Implementation and Assessment of Robot Team Cooperation Models Using Deliberative Control Components

José Manuel Gascueña Noheda, Francisco Garijo, Antonio Fernández-Caballero, Marie-Pierre Gleizes, Pierre Glize

Implementation and Assessment of Robot Team Cooperation Models Using Deliberative Control Components

José M. Gascueña[1], Francisco J. Garijo[2], Antonio Fernández-Caballero[1], Marie-Pierre Gleizes[2], and Pierre Glize[2]

[1] Universidad de Castilla-La Mancha, Departamento de Sistemas Informáticos & Instituto de Investigación en Informática de Albacete (I3A), 02071-Albacete, Spain
{JManuel.Gascuena,Antonio.Fdez}@uclm.es
[2] IRIT - Institut de Recherche en Informatique de Toulouse
118 route de Narbonne, 31062 Toulouse Cedex 4, France
fgarijo@pdi.ucm.es, {gleizes,glize}@irit.fr

Abstract. This paper proposes an organizational framework and an approach for development of cooperation models for teams of Autonomous Adaptive Vehicles (AAV), using goal-driven control components. The framework and the approach are illustrated through the development and assessment of task allocation in multi-robot teams. Two cooperation models have been implemented: i) a team model based on the Adaptive Multi-Agent Systems (AMAS) theory, where task responsibility is agreed between team peers, by exchanging individual estimations of the degree of difficulty and priority to achieve the task; ii) a hierarchical model where an AAV manager asks team members estimations and then assigns the task. Experimentation for team-cooperation assessment has been done taking into consideration environmental changes, communications and internal failure. The most significant results reported in this work concerns team coordination in stressing situations. The experimental setting and team performance figures are detailed in the paper.

Keywords: adaptive multi-agent systems, agent framework, robotics, distributed task allocation, cooperation models.

1 Introduction

Component based approaches are increasingly used to deal with heterogeneity and complexity of robotic systems [2]. A key advantage of componentization is to allow development of simulated models which could be seamlessly deployed fully or in part into the robot hardware. Ongoing work on robot simulation tools are also in this direction [3]. This paper proposes a component based layered architecture for Autonomous Adaptive Vehicles (AAV) which control is based on a deliberative goal driven agent pattern [10]. High-level deliberative control facilitates development and experimentation with different behaviour models by bridging the gap between analysis, design and implementation. It also allows

reusability, and ease traceability of the control process which is based on high level constructs close to human behaviour. However common pitfalls are: hard integration with software engineering standards, poor performance, and difficulty to control the deliberative process. Therefore, integration of symbolic deliberative components with imperative components is still a challenge.

This paper describes an architectural framework for implementing teams of AAVs capable to achieve individual and collective mission goals by taking into account unexpected changes in the environment, internal failure and availability of mission resources. The work is part of the research effort undergone in the ROSACE (Robots and Embedded Self-Adaptive Communicating Systems) project (http://www.irit.fr/Rosace,737), where the experimental setting is based on a simulated fire forest crisis management scenario where AAV teams should cooperate among themselves and with a Control Center broadcasting requests for helping potential victims jeopardized by fire. Ongoing work in ROSACE joints research efforts on MultiAgent System (MAS) coordination in other domains such as poisonous material accidental release in a city [4]. In the Combined System (http://combined.decis.nl) project, agents are used to implement a collaborative decision system for handling crisis situations. They are responsible for the coordinate tasks, plan actions and reroute information of different actors from different rescue organizations. Users can also benefit from agents' information using a dedicated geo-spatial language named OpenMap, and a dedicated interaction language named Icon. Multi-agent-based Distributed Perception Networks (DPN) are also another relevant application of multi-agent systems to crisis management by intelligently aggregating information coming from a network of sensors [11]. Our works focus on sensor and data, and the intelligence is embedded outside the devices, which implies a notable delegation for a tier of computing services. While most of the experimental results focus on simulated coordination for *best cases*, the most significant results reported in this work concerns team coordination in stressing situations. Performance testing has been done considering different team size, tasks to be achieved, and AAV deployment in different processing nodes in order to assess the impact of communication.

