Agent-Based Approach for Ship Damage Control

Eugénio Oliveira(✉) and Paulo Martins

LIACC, DEI / Faculdade de Engenharia, Universidade do Porto,
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
eco@fe.up.pt

Abstract. We here introduce a Multi-Agent System (MAS) approach for solving the crew resources assignment problem whenever a ship, under attack, suffers several damages and, thus, priorities must be assigned in order for it to survive. In the designed system, the ship is the MAS environment and the attacker, equipment, crew and officers (these last ones seen as decision makers) are represented through agents. Decisions on resources assignment are taken after a negotiation process and using an utility-based selection process. Agent-based system design was accomplished by following a systematic Agent Oriented Software Engineering approach, called PORTO, leading to the specification and the implementation of the system.

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 [5]. We are here 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

N.T. Nguyen et al. (Eds.): ICCCI 2016, Part II, LNAI 9876, pp. 3–13, 2016.
DOI: 10.1007/978-3-319-45246-3_1

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. Other authors have already used the MAS paradigm for dealing with kind of disastrous situations [6]. Our MAS architecture includes different kinds of agents ranging from simple ones, reactive agents perceiving the environment (the ship) to those who represent decision makers like XO_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, Sect. 3 formulates the problem followed by the requirement analysis and the system analysis (Sects. 4 and 5). Sections 6, 7 and 8 briefly specify the system organization, architecture and design. Last sections introduce implementation, testing and conclusions.

2 General Characteristics of the Problem's Class

Many sophisticated problems we usually address can be classified as belonging to the so called 3D class of problems. They reflect a reality that simultaneously is of a Distributed, Decentralized as well as Dynamic nature. This means that, besides input data and output actions being disperse (Distributed) at different nodes, also, and most important, decision-making can be, at least partially, taken at different nodes of the (Decentralized) system. Moreover, the system trying to solve the overall problem at stake, has to deal with a changing, evolving reality (Dynamic). Although Agent Oriented Programming has been pointed out as a natural paradigm to cope with such situations, and because of the intrinsic autonomous property of Agents, it is up to those agents to find out in run-time, according to the current situation, how to interrelate in order to reach their own intended goals. Finally, and most important, for the same purpose, how to coordinate joint work or reach mutual agreements together with the other agents?

3 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 [7] introducing SINGRAR, a decision support system that has been in use with success in the Portuguese Navy. Its main component is a fuzzy expert system that assists the command (decision making) during battle, for keeping equipment and auxiliary systems operational while the ship is under attacks/damages. The system integrates the information gathered at different locations along the ship and it proposes repair priorities and resources assignment under the scope of logistics activities. The system is not now in use since it is considered too dependent upon human interaction and its decision process is based upon utility functions depending on predetermined weights that 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 airline operations, 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 [8], which will also be further mentioned in the text.

4 Specific 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 off 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 engines room.

