EPiC Series in Computing

Volume 41, 2016, Pages 214–13

GCAI 2016. 2nd Global Conference on Artificial Intelligence

Cooperation in Adaptive Multi-Agent Systems through System of Systems modeling

Teddy Bouziat, Valérie Camps, Stéphanie Combettes

IRIT, Paul Sabatier University - Toulouse III

Abstract

This paper addresses the modeling and design of Systems of Systems (SoS) as well as inter multi-agent systems cooperation. It presents and illustrates a new generic model to describe formally SoS. Then, this model is used to propose a study of inter-AMAS (Adaptive Multi-Agent System) cooperation. Each AMAS, reified as a component-system of a SoS, uses a cooperative decision process in order to interact with other AMAS and to collectively give rise to a relevant overall function at the SoS level. The proposed model as well as the inter-AMAS study are instantiated to a simulated resources transportation problem.

1 Introduction

The evolution of technologies offers new services in ambient intelligence, IOT, ICT, Factory of Future etc. In those domains, systems tend to be more and more complex. One part of this complexity comes from the high number as well as the dynamics of interrelationships in these complex systems [13][4]. Indeed, complex systems are generally composed of many interdependent subsystems that usually have been independently designed but that are linked together to fulfill an overall goal [14]. In a general way, subsystems and the "global" system have to dynamically adapt the entire architecture to propose the best solution. Moreover all of these systems are often plunged into a dynamic and opened environment.

To face this complexity, current researches on SoS focuse on a large variety of problems [17] to develop new methods of engineering or architecting. SoS architecting research focuses on how, in an efficient manner, a SoS can have a dynamic, network-centric and collaborative architecture [14]. SoS literature shows that Agent-Based Modeling and Simulation (ABM&S) is a natural way to develop this new kind of architecture [3] [8] because they permit to describe and test new architecture dynamics, by varying the behavior of component-systems in the SoS. This paper presents a new model for formalizing SoS as well as an adaptive multi-agent approach for implementing SoS based on cooperation between component-systems.

After a general introduction, section 2 offers an overview of SoS, SoS architecting, architecting methodologies based on ABM and collaboration as well as the AMAS approach. Section 3 contains the description of SApHESIA (Sos Architecting HEuristic SImulAtor) model, a new one to implement SoSs. The cooperative decision algorithm of each component-system that enables dynamic and cooperative architecting is described in Section 4. Section 5 explains the

Table 1: Classical systems vs SoS architecture

Classical systems	SoS
• Controlled development	• Collaborative emergent development
• Domain specific systems level	• Network-centric
• Static architecture	• Dynamic architecture

case study for the evaluation of our proposal. Section 6 contains an instantiation of SAphESIA model and cooperative architecting as well as a comparison with different experiments on a resources transportation problem. We conclude and plan some future works in section 7.

2 Related Works

Even if the definition of SoS is not consensual, [13] explains that a SoS is *"an integration of a finite number of constituent systems which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal"*. Moreover, there is an agreement on what their main characteristics are [17]. For instance, Maier in [16] gives two main characteristics of a SoS : *"A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties: (1) managerial independence of the components and (2) operational independence of the components."*. More recently, these widely accepted characteristics have been extended by Firesmith in [11] : *"a SoS is a particular kind of system where each constituent tends to be: (1) managerial independent, (2) operationally independent, (3) physically distributed, (4) heterogeneous and (5) reusable."*. Finally, in [5], authors propose a SoS concept map that can be sum up as: *A SoS is a system: with no clear boundary, exhibiting complex properties(Emergence, Heterogeneity, evolution...) and composed of systems linked together and that are not SoS*. Notice that this last definition does not permit recursive definition of a SoS.

System of Systems Architecting - SoSs tend to have distributed control, and component-systems tend to choose themselves to participate or not in a SoS (i.e. decide to consume resources to achieve goal of the SoS). In other words, SoS architecting tends to be dynamic and focuses on interactions between component-systems. According to [13], SoS architecture is one of the main problems for developing SoS as the classical system architecting is really far from SoS architecting. Azani [3] identifies the main differences between classical engineering architecture and SoS one (table 1). The emphasis on SoS concerns interface architecting to foster collaborative functions among independent systems and the focus is put on the way to choose the collection of systems satisfying the requirements. Thus, contrary to classical systems, SoS architecting focuses on collaboration between component-systems to get the right organization.