The rest of the paper is organized as follows. Section 2 outlines the architectural principles for AAV design, and the rationale for adopting a goal oriented approach for implementing AAV control and team-cooperation. This approach is illustrated with the development of two cooperation models in the ROSACE project experimental setting: i) AMAS (Adaptative Multiagent System) model where each team member evaluates the cost to achieve the goal, sends its evaluation to their peers and then assumes the goal if it has the most suitable evaluation; ii) hierarchical model where a manager, asks each peer to estimate the evaluation of a given goal, then it proceeds to assign the goal to the most suitable peer. Section 3 details assessment metrics, and testing results using different configuration made up of various team size and number of victims to rescue. Stress testing has been done to compare both functional issues and performance issues on the AMAS model and the hierarchical model. Finally, conclusions and open issues are summarized in Sect. 4.

2 A Goal Oriented Approach for AAV Control and Cooperation

The proposed approach relies on a multi-layered component based architecture, which is populated by manageable components offering their services to other components through standard interfaces (see Fig. 1). The AAV behaviour is governed by the Robot Global Control Component (RGC) which gathers elaborated information from the rest of the components, makes choices, orders execution of actions, monitors results, and sends control information to relevant components when is necessary.

RGC is implemented with a declarative goal processor [5] that manages a goal space, and a working memory. Strategic and tactics criteria for generating goals and executing tasks and actions, in order to try to achieve goals, are defined by means of situation-action rules, where the situation part specifies a partial state of the working memory including the objective and its internal state, and

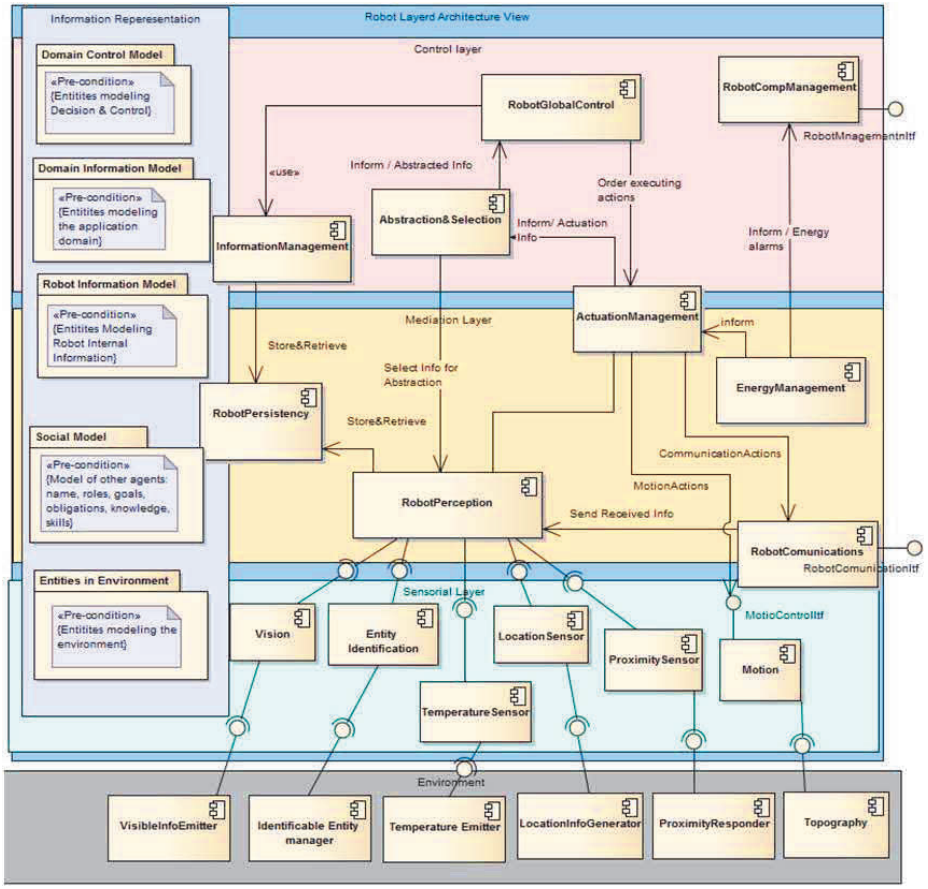

Fig. 1. General architecture

the action part contains statements for executing tasks. The processing cycle is droved by incoming information which is stored in the working memory. Then control rules are used to decide either to generate new goals, to focus on a new goal, to verify the resolution of pending goals, or to proceed to the resolution of pending goals by executing new tasks and actions. Componentization allows seamless integration of real or simulated components, then facilitating modelling, encapsulation and reuse of control strategies and cooperation models.