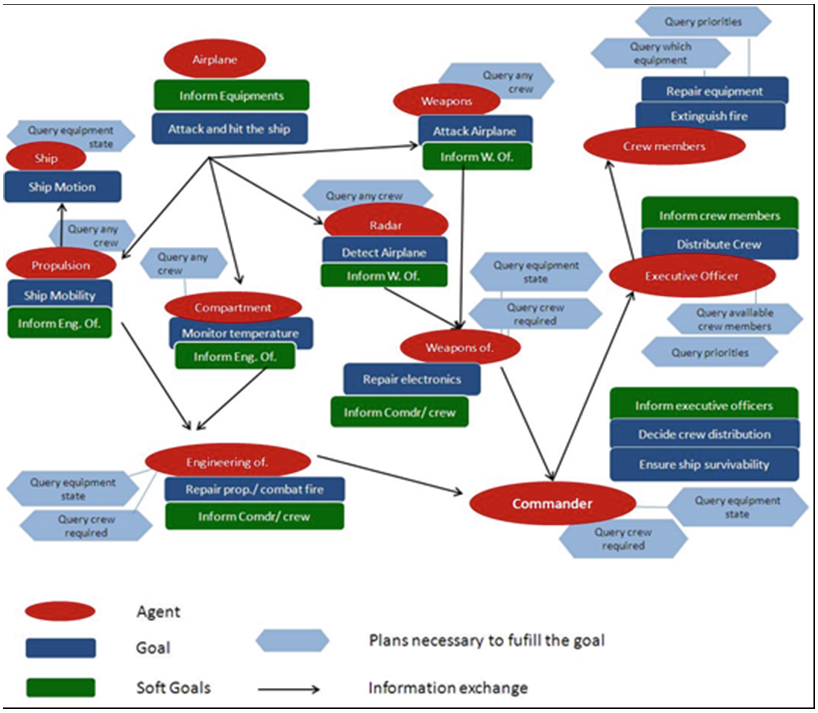

Fig. 1. Actors and Goals diagram

5 Requirement Analysis

As mentioned in introduction, we are going to follow the PORTO methodology that is described in reference [4]. 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 (Fig. 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 actor has established; (6) "Query which equipment" meaning that an actor requires knowing to which repair task he was assigned to.

6 Analysis

This stage follows GAIA [5], an Agent-Oriented Software Engineering method, and it includes five sub-phases that will be presented separately.

6.1 Subdividing the System into Sub-organizations

It primarily consists on looking to the problem trying to find sub-goals and sub-organizations dedicated only to achieve those goals. There are three distinct organizations, namely:

(1) Internal Battle state identification sub-organization;
(2) Decision making sub-organization;
(3) Crew distribution sub-organization.

6.2 Environment Model

We here distinguish between resources and active components. The first ones are seen as variables or tuples made available to the agents. The second ones are components and services capable of performing tasks with which agents must interact. Here, resources are: Aircraft information, Ship information, Crew information, Equipment information and Task requirements; Active components are: Damage Manager and Crew manager.

6.3 Preliminary Role Model

Preliminary roles relate with functionality and competences required to achieve the intended goals, independently of the organizational structure that will be further selected. Accordingly, by analyzing the Actors and goals diagram (Fig. 1) we identified 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.

6.4 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 Comunication Language) protocols. PORTO methodology also indicates that we should build an Enviroment and preliminary roles diagram presented in Fig. 2.

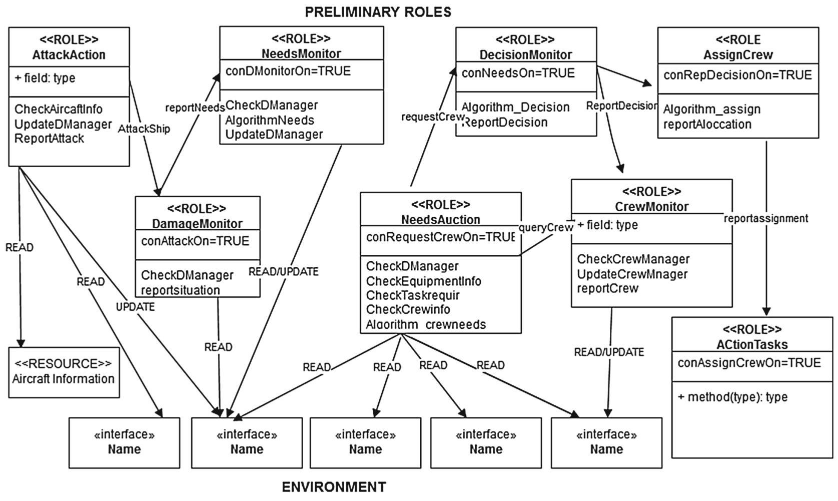

Fig. 2. Environments and preliminary roles diagram

6.5 Organizational Rules

The last task of this stage is to analyze the relationships between roles, as well as between protocols and, also, between roles and protocols. According to the followed methodology, the following constraints and relations must be taken into account: (1) Liveness organizational rules explain how the dynamics of the organization should evolve (relations); (2) Safety organizational rules state rules that are independent of the evolution and always true (constraints). Table 1 shows those above mentioned kind of rules.

