MAS for Ship Damage Control Decision Process

Paulo T. Martins and Eugénio Oliveira
Artificial Intelligence and Computer Science Laboratory, University of Porto
Porto, 41200-465, Portugal
(deg11010, eco)@fe.up.pt

Abstract

This paper introduces a Multi-Agent System (MAS) for solving the crew resources distribution problem when a ship suffers several damages and priorities must be assigned in order for it to survive. In our approach, the ship is the MAS environment and the attacker, equipment, crew and officers (the decision makers) are represented through agents. Decisions on resources assignment are taken after a negotiation process and using a kind of weighted utility-based selection process. We have followed a systematic Agent Oriented Software Engineering approach, called PORTO, to specify the implemented system. The final product is a method to evaluate new platforms survivability independently of crew effectiveness, which can also be used to find the best response procedure to deal with a damage event on board the platform, which than may be used for training of new crews.

keywords: multi-agent system, agent oriented software engineering, application.

1 Introduction

A ship is a complex system, where the crew interacts with equipment and auxiliary systems, within a closed space which is subjected to internal and external factors, such as environmental conditions, damage originated by equipment malfunction (e.g. fire or flooding on board), or by external agents (e.g. an aircraft launching a missile towards the ship; or another ship that collides with it). Either way, crew on board must act to prevent the total loss of the ship. This is a subject that must not be neglected during the ship design process, while defining its complement (crew numbers) and the allocation of compartment and equipment. Crew effectiveness when dealing with damage is dependent upon crew element numbers, crew technological knowledge and upon ships arrangement (compartments and equipment allocation within the physical boundaries of the ship). This work deals with the first problem, how to decide what to do in case of damage, how to prioritize when several damages occur at the same time, and how to allocate different crew elements taking account their ability to perform the required tasks. The other problems have already been addressed [6]. We are dealing with a multi-criteria decision problem on how to distribute limited resources without previously knowing all possible alternatives. Therefore, since alternatives are not known in advance, classical multi-attribute methods cannot be used (e.g. trade-off analysis), and multi-objective generation methods (e.g. multi-objective genetic algorithms) would probably require a considerable amount of time to find viable solutions and would provide a set of different pareto efficient solutions instead of a single one. Moreover, none of these techniques would be able to represent the natural interaction between the different actors in the decision process. Multi Agents Systems (MAS), on the other hand, provides a natural way of representing the different actors by autonomous computing entities (agents) that perform different tasks (actions), within an environment (ship). Through interacting and, possibly cooperating with each other, the agents will be able to change the state of the environment with an overall objective: ship survivability. The MAS architecture will include different kinds of agents ranging from simple ones, reactive agents perceiving the environment (the ship) to those who will represent decision makers like the Executive officer and the Commander who will decide upon several alternative actions taking the best utility into account. All along the system design, we followed an Agent-Oriented software engineering methodology, PORTO [3, 4] going through all its steps: i) requirements analysis; ii) analysis; iii) architectural design; iv) detailed design; v) implementation; and vi) testing and validation. Next section goes through the state of the art on this particular subject, section 3 formulates the problem followed by the requirement analysis and the system analysis (sections 4 and 5). Sections 6, 7 and 8 briefly specify the system organization, architecture and design. Last sections introduce implementation, testing and conclusions.
2 Related Work

