmic approach. In *Command and Control Research and Technology Symposium*, Monterey, CA, USA, October 2000.

Brahim Chaib-Draa, Peter Kropf, and Sébastien Paquet. A teamwork test-bed for a decision support system. In *Proceedings of EUROSIM 2001: Shaping Future with Simulation*, Delft, The Netherlands.

Brahim Chaib-draa, Sébastien Paquet, and Pierrick Plamondon. Proposal #222802-98: Resource management in complex sociotechnical systems: Report on first stage. Technical report, Laval University, Qc, 2001.

Bruce A. Chalmers and P. da Ponte. Algorithms for naval resource planning: Overview and preliminary assessment. Technical report drev-r-9420, Defence Research Establishment Valcartier, Valcartier, Québec, Canada, Qc, 1995.

P. Cohen, H. Levesque, and I. Smith. *On Team Formation*. Kluwer Academic Publishers, 1997.

T.L. Dean and M. Boddy. An analysis of time-dependant planning. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88)*, pages 49–54, Menlo Park, California, 1988. AAAI Press.

Frank Dignum and Mark Greaves, editors. *Issues in Agent Communication: An Introduction*, volume 1916 of *Lecture Notes in Computer Science*, 2000. Springer. ISBN 3-540-41144-5.

Edmund H. Durfee and Victor R. Lesser. Negotiating task decomposition and allocation using partial global planning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 229–243. Morgan Kaufmann, San Mateo, CA, 1989. ISBN 0-273-08810-6.

Edmund H. Durfee and Victor R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, - 1991.

Cora Beatriz Excelente Toledo. *The Dynamic Selection Of Coordination Mechanisms In Multiagent Systems*. PhD thesis, University Of Southampton, Department of Electronics and Computer Science, Faculty of Engineering and Applied Science, February 2003.

L. Festinger. *A Theory of Cognitive Dissonance*. CA : Stanford University Press, 1957.

Donald Gillies. Faq, March 2004.

S. Hallé, B. Chaib-draa, and J. Laumonier. Car platoons simulated as a multiagent system. In *Proceedings of Agent Based Simulation 4 (ABS4)*, March 2003a.

S. Hallé, F. Gilbert, and B. Chaib-draa. Architectures for collaborative driving vehicles: From a review to a proposal. Technical report, Université Laval, Ste-Foy, Québec, 2003b.

Eric A. Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1–2):139–157, 2001.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

Minghua He, Nicholas R. Jennings, and Ho-Fung Leung. On agent-mediated electronic commerce. *EEE Transactions on Knowledge and Data Engineering*, 15(4):985–1003, 2003.

Martin Hiller. Software fault-tolerance techniques from a real-time systems point of view - an overview. Technical report, 1998.

Bernardo A. Huberman and Tad Hogg. Distributed computation as an economic system. *Journal of Economic Perspectives*, 9(1):141–152, 1995.

Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. chapter 1, pages 1–23. Morgan Kaufmann, San Francisco, 1998.

Michael N. Huhns and Larry M. Stephens. Multiagent systems and societies of agents. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 79–120. The MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-23203-0.

Nicholas R. Jennings. Coordination techniques for distributed artificial intelligence. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 187–210. John Wiley & Sons, 1996.

Nicholas R. Jennings. An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41, 2001. ISSN 0001-0782.

Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

Nicholas R. Jennings and Michael Wooldridge. Applications of intelligent agents. pages 3–28, 1998.

Peter G. Kropf, Brahim Chaib-draa, and Bruce A. Chalmers. Resource management in socio-technical systems: A multi-agent coordination framework. In *HMS 2000*, page pp. 6570, Portofino, Italy, October 2000.

Marc-André Labrie. Langage de communication agent basé sur les engagements par l'entremise des jeux de dialogue. Master's thesis, Computer Science & Software Engineering Department, Laval University, 2004.

Thé Liang. Getting the act together: Hardkill-softkill coordination for littoral waters. In *International Defense Review 8*, pages 54–56, Netherland, August 1995.

Thé Liang and K.D. Liem. Integrated naval air defence: co-ordinating hardkill and softkill weapons. In *International Defense Review 6*, pages 567–569, Netherland, 1992.

Micha F. Lindemans. Nereus, March 2004.

Sylvain Lizotte. Coordination dans les sociétés d'agents: une approche basée sur la gestion des dépendances. Master's thesis, Computer Science & Software Engineering Department, Laval University, 1996.

S. P. Lloyd and H. S. Witsenhausen. Weapons allocation is np-complete. *Proceedings of the 1986 Summer Computer Simulation Conference*, 1986.

Thomas W. Malone. Modeling coordination in organization and markets. In *Management Science*, volume 33, pages 1317–1331, October 1987.