7 Architectural Design

The architectural design consists upon translating the previous work into a MAS architecture. This will imply definite decisions about how the next phases will be conducted. Some changes may occur due to implementation difficulties, as it is the case of, due to using a specific ACL protocol, a single agent is replaced by others simpler and more reliable agents. This phase includes: (i) defining the organizational structure; (ii) completing the role and interaction model; (iii) 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:

Table 1. Liveness and safety rules

Liveness Organizational Rules	Description
reportattack (AttackAction(attack(x))) $\Longrightarrow$ *reportsituation (DamageMonitor(attack(x)))*	The situation may only be reported after the attack
1- **reportsituation (DamageMonitor(attack(x)))** $\Longrightarrow$ **reportcrew (CrewMonitor(attack(x)))**	The crew report may only be done after evaluating the new current situation
2- **querycrew (CrewMonitor(attack(x)))** $\Longrightarrow$ **requestcrew (NeedsAuction(attack(x)))**	The auction for crew ressources may only be done after knowing the new needs for crew resources
3- **requestcrew (NeedsAuction(attack(x)))** $\Longrightarrow$ **reportdecision (DecisionMonitor(attack(x)))**	The decision may only be done after the requests by the different officers
4- **reportdecision(DecisionMonitor(attack(x)))** $\Longrightarrow$ **reportallocation (AssignCrew(attack(x)))**	The allocation of crew resources may only be done after the decision on how to do it has been reported
4- **Safety Organizational Rules**	Description
4- **AttackAction1...n**	There will be n attacks by the aircraft
4- **DecisionMonitor(Commander)**	The decision can only be done by the Commander

"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 interdependent, 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 Fig. 3, it is then necessary to take into account both

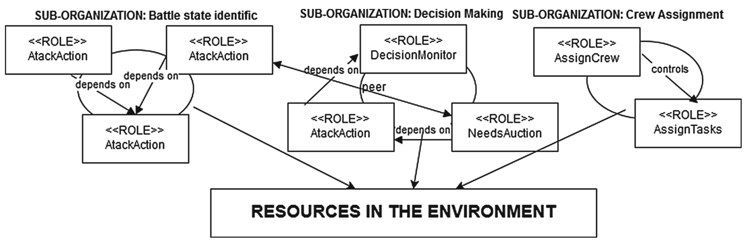

Fig. 3. Final role, interaction and enviroment diagram

protocol abstractions and role abstractions: Protocol abstractions represent roles' interactions as it is the case represented in Fig. 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 Fig. 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 following Agents Model (Fig. 4):

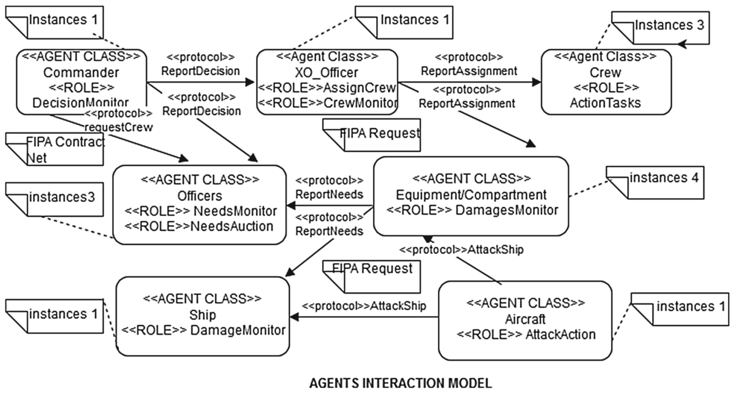
Instances 1
<<AGENT CLASS>> Commander <<ROLE>> DecisionMonitor
<<protocol>> ReportDecision
<<protocol>> ReportDecision
<<protocol>> requestCrew
Instances 1
<<Agent Class>> XO_Officer <<ROLE>>AssignCrew <<ROLE>>CrewMonitor
<<protocol>> ReportAssignment
<<protocol>> ReportAssignment
Instances 3
<<Agent Class>> Crew <<ROLE>> ActionTasks
FIPA Contract Net
FIPA Request
instances3
<<AGENT CLASS>> Officers <<ROLE>> NeedsMonitor <<ROLE>>NeedsAuction
<<protocol>> ReportNeeds
<<protocol>> ReportNeeds
<<AGENT CLASS>> Equipment/Compartment <<ROLE>> DamagesMonitor
instances 4
<<protocol>>AttackShip
FIPA Request
instances 1
<<AGENT CLASS>> Ship <<ROLE>> DamageMonitor
<<protocol>>AttackShip
<<AGENT CLASS>> Aircraft <<ROLE>> AttackAction
instances 1
AGENTS INTERACTION MODEL