2.1 Using the Approach for Developing Team-Cooperation Models

Work on team-cooperation focus on evaluating different control architectures and cooperation models allowing an AAV-team achieve efficiently mission goals. The experimental setting for AAV operation is based on fire forest crisis management scenarios. The AAV team is situated in the intervention area to help people jeopardized by fire. The Control Center (CC) broadcast requests to help potential victims indicating the priority, location, and additional detail when needed.

The team should be capable of interpreting and evaluating the CC requests taking into account their current work-load, then decide which member of the team would assume the goal for helping the victim, and finally the team-mate who have accepted the responsibility for the goal should proceed to help the victim. When new requests for helping victims are sent by the CC, the team should reallocate their current goals in order to satisfy the new demands.

Initial experiments have started implementing the AMAS cooperation model [9], [1], [8]. AAVs are supposed to have a cooperative attitude which allows them to take decisions in order to sharing resources and/or assuming goals (tasks), avoiding possible conflicts. The generic process for team-cooperation to achieve this common goal is the following.

Each AGV: (i) estimates the cost to achieve the new goal; (ii) sends its estimated cost to the team members; (iii) receives estimated costs from team members, and (iv) takes a decision to assume the goal based on the estimations received from its peers. Three cases might happen. (C1) The agent has the best estimation: it sends to their peers its proposal to achieve the goal, and waits to receive their confirmation. (C2) There are other team-mates better suited than him to achieve the goal: it sends them the agreement for them to achieve the goal. (C3) The agent has the optimal cost, but it is tied with other team-mates: the tied peers add an randomly generated number to their estimations, and send the new estimation to tied peers, in order to allow one of them to take the responsibility of the goal.

Goal Allocation and Cost Estimation. A formal definition for the multi-robot goal allocation problem can be defined as *"given a number of goals, G_1, G_2, ..., G_t, a team of robots, R_1, R_2, ..., R_r, and a function $F_{eval}(G_i, R_k)$, that specifies the evaluation function (cost) of allocating goal G_k for the robot R_r, find the assignment that allocates the goal for the robot with lowest cost according the established criterion by the evaluation function"*. The cost evaluation function is calculated as follows. Goals consist on helping victims which should be rescued according to its priority. When there two victims with the same priority the victim that was notified first will be helped first. $PriV_v$ is the priority of victim

V_v. $Q_R = V_1, V_2, ..., V_{new}, ..., V_{n-1}, V_n$ is an ordered list by priority of victims that are assigned for robot R, where the new notified victim (V_{new}) is the victim that is being considered by the team in order to decide which robot will be responsible for its rescue. D(V_i,V_k) is the distance between the victims V_i and V_k. D(R_r,V_1) is the distance between the robot R_r and the first victim (V_1) located in Q_R. The distance is calculated using the Euclidean distance formula, in three-dimensions. Length(Q) is the number of victims of Q. Using these concepts, for goal (victim) allocation the evaluation function is defined as follows:

$$\begin{aligned} &\text{If VPD } > \text{ RE then } F_{eval} = -1.0 \\ &\text{else } F_{eval} = W_1 * VPD + W_2 * AT \end{aligned} \tag{1}$$

where RE is the robot available energy; VPD is the path distance to visit each victim belonging to Q (see equation 2); AT is the required time for attending victims belonging to Q (see equation 3); W_1, W_2 and W_3 are weights for VPD, AT and $PriV_v$ values. For the experiments the weight values are $W_1 = 10.0$; W_2=3.0; W_3=3.0.

$$VPD = D(R_r, V_1) + \sum_{i=1}^{n=Length(Q)-1} D(V_i, V_{i+1}) \tag{2}$$

$$AT = \sum_{i=1}^{n=Length(Q)} W_3 * PriV_i \tag{3}$$

Performance evaluation of the goal allocation algorithm is based on the following three parameters. (1) The time required for a goal to be assigned to a team-mate. This time is calculated using the processor real time clock, as the time difference between the instant when the control center sends the request, and the instant when the goal to help the victim was accepted by a team-mate. (2) Goal distribution among team members. (3) The cost of the robot team, which corresponds to the highest cost of the goals assumed by each team member.