ABM&S (Agent-Based Modeling & Simulation) are powerful techniques to model and simulate SoS. Indeed, Bonabeau in [6] wrote that it is best to use ABM when *"the interactions between the agents are complex, nonlinear, discontinuous, or discrete [...] the population is heterogeneous,[...]*; when *the topology of the interactions is heterogeneous and complex,[...]* and when *the agents exhibit complex behavior, including learning and adaptation."*. Thanks to these characteristics, ABM&S have been used to study SoS and proposed new ways to architecture

them using collaboration between components.

Collaborative Architecting - In [8], authors develop a methodology to model collaboration between systems, which uses a global social utility function for SoS. Based on satisficing game theory [18], this function allows the SoS to calculate the best options from the preferences and interdependencies of component-systems. To calculate its preferences, each component-system has two "roles": one based on the effectiveness of an action and the other one based on the inefficiency of an action and an interdependence function is computed from inter-dependencies of systems However, this approach is limited by the complexity of the construction of this function as designers have to define all inter-dependencies of component-systems, which are statics problem-dependent and difficult to define with numerous systems [18].

Agent-Based Wave Model - The methodology based on an agent-based wave model developed in [2] couples a genetic algorithm, fuzzy logic and negotiation to propose new architecture of SoS during time. In this model, a variable represents the propensity for an agent to collaborate with the SoS and other component-systems. For the genetic algorithm part, a chromosome is used as a representation of the current SoS architecture. Then, a fitness function defined by a fuzzy assessor is able to propose and to rate new chromosomes (representing new SoS architectures). The limitation of this approach is the need to design a fitness function that is problem dependent, and the irrelevancy of the assessor that leads to the irrelevance of the proposed architecture.

The AMAS approach - This approach is relevant to design adaptive multi-agent systems. It enables to solve complex problems that can be incompletely specified and for which an *a priori* known algorithmic solution does not exist. It considers the system as composed of parts (i.e. agents) and focuses on the local agent behavior for making them adaptive (to their local environment) while ensuring that the collective behavior emerging from interactions between agents is the one expected; in that case the system is said "*functionally adequate*". Each agent must have a local cooperative behavior for this purpose [9] [12]. The AMAS approach incorporates the notion of **criticality**, defined as the "*distance between the current situation and the local purpose of the agent*" [15]. Thus, "the further the agent is from its goal, the more critical it considers its current situation". Considering this notion, an agent is cooperative if it acts in order to help the most critical agent of its neighborhood. Thus, all agents within an AMAS try to continuously reduce the criticality of the most critical agent (possibly itself), while avoiding another agent becoming even more critical. If an agent is not able to help the most critical agent of its neighborhood, it may help other less critical agents. Thus, doing so, it hopes these agents will be able to help the most critical agent thanks to the reduction of their own criticality.

3 SApHESIA model

We propose a SoS model to be able to model more expressive problems than existing SoS models [1] [4], to compute them and to compare architecting approach in the same manner. We introduce the concept of resources that permit to model any kind of SoS example where resources are needed. Thus we describe the SApHESIA model used to represent a SoS, its components and the environment of the SoS.

3.1 Component-System Model

A component-system S_i that is the smallest part of a SoS (it represents the second S of SoS) is defined as: $S_i = \{F, R, G, L\}$ where:

- $F = \{F_1, ..., F_m\}$ is a set of **functionalities**;
- $R = \{R_1, ..., R_n\}$ is a set of **resources**;
- $G = \{G_1, ..., G_p\}$ is a set of **goals**;
- $L = \{L_1, ..., L_q\}$ is a set of **links** with others component-systems.

A **resource** i is a structure $R_i = \{type : String, quantity : Float\}$ representing passive elements in the SoS (i.e. which have no effector on the environment or on the SoS itself). A **functionality** is an effector on the environment or on the SoS itself which enables to give *operational independence* to the component-system. The functionality can affect the resources, the state and/or the links of a component-system. A functionality $\mathcal{F}$ is defined as a triplet : $\mathcal{F} : \{f, t, p\}$ where f is the function of $\mathcal{F}$ defined as: $f : Conditions \rightarrow Effects$; t is the execution time of $\mathcal{F}$ and $p \in [0, 1]$ is the performance of $\mathcal{F}$ (it represents the probability of $\mathcal{F}$ to succeed). $Conditions$ and $Effects$ can concern (i) a certain quantity of resources; (ii) the existence of a link between two component-systems and (iii) the existence of a component-system.