The problem we intend to tackle has two main aspects. First, crew motion simulation and how it should act upon the equipment (e.g. propulsion, radars, weapons) and ship auxiliary systems (e.g. pipes, electrical distribution system); second, the multi-criteria decision process of crew limited resources distribution. The first aspect is not the object of our study. We will focus on the decision making problem on a ship under damage. As far as we know, the problem was first addressed in reference [5]. This paper presents SINGRAR, a decision aid software system that has been in use with success in the Portuguese Navy for several years, and got the best reviews in the military community. Its main component is a fuzzy expert system that was developed to assist the command and control (decision making) during battle, in order to maintain the equipment and auxiliary systems as much operational as possible while the ship is under attacks/ damages. The system has the ability to integrate the information gathered at different locations along the ship; and it proposes repair priorities and resources assignment under the scope of technical/logistical activities. The system has not been used since it is considered as too dependent upon human interaction and its decision process is based upon utility functions that make use of predetermined weights for each criteria, which may lead to non-efficient decisions. The use of multi agent systems in similar problems may be found in several references in the literature, such as references [3, 4]. This last reference, in particular, deals with how to manage disruptions in operations of airline, such as the ones caused by bad weather, malfunctions and crew absenteeism. In order to solve this problem the reference presents a new negotiation protocol entitled Generic Q-Negotiation (GQN), which includes the Q-learning algorithm. As far as the methodological approaches for the development of software based on agents we will follow the PORTO methodology [3, 4], which has its groundings in GAIA methodology [7], which will also be further mentioned in the text.

3 Problem formulation

The command activities of most war ships include two officers, the commanding officer and the executive-officer, which are advised by other officers with different technical knowledge and expertise, such as the weapons-officer, the engineering-officer, and the tactical-officer. Further, the crew is made of different petty-officers and unlisted men, all of them with their own expertise from the cook to the radar controller, and from the nurse to the electrician. In case of damage, or combat, the ultimate decision maker is the commanding officer; the responsible for the crew is the executive-officer; the other officers are the ones who decide on how to act in order to achieve the goals established by the commanding officer. We aim to develop a system that may be used in design of new ships to assess the platform independently of how good the crew is, so it should always select the best decision possible taking into account the ship and the scenario (environment). The system must be able to reflect the decision making process of the commanding officer, as far as prioritizing the internal battle space actions and crew resources assignment. For this paper we selected a simple scenario where an aircraft attacks the ship and it hits six times the same compartments/ equipment. As a consequence, there are three damaged equipment (radar, propulsion and weapon) and a fire in the engine room.

4 Requirements analysis

As mentioned in introduction, we are going to follow the PORTO methodology that is described in references [3]. This methodology proposes to start the procedure by goal-oriented early requirements analysis as in the methodology TROPOS [2]. After having selected the different actors and goals, we have built up the actors and goals diagram (figure 1), where interactions are presented, and several potential queries have been identified, namely:

1) "Query any crew" meaning that an actor requires intervention from crew members and it asks if there are any crew members available;
2) "Query equipment state" meaning that an actor requires knowing state of equipment;
3) "Query crew required" meaning that an actor requires knowing from equipment or another actor, who is required either to repair equipment or extinguish a fire in a compartment;
4) "Query available crew members" meaning an actor requires knowing the available crew members for the needed repair tasks;
5) "Query priorities" meaning that an actor requires knowing priorities other agent has established;
6) "Query which equipment" meaning that an actor requires knowing to which repair task he was assigned to.

5 Analysis

This stage has been imported from GAIA [7], an Agent-Oriented Software Engineering method, and it
includes five stages that will be presented separately.

5.1 Subdividing the system into sub-organizations

This stage consists primarily on analysing the problem trying to find sub-goals and sub-organizations dedicated only to achieve those goals. In our case, there are three distinct organizations, namely:

(1) Internal Battle state identification sub-organization;

(2) Decision making sub-organization;

(3) Crew distribution sub-organization.

5.2 Environment model

The aim of this stage is to distinguish between resources and active components. The first ones are variables or tuples made available to the agents. The second ones are components and services capable of performing tasks with which agents must interact. In our case, resources are: Aircraft information, Ship information, Crew information, Equipment information and Task requirements. The active components are: Damage Manager and Crew manager.

5.3 Preliminary role model

Preliminary roles are the functionalities and competences required to achieve the established goals, independently of the organizational structure that will be developed in the next stages of the methodology. Accordingly, by analysing the Actors and goals diagram (figure 1) we have established the following roles: AttackAction, DamageMonitor, NeedsMonitor, NeedsAuction (associated with Officers demands on needed personnel to the Commander), CrewMonitor (monitoring which tasks the crew is assigned to), AssignCrew (assigning roles to the crew), DecisionMonitor (associated with Commanders decisions on assigning priorities and crew members to tasks), ActionTasks.