Dealing with Uncooperative Peers. Cooperation comes out from the need by each agent to get information from their team mates to achieve their own goals. The cooperation process is highly dependent on team-communication which quality cannot be guarantee in a hazardous and changing environment as the fire-forest. Cooperation might fail when communication is missing, and also due to internal processing factors such as lack of synchronization in the cooperation process, and malfunctioning of internal components like sensors, motion, vision, position, computing, and others. Consequently each agent should be capable to deal with situations where: i) they cannot communicate with their peers; ii) communication is possible but team-mates do not send the expected information, and/or they do not respond to requests; and iii) they send unexpected or out-dated information. In these cases individual decisions should be taken to achieve the goals/tasks requested by the CC. To cope with "worst cases" which correspond to real situations the AAV team model has been extended to take into account: deadlines for decision making, missing information from the team mates, the current workload of the AAV, and stressing requests from the CC.

A *hierarchical team model* has been implemented to have a reference for assessing the strengths and weakness of the AMAS model, and for the utilization of a "heavy deliberative control architecture" for implementing these models.

2.2 Implementation Approach Using ICARO's Deliberative Control

The ICARO framework has been used in ROSACE and in other areas to model AAVs using reactive patterns [6,7] which control is based on a Finite State Automata. The deliberative control pattern is based on a goal processor. AAV behaviour is characterized by: i) the set of goals which can be achieved; ii) the activities, process and actions needed to achieve the goals; iii) the information model representing the domain and environmental entities, the computing entities needed for representing goal achievement states, and intermediate results produced by activities and actions, and iv) the process defining the life cycle of goals. This is done through situation-action rules expressing conditions for: a) goal generation; b) goal focalization; c) goal achievement, and d) executing activities and actions to make it possible that pending goals satisfy their achievement conditions.

Goals are represented as classes from which multiple object instances can be generated. Activities and actions needed to achieve goals are represented as tasks. The work-flow of activities and actions needed to achieve goals are first defined with UML activity diagrams, and then implemented with situation-action rules. In AMAS team model, the goal resolution process is defined with 41 structured rules. From each behaviour model multiple distributed deployment instances might be generated. The ICARO framework provides deployment, monitoring and communication transparency among component instances.

The **AMAS team-model** is implemented with a common behaviour model for all AAVs. Teams are made up of cloning instances; they have the same goals, tasks, information model and goal-resolution rules. Requests sent by the CC are received by all team-members which generates similar goal instances: *helpVictim()* and *decideWhoShouldGo()*. Cooperation is modeled in the protocol for making collective decisions, that is, to achieve the goal *decideWhoShouldGo()*. This is done by exchanging cost estimations, and then deciding which member of the team is the best situated to help the victim.

Although all the team members participate voluntary on the decisions process, the way in which each AAV achieves their own goals is dependent on its situation in the environment and on its internal state which is characterized by information objects in its working memory including previous goals and the current focus representing the goal under resolution. Experimentation has been done for fine-tuning the model in order to: 1) allow the AAV takes individual decisions when collective decisions fails; 2) determine deadlines for expected information and for taking collective decisions. As most of these parameters are dependent of hardware and communication performance, they are defined as configurable.

The **hierarchical team-model** has two roles implemented with two behavioural models: a) the *AAV-boss* which is in charge of interpreting the requests from the CC and deciding which team-member should be assigned to achieve the goal; b) the *AAV-subordinate* which receives messages from his boss, first

requesting to estimate their cost for achieving the goal, and then to accept/refuse proposals for assuming the goal. Subordinates might refuse proposals when they do not have the necessary means to achieve it, however the final decision to assign the goal correspond the boss. Deadlines for expected answers and deadlines for taking decisions are similar to the AMAS model. The information model is the same as for AMAS, goal and tasks might also be shared, however the boss role is implemented with 15 rules, and the subordinate role with 6 rules.