A **goal** is a special state that a component-system tries to reach and it enables to give the *managerial independence* to this component-system. Thus, a goal can be defined in two distinct ways :

- $G_R = \{type : String, value : Float, kind : \{=, \neq\}, p : Int\}$: a component-system tries to (un)equal a certain resource *type* to the value *value* with a priority p;
- $G_L = \{S_j\}$: a component-system tries to add a link with another component-system S_j.

A **link** is an oriented association between two component-systems permitting to represent the acquaintance between them and to share and exchange resources. As indicated in the functionality paragraph, links may be created and destructed by a functionality.

3.2 SoS Model

A SoS is defined as $SoS = \{\mathcal{S}, \mathcal{G}\}$ where $\mathcal{S}$ is a set of component-systems and $\mathcal{G}$ is a set of goals of the component-systems of $\mathcal{S}$. $\mathcal{G}$ represents the high-level goals of the SoS : $SoS = \{\{S_1, S_2, S_3\}, \{G_R\}\}$.

3.3 Environment Model

The SoS environment is the frame in which the SoS evolves and interacts with. It represents **entities** that do not belong to the SoS and **rules** (physical, economic, social...).

Formally, an environment is defined as $\mathcal{E} = \{E, Rules\}$ where E is a set of entities and $Rules$ is a set of rules.

Entity Model - An entity is an active independent object able to affect the environment or the SoS itself; it is not a part of the SoS.

$E_i = \{F, R, G, L\}$ where:

- $F = \{F_1, ..., F_m\}$ is a set of **functionalities**;
- $R = \{R_1, ..., R_n\}$ is a set of **resources**;
- $G = \{G_1, ..., G_p\}$ is a set of **goals**;

- $L = \{L_1, ..., L_q\}$ is a set of **links** with entities or component-systems.

It is important to notice that an entity can be linked to a component-system or to another entity.

Rule Model - A rule represents the frame in which the SoS evolves. It permits to model how the environment reacts while interacting with the SoS. A rule needs conditions to be fulfilled to apply effects and can affect all the entities in the environment or all component-systems in the SoS. A rule is like an "omniscient" functionality.

$Rule = \{Conditions \rightarrow Effects\}$.

With all these elements, SApHESIA model is generic and expressive enough to model a large variety of problems (economical, transport,...). Moreover, it is easily computable and its genericity permits to focus on generic architecting problems such as dynamic evolution of interactions between component-systems and emergence.

4 Cooperative Agent Decision

To present a SoS architecting methodology based on cooperation, we propose a decentralized decision algorithm using the AMAS (Adaptive Multi-Agent System) approach. In this approach, each agent has to decide its next action by taking into account the instant difficulty (criticality) of its neighborhood.

4.1 The Criticality : Metric of Cooperation

[7] presents a generic multi-agent evaluation metric in order to know the criticality an agent is faced with. More precisely, this metric represents the distance between the current state of an agent and the final state it tries to reach. Basically, each agent tries to minimize both its own criticality and the criticality of its neighbors. We integrate this metric in SApHESIA to have a cooperative decision algorithm for architecting SoS. Thus, each component-system is agentified and this metric is translated using resources and goals. Indeed, the current state of a component-system can be represented by its resources and the state to reach (its goals). To be able to compare its own criticality with the criticality of other component-systems, each component-system calculates its criticality with the same function C_{S_i} defined as:

$$C_{S_i}(t) = \frac{\sum_{G_j \in G_i}(C_{G_j}(t) \star G_j.priority)}{\sum_{G_j \in G_i}(G_j.priority)}$$

with $S_i = \{F_i, R_i, G_i, L_i\}$ and $C_{G_j}(t)$ is the criticality of the goal G_j at time t. This one is calculated with the following sigmoid functions:

$$C_{G_j}(t) = \begin{cases} 1 + \frac{1}{e^{\Delta_{G_j}(t)+a}} - \frac{1}{e^{\Delta_{G_j}(t)-a}} & \text{if } G_j.kind = EQ \\ -\frac{1}{e^{\Delta_{G_j}(t)+a}} + \frac{1}{e^{\Delta_{G_j}(t)-a}} & \text{if } G_j.kind = NEQ \end{cases}$$

with $\Delta_{G_j}(t) = G_j.value - S_i.R_i(G_j.type)(t)$. $S_i.R_i(G_j.type)(t)$ is the amount of resource $R_i(G_j.type)$ at time t.