Fig. 4. Agent Model

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: (i) input; (ii) output; (iii) pre-condition; (iv) post condition. Due to lack of space we are not able to show the figure displaying services used by the agents.

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

$$final\_proposal_i = w_i p_i \; and \; U = \Sigma_{(i=1...4)} w_i p_i \tag{1}$$

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 behavior, which is a FIPA-compliant behavior included in its library. Similarly to the ContractNetResponder, this behavior 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 is very complex and would not be visible clearly enough, we list most of the relevant messages below:

1 Aircraft attacks Equipment/Compartment by sending a message reporting it;
2 Equipment/Compartment sends message with new state to the corresponding Officer and updates the Environment state (equipment/compartment state);
3 Officers send a message to the Evaluation_Agents (one per needed equipment) containing the value to use as a proposal in the contract net (how many crew members 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 "importance factor" (weights from 1 to 10 and $\sum W = 10$ given by the user);
5 The Evaluation_Agent whose proposal was accepted sends a message to XO_Officer;
6 XO_Officer sends message to JADE's DF asking for specific services previously registered by the Crew (fire, mechanical repair and electronics repair);
7 JADE's DF assigns the service to one of three different agents: Firefighter_Agent; Engineer_Agent; or Electronic_Agent, previously registered;
8 After reply from DF, XO_Officer sends a message to the corresponding Equipment/ Compartment indicating crew members already assigned to it. The agent updates its state by changing the number of people acting upon the damage;

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

10 The Equipment/Compartment that has no more needs (specified in its 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 other personnel. After that, when participating in the contract net protocol, this last agent will refuse to send any proposal;

11 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 and Conclusions

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. In conclusion we want to emphasize that this work approaches a complex problem which is adequate to be solved by using multi-agent systems. A great effort was applied in analyzing 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 not all the intended goals were accomplished, we believe that the following objectives were achieved:

First and very important is the fact that the system reflects a simplified version of the existing organization on board and reflects the way limited crew resources assignment should be dealt with;

Also relevant is that the system takes into account the different kinds of knowledge of the crew and its respective limitations;

Finally, and decisive, the system is able to be, in an automatic way, close to the decision making process of the commanding officer;

References

1. Bellifemine, F., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley, Chichester (2007)
2. Bresciani, P., Perini, A., Georgini, P., Giunchiglia, F., Myloupoulos, J.: Tropos: An agent-oriented software developemnt methodology. J. Auton. Agent. Multi Agent. Syst. **8**(4), 203–236 (2004)
3. Castro, A., Oliveira, E.: The rationale behind the development of an airline operations control centre using gaia based methodology. J. Am. Soc. Inf. Sci. Technol. **2**(3), 350–377 (2008)
4. António, J., Castro, M., Rocha, A.P., Oliveira, E.: A New Approach for Disruption Management in Airline Operations Control, vol. 562. Springer Verlag, Heidelberg (2014). studies in computational intelligence

5. Rossetti R., Brito Carvalho Martins, T.: Ship damage control action simulation using hla. In: Proceedings of The International Conference on Harbour, Maritime and Multimodal Logistics Modelling and Simulation, HMS, pp. 487–499. ACM Press (2013)
6. Scafes, M., Badica, C.: Preliminary design of an agent-based system for human collaboration in chemical incidents response. In: Proceedings of the 7th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, pp. 53–62. Scitepress (2009)
7. Marques, M.S., Pires, J.: Singrar â a fuzzy distributed expert system toassist command and control activities in naval environment. Europ. J. Oper. Res. **145**, 343–362 (2003)
8. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the gaia methodology. ACM Trans. Softw. Eng. Method. **12**(3), 317–370 (2003)