5.4 Preliminary interaction model

It became then necessary to specify the needed interaction between roles, their dependencies and relationships. To establish the communication protocols we have used FIPA ACL (Agent Communication Language) protocols. PORTO methodology also indicates that at this stage it is better to build an Environment and preliminary roles diagram, which is presented figure 2.

6 Organizational rules

The last task of this stage is to analyse the relationships between roles, as well as between protocols and, also, between roles and protocols. According to the followed methodology, two different types of constraints and relations must be taken into account. Firstly, the liveness organizational rules explain how the dynamics of the organization should evolve (relations). Secondly, safety organizational rules state rules that are independent of the evolution and always true (constraints). In this work, the rules considered were the following:

(1) Liveness Organizational Rules;
Figure 2: Environments and preliminary roles diagram

(a) The situation may only be reported after the attack:
\[\text{reportattack}(\text{AttackAction}(\text{attack}(x))) \Rightarrow \text{reportsituation}(\text{DamageMonitor}(\text{attack}(x)));\]

(b) The decision can only be done by the Commander:
\[\text{DecisionMonitor}(\text{Commander}).\]

7 Architectural design

The architectural design consists of translating the previous work into a MAS architecture. This will imply definite decisions about how the next phases will be conducted, though some changes may be done due to implementation difficulties, as in this particular case where due to using a specific ACL protocol a single agent was replaced by other ones, simpler and more reliable agents. This phase includes:

1. defining the organizational structure;
2. completing the role and interaction model;
3. graphical representation using UML 2.0.

In our case we will include the final organizational structure and a combined representation of the model reached, after the first two tasks.

7.1 Organizational structure

The resulting organizational structure may be described by a set of rules mainly derived from the previous analysis leading to three different sub-organizations: "Internal Battle State" including roles "AttackAction", "NeedsMonitor" and "DamageMonitor"; "Decision Making" including the roles: "DecisionMonitor", "CrewMonitor" and "NeedsAuction"; "Crew Distribution" including roles: "AssignCrew" and "AssignTasks". Roles included in each one of the sub-organizations although tightly dependent between them, can also be related with roles of other different sub-organization.

7.2 Graphical representation

Finally, all previous work can be described in a single diagram where protocols abstractions and role abstractions are added to the previous graphical representation. To understand figure 3, it is then necessary to take into account both protocol abstractions and role abstractions: Protocol abstractions represent the interactions between roles represented in figure 2. Role abstractions include the attributes of the class, and the organizational rules seen as dependencies after recognizing the above mentioned sub-structures. As an example, since Role DamageMonitor relies on information from Role AttackAction (they belong to the same sub-organization, "Internal Battle Station", this can also be seen in figure 3 (conAttackOn = true).
8 Detailed design

Detailed design, according to PORTO methodology, is the stage where both the agent model and service model are produced. The first model is concerned with which agents are going to be implemented and the second is concerned with the services required and who will implement them. We have made the correspondence between agents and roles to the previously identified actors leading to the Agents Model presented in figure 4.

8.1 Service model

Now that we identified the agents and their roles, we can identify their services. Following both PORTO and GAIA methodologies, it will be defined for each service:

(1) input;
(2) output;
(3) pre-condition;
(4) post condition.

In figure 5 the services for four of the agents are presented: the Aircraft, the Commander, the Executive officer and the agent representing any Officer, while the ones missing are Ship, Equipment and Crew.