The system is implemented in java. It may run in a central node or in a network of processing nodes with OS windows/linux and virtual machine Java 6.xx. The rule processor used for implementing the deliberative agent pattern is based on Drools 5.x. and communication among AAVs is done through RMI. A public version will be available for demonstration during the conference.

3 Experimental Results

Metrics to assess both the model and the implementation approach using the deliberative architecture considers two main aspects: functional conformity and performance. Functional conformity focus on the quality of goal allocation, and goal distribution among team members. Performance considers the time needed for the team to assume goals for helping victims requested by the CC. Metric values have been gathered from testing experiments considering the following parameters: i) the team size and the number of victims to rescue; ii) the frequency of messages sent by the CC in order to assess the response of the team face up to stressing requests; iii) deployment in different processing nodes to assess the impact of real parallel processing and communication.

Experimentation in one central node has been done in a processor AMD Phenom II X4 at 3,20 GH with 4MB Ram and SO Windows 7. The two additional nodes for distributed experiments are based on Intel core I7 at 2,20 Ghz with 8Gb of Ram, SO windows 7 , and AMD Turion X2 at 2 Ghz , 2Gb of Ram and SO Windows XP. The most significant results are summarized below.

The **AMAS model** works as expected in situations where the CC sends requests at frequency greater than the time needed for deciding the responsibility to assume the goal. As the time required to take decisions increases with the size of the team, deadlines for waiting responses and for taking decision should also be increased to synchronize goal resolution. When deadlines are not met the same goal might be assumed by two or more team members, however this rarely happens. Tie-brakes for cost evaluation are satisfactorily solved.

Fig. 2 shows performance results for AAVs deployed in one central node and deployed in 3 nodes. Time for allocating goals is quite similar. Stressing requests degrade team performance due to the perturbation caused by the interpretation of incoming requests during collective decision making.

The first consequence of increasing the frequency of CC requests is desynchronising the process for achieving goals. CC messages are received at different time and processed at different speed by team-peers. When a team-member receives a request from the CC, it retrieves the victim's priority and generates new goals for helping the victim and for deciding who should assume that goal. If the priority