Nicolas Maudet and Brahim Chaib-draa. Commitment-based and dialogue-game based protocols: new trends in agent communication languages. *The Knowledge Engineering Review*, 17(2):157–179, June 2002.

Jean-François Morissette and Brahim Chaib-draa. An agent-based decision support system for naval anti-air warfare: An exploration based on simulation. Submitted to *The 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, March 2004.

Jean-François Morissette, Brahim Chaib-draa, and Pierrick Plamondon. Resource allocation in time-constrained environments: The case of frigate positioning in anti-air

warfare. Submitted to *Modelling, Computation and Optimization in Information Systems and Management Sciences*, March 2004.

Thierry Moyaux, Brahim Chaib-draa, and Sophie D'Amours. Multi-agent coordination based on tokens: Reduction of the bullwhip effect in a forest supply chain. In *Proceedings of the 2nd int. joint conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, Melbourne (Victoria, Australia), 2003.

Thierry Moyaux, Brahim Chaib-draa, and Sophie D'Amours. Multi-agent simulation of collaborative strategies in a supply chain. In *Proceedings of the 3rd int. joint conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, New-York (New-York, USA), 2004.

Ranjit Nair, Milind Tambe, and Stacy Marsella. Role allocation and reallocation in multiagent teams: towards a practical analysis. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 552–559. ACM Press, 2003. ISBN 1-58113-683-8.

James Odell, H. Van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in uml. In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 121–140. Springer-Verlag New York, Inc., 2001. ISBN 3-540-41594-7.

Sébastien Paquet. Coordination de plans d'agents: Application à la gestion des ressources d'une frégate. Master's thesis, Computer Science & Software Engineering Department, Laval University, 2001a.

Sébastien Paquet. Document calculs des distances. Technical report, DAMAS, September 2001b.

Philippe Pasquier and Brahim Chaib-draa. The cognitive coherence approach for agent communication pragmatic. In Jeffrey S. Rosenschein, Tuomas Sandholm, Michael Wooldridge, and Makoto Yokoo, editors, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'03)*, pages 544–551. ACM Press, 2003. 14–18 July 2003, Melbourne, Australia.

Pierrick Plamondon. A frigate survival approach based on real-time multiagent planning. Master's thesis, Computer Science & Software Engineering Department, Laval University, 2003.

Pierrick Plamondon, Brahim Chaib-draa, Patrick Beaumont, and Dale Blodgett. A frigate movement survival agent-based approach. In V. Palade, R.J. Howlett, and L.C. Jain, editors, *Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'2003)*, Lecture Notes in

Artificial Intelligence. Springer, 2003. 3-5 september 2003, University of Oxford, United Kingdom.

Publius Ovidius Naso. *Metamorphoses*. Indiana University Press, December 1955.

David V. Pynadath and Milind Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems*, 7:71–100, 2003.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (second edition)*. Prentice-Hall, Englewood Cliffs, NJ, 2003.

Tuomas Sandholm, Sandeep Sikka, and Samphel Norden. Algorithms for optimizing leveled commitment contracts. In *IJCAI*, pages 535–541, 1999.

Mary Wollstonecraft Shelley. *Frankenstein, or The Modern Prometheus*. 1818.

Yoav Shoham. Agent oriented programming. Technical Report Technical Report STAN-CS-1335-90, Computer Science Department, Stanford University, Stanford, CA 94305, 1990.

Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 276–281, San Jose, California, 1992. AAAI, AAAI Press.

Munindar P. Singh. A social semantics for agent communication languages. In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, pages 31–45. Springer-Verlag: Heidelberg, Germany, 2000.

Reid G. Smith. The contract net protocol. high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):61–70, 1980.

Martin Soucy. Planification déliberative en temp réel. Master's thesis, Computer Science & Software Engineering Department, Laval University, 2003.

M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7: 83–124, 1997.

The American Heritage Dictionary of the English Language: Fourth Edition. Deadline, March 2004.

The Foundation for Intelligent Physical Agents. Fipa acl message structure specification. Component 00061, FIPA, December 2002b.

The Foundation for Intelligent Physical Agents. Fipa contract net interaction protocol specification. Component 00029, FIPA, December 2002a. Interaction Protocols.

Venu Vasudevan. Comparing agent communication languages. Objs technical note, Object Services and Consulting, Inc., 1998.

E.L. Waltz and D.M. Buede. Data fusion and decision support for command and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(6):865–879, 1986.

Gerhard Weiss. Prologue. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 1–23. The MIT Press, Cambridge, MA, 1999. ISBN 0-262-23203-0.

Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

Michael Wooldridge. Agent-based software engineering. *IEE Proceedings Software Engineering*, 144(1):26–37, 1997.

Michael Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, Inc., Chichester, England, February 2002. ISBN 0 47149691X.

Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 35:304–338, 2001.