It is important to notice that criticality is always between 0 and 1. So, if agents have different priority scales on goals, each agent has the same importance in term of criticality. Thus an agent cannot always become more critical than the others because of goal priorities.

4.2 Cooperative Agent Algorithm

This algorithm is based on criticality comparison between agents. Basically, each agent compares its criticality with its neighborhood for each of its available actions. Then, it chooses the action that leads to the minimum of the maximum of the criticality of its neighborhood. To do that, an agent A_1 computes its own anticipated criticality as well as the anticipated criticality of its neighborhood until a final time tf (corresponding to the time it cannot use its action anymore). Then, A_1 computes a set of *comparable actions* (called $F_{10\%}$ in algorithm 1) in order to eventually find an action leading to a similar result in a quicker time. This behavior tends to minimize the maximum of neighborhood criticality: each agent "helps" its neighborhood by choosing the action that, in the worst case, provokes the minimum raise of criticality. More details about the cooperative decision of component-systems of a SoS are given in algorithms 1 and 2 where the component-system (respect. functionality) corresponds to an agent (respect. an action).

Let's take $S = \{S_1, ..., S_n\} \mid n \in \mathbb{N}\}$ where: $\forall i \in n$, $S_i = \{F_i, R_i, G_i, L_i\}$ and $\forall i \in n$, $C_{S_i}(t)$ is the criticality of S_i at time t.

forall the $f \in F_i$ **do**
  $\Delta_f \longleftarrow \emptyset$;
  $t_f \longleftarrow calculateFinalTime(f)$;
  forall the $S_j \in L_i$ **do**
    $C'_{S_j}(t) \longleftarrow calculateAnticipatedCrit(S_j, f)$;
    $\Delta_f S_j \longleftarrow C'_{S_i}(t_f) - C'_{S_j}(t_f)$ *Calculate diff of criticality for neighbors*;
    $\Delta_f \longleftarrow \Delta_f \cup \Delta_f S_j$;
  end
end
Let's define $best_f \in F$ such as $\min_{g \in F_i}(\max_{S_j \in S}(\Delta_g S_j)) \in \Delta_{best_f}$;
$min\Delta \longleftarrow \min_{g \in F_i}(\max_{S_j \in S}(\Delta_g S_j))$ *Choose f that minimize the max of criticality*;
$F_{10\%} \longleftarrow \{g \in F \mid \max_{S_j \in S}(\Delta_g S_j) \pm 10\% \times min\Delta\}$;
forall the $g \in F_{10\%}$ **do**
  if $t_g << t_f$ **then**
    $best_f \longleftarrow g$
    $t_f \longleftarrow t_g$
  end
end

Algorithm 1: Cooperative component-system S_i decision

The function $Effect$ (not described here) returns a delta representing how the application of f will influence S_j. The *calculateAnticipatedCrit* procedure returns a linear approximation of the anticipated criticality of S_j if f is applied by S_j until time t. In this manner, a component-system is able to approximate the functionality influence on its neighborhood. It is important to notice that the component-systems do not need to exchange a lot of information as they only need to know their criticality and the one of their neighborhood to take their decisions.

/*Find a and b, such as $C'_{S_j}(t) = a \times t + b$*/
calculateAnticipatedCrit(System S_j, Functionality f):
$b \longleftarrow C_{S_j}(t-1)$;
$t_0 \longleftarrow t-1$;
if $\neg$ *hasEffect*(S_j, f) **then**
  $a \longleftarrow C_{S_j}(t) - C_{S_j}(t-1)$;
end
else
  $a \longleftarrow (C_{S_j}(t) + Effect(S_j, f)) - C_{S_j}(t-1)$;
end
return $C'_{S_j}(t) = a \times (t - t_0) + b$

Algorithm 2: Anticipated criticality of S_j when applying f until t.

5 Case Study: A Box Transportation System

CoCaRo is an AMAS used to model and simulate a carrying system of colored boxes by robots. More precisely a robot has to find and catch a box and to deposit it in the nest having the same color as the box. Each robot has an initial amount of energy that it consumes at each movement. However when a robot deposits a box, it receives a reward in the form of energy allowing it to remain longer alive. The value of the reward depends on the color of the deposited box compared to the color of the robot; i.e. a robot will get a better reward if it deposits a box of its own color.