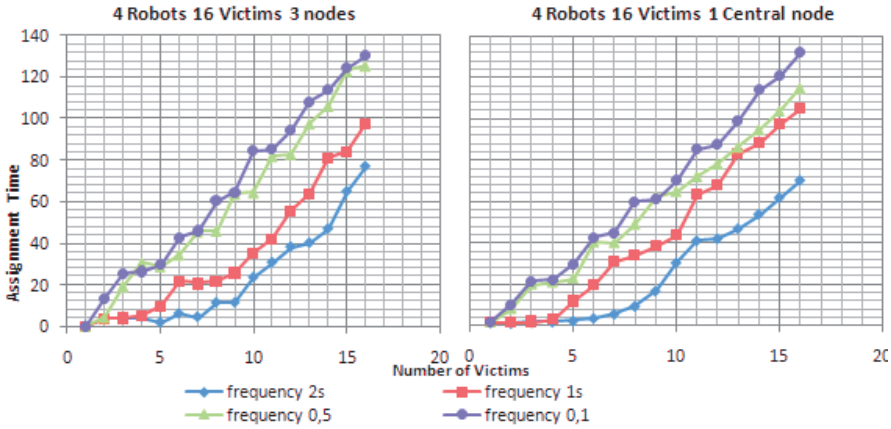

Fig. 2. AMAS model goal assignment

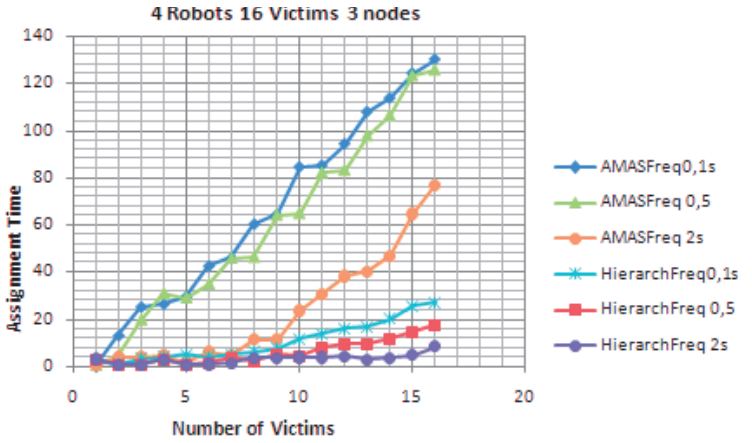

Fig. 3. AMAS model versus Hierarchical model

of the new victim is higher than the victim which decision is trying to achieve, it delays the resolution of the current goal and starts a new decision process to help this new victim. It is assumed that their team-mates will do the same, consequently it estimates its cost to achieve the goal and send it to its companions. However it happens that team-peers receive cost estimations and requests for sending its estimations before the message from the CC was processed. This lack of synchronization might lead to various peers taking the responsibility to assume the same goal. To deal with this situation, the peer receiving cost estimations, or requests for sending estimations about unknown victims, acts as if it were informed by its peer about the CC request. It trusts peer's information, and then it generates the goals and starts participating in the decision process. When the CC request arrives interpretation is already done. If the CC request cannot be received the AAV has been indirectly informed through its team-mates. Goal desynchronisation delays decisions due to multiple interruptions during the

decision process, and consequently decrease team-performance but the goals still allocated correctly. Experimentation shows (see Fig. 2) progressive degradation of performance when stressing demand increase, however quality still assured. This confirms the robustness of the model.

Performance of the hierarchical model compared to the AMAS model is in Fig. 3. Centralization of CC message interpretation and decision making facilitates conflict resolution reducing the number of messages needed for goal assignment. Performance with respect to the AMAS model is shown in Fig. 3. It is 10 times faster than AMAS model, however stress has more impact in its performance. Stressing requests degrades performance by a factor of 3,3 while the impact in the AMAS is of 1,6. The main weakness of this model concerns robustness since the efficiency of the team is dependent on boss decisions. If the boss fails or communication among the boss and the subordinates fails, the team became inactive.

4 Conclusions and Future Challenges

Experimentation with decision models using deliberative architecture requires availability of engineering tools facilitating quick development, deployment and evaluation. Face to the wide range of papers devoted to team modeling, availability of systems allowing verification and extension of these models are scarce. Work have faced two related challenges: model validation taking into account realistic constraints, and engineering evaluation mainly focused on the utilization of heavy deliberative architecture for controlling the behaviour of complex entities such as AAVs. Experimentation has gone beyond "best cases" to be focused on stressing test cases in order to validate key aspects of cooperative decision making such as performance, quality and robustness. The most significant results are obtained in *worse case scenarios* were team-members should face up with internal failure, communication failure, and stressing requests. AMAS performance is significantly lower than the hierarchical model; however this weakness might be compensated by higher robustness. Stress decrease performance in both models, most significantly in the hierarchical model, but quality is guaranteed. Utilization of encapsulated deliberative architecture facilitates high level modeling, and the traceability of the collaborative decision making process, then allowing incremental development and bridging the gap between analysis design and implementation. Seemly creation of multiple parallel instances can be done without penalizing deployment and performance.

The current system is made up of open source, reusable, components provided by ICARO . Extensibility, manageability, integration and deployment might be done with most popular IDEs. This paves the way to the development and experimentation with new team models were teammates might change dynamically their role. For example, the implementation of a team which starts hierarchical but becomes AMAS when the boss looses connection with their peers, can be done without significant effort. Other models such as selecting a new boss or creating partial hierarchy for big teams might be quickly developed.

Future work aims to go beyond simulation to validate the models incorporated into actual AAVs evolving in a physical environment.