9 Implementation

This system was implemented using JADE [1] as support to build the different agents. We here only describe one of the most significant agent: the Commander Agent. The commander is responsible for the decision process, which is incorporated in the role DecisionMonitor. To implement this we made use of a utility function ($U$) where the proposal values ($p$) were
Figure 5: Service model for agents: Aircraft, Commander, Officers, Executive officer

multiplied by a set of weights \((w)\) set by the user, in such way that:

\[
1 = w_1 + w_2 + w_3 + w_4 \quad (1)
\]

\[
\text{final proposal}_i = w_ip_i \quad (2)
\]

\[
U = \sum_{i=1}^{4} w_ip_i \quad (3)
\]

So the best final proposal will be accepted by the Commander, instead of the best proposal. We intend to improve the decision process by making this decision process adaptable by means of Q-learning algorithm that learns the best possible weights for the intended outcome. At present, the Commander Agent is using the JADE ContractNetInitiator behaviour, which is a FIPA-compliant behaviour included in its library. Similarly to the ContractNetResponder, this behaviour is also defined by handles, namely the handleAllResponses which we had to change in order to incorporate the weighted utility function.

10 Testing

We have tested the system through the analysis of the messages sequence exchanged between agents and decide whether or not it was according to the expected. In fact, after the event Aircraft attacks the Equipment/compartment and the respective message, all the operational process starts until all the commander decisions on crew and resources assignment to tasks have been made. Since the agents messages exchange graph would not be visible clearly enough, we list most of the messages below:

1. Aircraft attacks the Equipment/Compartment by sending a message;
2. Equipment/Compartment send messages specifying its state (using objects) to the corresponding Officer and actualize their state in the Environment (stores the equipment/compartment state);
3. Officers send a message to the Evaluation Agents (one per equipment) containing the value to use as a proposal in the contract net (how many people are required);
4. Commander initiates a FIPA Contract net protocol in which the Evaluation Agents participate. Commander selects the best proposal multiplying each one he receives by a weight (from 1 to 10 and \(\sum W = 10\) input by the user);
5. The Evaluation Agent whose proposal was accepted sends a message to the Executive officer;
6. Executive officer sends a message to jades DF asking for specific service previously registered by the Crew (fire, mechanical repair and electronics repair);
Jade DF assigns the service to one of three different agents: Fire-fighter Agent; Engineer Agent; or Electronic Agent, previously registered.

After reply from the DF, the Executive officer sends a message to the corresponding Equipment/Compartment indicating that people are already assigned to it. The agent actualizes his state changing the number of people acting upon the damage.

Next time Aircraft attacks the Equipment/Compartment, upgrade their state and the cycle re-initiates.

The Equipment/Compartment that has no need for any more personnel (specified in his state) will send a message to the Officer who will send a message to the Evaluation Agent saying its proposal is 0, i.e., it does not require any personnel. After that, when participating in the contract network protocol, this last agent will refuse to send any proposal (proposal is 0).

After four runs, there should be no more demand for personnel and all participants in the contract net should be refusing the participation.

11 Future work

We feel the need to a more adaptive capability to the changing environment, by sensing exceptional situations in which the system has to immediately react even before the chain of command does it. We also need to introduce some uncertainty about the probability with which the ship, the compartments, and the equipment are really hit and damaged under each specific attack.

12 Conclusion

This work approaches a complex problem which is adequate to be solved by using multi agent systems. A great effort was applied in analysing and formulating the problem, applying the agent-oriented software engineering methodology PORTO, which is based on GAIA and TROPOS. As far as the implementation is concerned, we followed FIPA guidelines, using ACL protocols, and JADE. Although we recognize that further improvement is needed, mainly in the implementation of the reinforcement learning algorithm and in the agents autonomy, we believe that the groundings of a new method for the evaluation of new ship designs has been defined independently of crew response. Further, the same method may also be used to analyse existing platforms in order to define the best response procedures to be followed by the crew in case of a damaged event occurs, that may be used in training. All in all, the following objectives were achieved:

1. The system reflects a simplified version of the existing organization on board;
2. The system is able to reflect the limited crew resources attribution process;
3. The system is able to reflect the different knowledge of the crew and its number limitation;
4. The system is able to reflect the decision making process of the Commanding officer; the software system is partially independent from users interface.

References