The **environment** is composed of three different object types: (*i*) The *Displacement Grid* that is an object of the plan and is made of 50x50 square cells. A cell may be empty or may contain a robot, a box and/or a nest; (*ii*) the *Nest* that is an object in which robots deposit boxes. The grid contains three nests that are blue, red and green and that are equidistant from each other in order to prevent bias related to the proximity of 2 nests into simulations and (*iii*) the *Boxes* that may be carried and deposited into nests by robots. A box may be red, blue or green and appears randomly on the grid, at regular time interval.

The **robot agent** may perform one of the following *actions*: *move*, *deposit*, *take* and *go*. The robot agent moves on the grid according to a Monte Carlo distribution until it finds a box. Moreover, an agent has four *states* : *carried* (the agent is carrying a box), *target* (the agent has targeted a box), *onPosBox* (the agent is on the same cell as its target box) and *onPosNest* (the agent is on the same cell as the nest corresponding to its carried box). The agent chooses the most appropriate action according to its current state and its perceptions:

- if it perceives *target* then it *go*(*targeted_box*);
- if it perceives $target \wedge onPosBox$ then it *take*(*targeted_box*);
- if it perceives $carried \wedge onPosNest$ then it *deposit*(*carried_box*);
- if it perceives $\neg target \wedge \neg carried$ then it *move*;

The **criticality** indicates the level of difficulty of a robot agent and is defined in terms of its energy level. The effectiveness of an agent is related to its energy level and may rapidly deteriorate. In this context, the criticality C_{r_i} of a robot agent r_i is a temporal function calculated as follows, where $Ne_{r_i}(t)$ is the battery level of agent r_i at time t and Max_{Ne} is the maximum battery level:

$$C_{r_i}(t) = Max_{Ne} - Ne_{r_i}(t)$$

The criticality is thus an integer ranging from 0 to Max_{Ne} (agents having a big criticality).

The **anticipated criticality** is a function allowing the robot agent to know the criticality it is going to get once the box b_k deposited in the nest. It enables the robot agent to choose the box that will offer it most energy. The anticipated criticality of robot agent r_i for the box b_k is calculated as follows: $CA_{r_i}(b_k, t) = Max_{Ne} - Ne_a(b_k, t)$ with

$$Ne_a(b_j, t) = \begin{cases} N_{r_i}(t + t_d) + rec_{r_i}(b_k) & \text{if } 0 < . < Max_{Ne} \\ Max_{Ne} & \text{if } . \geqslant Max_{Ne} \\ 0 & \text{else} \end{cases}$$

t_d is the time taken by the agent r_i to go from its current position to the box b_k and then to the nest : $t_d = \frac{distance(r_i,b_k)+distance(b_k,Nest)}{Speed_{r_i}(t)}$ $Speed_{r_i}(t)$ is a function representing the speed of agent r_i. This speed decreases with $Ne_{r_i}(t)$. It permits to represent that a robot is less efficient when its battery is low. Initially, the case study of CoCaRo has to show the usefulness of the cooperation algorithm we propose. To this aim two systems have been implemented.

5.1 System 1 : Non cooperative Agents

In this system, robot agents do not use any cooperative mechanisms. An agent is looking for the most interesting box for it in order to have a maximum amount of energy and therefore to transport a maximum of boxes. Agents do not try to exchange boxes even if it could be advantageous for them.

Decision Algorithm - To achieve its goal (to maximize its energy level), a robot agent applies a decision algorithm using its current criticality and its anticipated criticality. While returning to the nest an agent may change its carried box if it finds a box enabling it to get more energy than it would have with the one it carries. This algorithm is described in algorithm 3 by reading only the black part.

5.2 System 2 : Cooperative Agents

In this system, robot agents are cooperative; they can exchange boxes between them through message sending according to their current and anticipated criticalities.

Decision Algorithm - When a robot agent r_i is detecting a box b_k enabling it to get more energy, r_i is checking if the box is already owned by a robot agent r_j. If it is the case, the agent r_i is sending a message to r_j which is containing the criticality and the anticipated criticality of r_i for the box b_k. These information required by the cooperative mechanism make the agent r_j to choose either to exchange the box b_k (if r_j does not become too critical because of the exchange), or to keep it (if r_j becomes too critical because of the exchange). The algorithm 3 (with red part) shows the decision process.