Acknowledgements. This work is supported by the French RTRA-STAE foundation (Reseau Thematique de Recherche Avancee Sciences et Technologies pour l'Aeronautique et l'Espace) in the scope of the ROSACE project. This work is also supported by the Spanish Ministerio de Ciencia e Innovación / FEDER under project TIN2010-20845-C03-01 and by the Spanish Junta de Comunidades de Castilla-La Mancha / FEDER under projects PII2I09-0069-0994 and PEII09-0054-9581.

References

1. Clair, G., Kaddoum, E., Gleizes, M.-P., Picard, G.: Self-Regulation in Self-Organising Multi-Agent Systems for Adaptive and Intelligent Manufacturing Control. In: 2nd IEEE Intl. Conf. on Self-Adaptive and Self-Organizing Systems (SASO 2008), pp. 107–116 (2008)
2. Domínguez-Brito, A.C., Santana-Jorge, F.J., Cabrera-Gámez, J., Hernández Sosa, J.D., Isern-González, J., Fernández-Perdomo, E.: Exploring Interfaces in a Distributed Component-based Programming Framework for Robotics. In: 4th Intl. Conf. on Agents and Artificial Intelligence (ICAART 2012), pp. 667–672 (2012)
3. Echeverria, G., Lassabe, N., Degroote, A., Lemaignan, S.: Modular Open Robots Simulation Engine – MORSE. In: 2011 IEEE Intl. Conf. on Robotics and Automation (ICRA 2011), pp. 46–51. IEEE (2011)
4. García-Magariño, I., Gutierrez, C., Fuentes-Fernández, R.: Organizing Multi-Agent Systems for Crisis Management. In: 7th Ibero-American Workshop in Multi-Agent System (IBERAGENTS 2008), pp. 69–80 (2008)
5. Garijo, F., Bravo, S., Gonzalez, J., Bobadilla, E.: BOGAR LN – An Agent Based Component Framework for Developing Multi-Modal Services Using Natural Language. In: Conejo, R., Urretavizcaya, M., Pérez-de-la-Cruz, J.-L. (eds.) CAEPIA/TTIA 2003. LNCS (LNAI), vol. 3040, pp. 207–220. Springer, Heidelberg (2004)
6. Gascueña, J.M., Fernández-Caballero, A., Garijo, F.J.: Using ICARO-T Framework for Reactive Agent-Based Mobile Robots. In: Demazeau, Y., Dignum, F., Corchado, J.M., Pérez, J.B. (eds.) Advances in PAAMS. AISC, vol. 70, pp. 91–101. Springer, Heidelberg (2010)
7. Gascueña, J.M., Fernández-Caballero, A., Navarro, E., Serrano-Cuerda, J., Cano, F.A.: Agent-Based Development of Multisensory Monitoring Systems. In: Ferrández, J.M., et al. (eds.) IWINAC 2011, Part I. LNCS, vol. 6686, pp. 451–460. Springer, Heidelberg (2011)
8. Georgé, J.-P., Peyruqueou, S., Régis, C., Glize, P.: Experiencing Self-adaptive MAS for Real-Time Decision Support Systems. In: Demazeau, Y., et al. (eds.) PAAMS 2009. AISC, vol. 55, pp. 302–309. Springer, Heidelberg (2009)
9. Gleizes, M.-P., Camps, V., Georgé, J.-P., Capera, D.: Engineering Systems Which Generate Emergent Functionalities. In: Weyns, D., Brueckner, S.A., Demazeau, Y. (eds.) EEMMAS 2007. LNCS (LNAI), vol. 5049, pp. 58–75. Springer, Heidelberg (2008)
10. Lacouture, J., Gascueña, J.M., Gleizes, M.-P., Glize, P., Garijo, F.J., Fernández-Caballero, A.: ROSACE: Agent-Based Systems for Dynamic Task Allocation in Crisis Management. In: Demazeau, Y., Müller, J.P., Rodríguez, J.M.C., Pérez, J.B., et al. (eds.) Advances on PAAMS. AISC, vol. 155, pp. 255–260. Springer, Heidelberg (2012)
11. Maris, M., Pavlin, G.: Distributed Perception Networks for Crisis Management. In: 3rd Intl. Conf. on Information Systems for Crisis Response and Management (ISCRAM 2006), pp. 376–381 (2006)