Cooperative Mechanism - The cooperation mechanism of a robot agent is implemented in the function *Cooperative_request_process* invoked in the algorithm 3 and detailed in the algorithm 4.

The agent r_j carrying the box b_k considers at each time step, the cooperative request received at $t - 1$. As it is cooperative, the agent r_j has to determine if the request sender r_i is more critical than it, in which case r_j has to give the box to r_i. For that, r_j compares its anticipated criticality with the one of r_i and then checks if the exchange will not cause a criticality increase on the long term.

```
while N_e(t) ≠ 0 do
    if carried then
        CA_current = CA_{r_i}(carried_box, t)
    end
    if target then
        CA_current = CA_{r_i}(targeted_box, t)
    end
    Update visible_boxes ;
    forall the b_k ∈ visible_boxes do
        temp_C_a := CA_{r_i}(b_k, t) ;
        if temp_C_a < CA_current then
            if holder(b_k) ≠ null then
                SendCoopReq(r_j, C_{r_i}(t), CA_{r_i}(b_k, t))
            end
            else
                CA_current := temp_C_a ;
                targeted_box := b_k ;
                if carried then
                    deposit(carried_box) ;
                end
            end
        end
    end
end
coop_request_processing();
```

Algorithm 3: Decision for a cooperative robot agent r_i

```
if C_{r_j}(t) < C_{r_i}(t) then
    if CAr_i(b_k, t) < C_{r_i}(t) then
        accept_exchange();
        deposit(b_k);
    end
    else
        refuse_exchange() ;
    end
end
else
    if CAr_j(b_k, t) < C_{r_j}(t) then
        refuse_exchange() ;
    end
    sinon
        accept_exchange() ;
        deposit(b_k);
    fin
end
```

Algorithm 4: coop_request_processing() of agent r_j carrying b_k

5.3 System 3 : Instantiation of SApHESIA model

To validate our architecting cooperative approach for SoS presented in section 4 and to study inter-AMAS cooperation, we decide to instantiate CoCaRo as a SoS by using SApHESIA model. First, it seems natural that each group of robots of the same color of the AMAS CoCaRo can be seen as a "sub"-AMAS and may be reified as component-system (red, green and blue) of a SoS: each of them can be modeled with goals, functionalities, resources, links and then criticality. For instance, the goal of a component-system is to avoid robots with empty battery. Resources can be robots themselves. Functionalities can be actions of robots, etc. To enable cooperation, resources and functionalities are added to component-systems to compute a representative criticality and then help other component-systems. To design functionalities, a component-system may help another one by simply giving the right boxes to others. To do that, each component has the possibility to change perceptions of its robots concerning box rewards. We illustrate our reasoning by considering the red component-system S_r. S_r has the functionalities to change the perceptions about blue and green boxes for red robots. Thus, red robots may consider these boxes as much important as red ones for themselves. In this way, red robots may take blue and/or green boxes and may give them to other robots. At the SoS level, it may be seen as a way for red component-system to help green and/or blue ones when they are more critical. Thereafter are the details of the model for a red component-system defined as in section 3.1,
$S_r = \{F_r, G_r, R_r, L_r\}$ with: $R_r = \{robot, robot.color = "red"\}$;
$R_{Dying} = \{robot, robot.battery < Max_{N_e}/3\}$;
$F_{helpblue} = \{R_{blue} \rightarrow \{robot.rec(blue) \leftarrow robot.rec(red)\}\}$;
$F_{helpgreen} = \{R_{green} \rightarrow \{robot.rec(green) \leftarrow robot.rec(red)\}\}$;
$F_{contact} = \{R_{Dying} > R_r/2 \rightarrow R_{red}To\{S_b, S_g\}\}$;
$G_r = R_{Dying} = 0$, $L_r = \{S_b, S_g\}$.

6 Experimentation

The three previously presented systems have been implemented using GAMA [10]. GAMA is a platform to model and simulate large-scale multi-agent systems, easy to use, which integrates analysis and performance visualization tools. Each system is then assessed in terms of efficiency and robustness using three metrics representing the state of the system over time: the number of functional robots, the average energy level of the agents and the number of boxes present in the environment.

6.1 Description

Simulations for the 3 systems have the same initial conditions: (i) the number of robots is set to 90 (30 of each color); (ii) the initial energy level (N_i) is set to 300; (iii) the maximum energy level (Max_{N_e}) is set to 300; (iv) the energy consumption per time ($conso$) is set to 1; (v) perception and communication scopes of a robot are set to a radius of 3 squares around its current position; (vi) the number of boxes appearing in the environment is set to 1 every 3 time units and (vii) the initial placement of boxes and robots is the same for all simulations.

$F_{helpblue}$ and $F_{helpgreen}$ (in system 3) permit to change the perceptions about blue and green boxes for red robots. As shown in formal definition of S_r, these functionalities can be activated only when another system (S_b or S_g) will use $F_{contact}$. It happens only when at least the half of the system robots have their battery under $max/3$. The resource R_{Dying} represents the number

of robots with a low battery. Then the goal G_r represents the fact that each component-system tries to avoid low battery robot. Finally, to decide if a component-system will help others, it simply computes its criticality thanks to goal G_r (cf. section 4.1) and uses the algorithm 1 to take its decision.

6.2 Results and Discussion

In fig.1 the black curves represent the system without cooperation, the light gray color curves represent the system with cooperation and the dark gray curves represent the SoS according to SApHESIA. The top-left curves present the mean battery level of robots. The top-right curves present the number of boxes in the environment. Finally, the bottom curves present the number of alive robots in the environment. The comparison of the three systems clearly shows that the exchange of criticality improves the systems efficiency. Indeed, the underlying mechanism of cooperation with this exchange allows a greater number of agents to survive. Consequently the cooperative system covers a larger part of the grid thus improving the system's ability to detect boxes. In a second time, the use of our cooperation process for SoS enables cooperation at a higher level as the reification of each group of robots as a component-system permits to improve global performance (number of boxes, battery level and number of alive robots). These results legitimate the study of cooperation in SoS and are encouraging concerning the inter-AMAS cooperation.

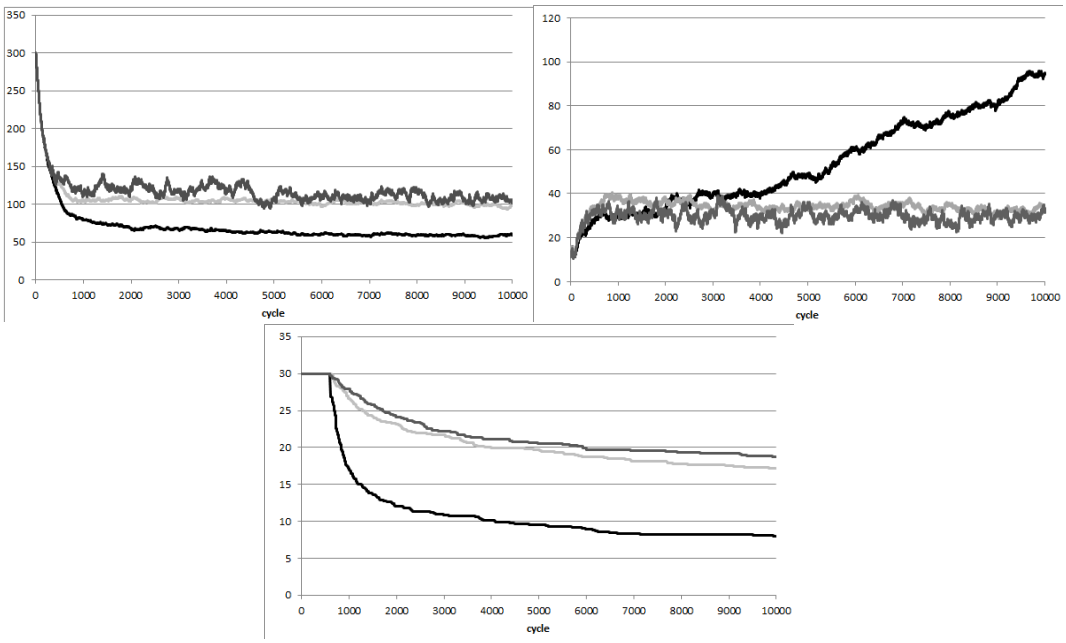

Figure 1: Mean battery of robots, Boxes number, Number of alive red robots

7 Conclusion and perspectives

The aim of this paper was to show (*i*) the contribution of the criticality as a social attitude of an agent in terms of cooperation and (*ii*) the contribution of SoS modeling with SApHESIA.

First, criticality permits an agent to choose its action in a cooperative way. Cooperation is a social attitude of each agent, which leads to a global increase of the system performance comparing to selfish agents. Then, to study the benefit of SoS in this kind of problems, we instantiate a new model called SApHESIA and show that performance of the system is better than the one with simple cooperation. Indeed, SoS approach enables a "macro-cooperation" between systems and leads to better anticipate and react to problems that may appear in the environment. Concerning futures works, the proposed architecture will be evaluated in more problem domains to show its generality. Then, energy and time consumption of communication, perception and decision making will be taken into account in our simulations to be closer to the reality in our case studies.

References

[1] P. Acheson, L. Pape, C.H. Dagli, N. Kilicay-Ergin, J. Columbi, and K. Haris. Understanding system of systems development using an agent- based wave model. *Procedia Computer Science*, 12:21 – 30, 2012. Complex Adaptive Systems 2012.

[2] S. Agarwal, L.E. Pape, N. Kilicay-Ergin, and C.H. Dagli. Multi-agent Based Architecture for Acknowledged System of Systems. *Procedia Computer Science*, 28:1–10, 2014.

[3] C. Azani. *An Open Systems Approach to System of Systems Engineering*, pages 21–43. John Wiley and Sons, Inc., 2008.

[4] W.C. Baldwin and B. Sauser. Modeling the characteristics of system of systems. *2009 IEEE International Conference on System of Systems Engineering (SoSE)*, 2009.

[5] M. Bjelkemyr, D. Semere, and B. Lindberg. An engineering systems perspective on system of systems methodology. In *Systems Conference, 2007 1st Annual IEEE*, pages 1–7, April 2007.

[6] E. Bonabeau. Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl. 3):7280–7287, 2002.

[7] T. Bouziat, S. Combettes, V. Camps, and P. Glize. La criticite comme moteur de la cooperation dans les systemes multi-agents adaptatifs. In *Journees Francophones sur les Systemes Multi-Agents (JFSMA)*, pages 149–158. Cepadues Editions, 2014.

[8] D.S. Caffall and J.B. Michael. System of Systems Collaborative Formation. *Systems Journal*, 3(3):385–401, 2009.

[9] V. Camps. *Vers une théorie de l'auto-organisation dans les systèmes multi-agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie*. PhD thesis, Univ. Paul Sabatier, Toulouse, 1998.

[10] A. Drogoul, E. Amouroux, P. Caillou, B. Gaudou, A. Grignard, N. Marilleau, P. Taillandier, M. Vavasseur, D. A. Vo, and J.D. Zucker. GAMA: multi-level and complex environment for agent-based models and simulations. In *AAMAS*, pages 1361–1362, Saint-Paul, MN, USA, 2013.

[11] D. Firesmith. Profiling systems using the defining characteristics of systems of systems (sos). Technical Report CMU/SEI-2010-TN-001, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2010.

[12] M.-P. Gleizes, V. Camps, J.-P. Georgé, and D. Capera. Engineering Systems which Generate Emergent Functionalities. In Danny Weyns, Sven Brueckner, and Yves Demazeau, editors, *Engineering Environment-Mediated Multiagent Systems - Satellite Conference held at The European Conference on Complex Systems (EEMMAS), Dresden, Germany, 01/10/2007-05/10/2007*, number 5049 in Lecture Notes in Artificial Intelligence (LNAI), http://www.springerlink.com/, juillet 2008.

[13] M. Henshaw, C. Siemieniuch, M. Sinclair, V. Barot, S. Henson, C. Ncube, S. Lim, H. Dogan, M. Jamshidi, and D. Delaurentis. The Systems of Systems Engineering Strategic Research Agenda Systems of Systems Engineering. (2), 2013.

[14] M. Jamshidi. System of systems engineering - new challenges for the 21st century. *IEEE Aerospace and Electronic Systems Magazine*, 23(5):4–19, May 2008.

[15] S. Lemouzy. *Systèmes interactifs auto-adaptatifs par systèmes multi-agents auto-organisateurs : application à la personnalisation de l'accès à l'information.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, juillet 2011. (Soutenance le 13/07/2011).

[16] M. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284, 1998.

[17] S. Selberg and M. Austin. Toward an evolutionary system of systems architecture. *18th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2008*, 4(1):2394–2407, 2008.

[18] W.C. Stirling and R.L. Frost. Social utility functions-part ii: applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):533–543, 2005.