An approach for the integration of swarm intelligence in MAS: an engineering perspective

Vincent Hilaire[a], Massimo Cossentino[b], Franck Gechter[a], Sebastian Rodriguez[a,c], Abderaffiaa Koukam[a]

[a]*laboratoire Systmes et Transports, UTBM, 90010 Belfort Cedex FRANCE*
[b]*ICAR Institute,National Research Council, Palermo, ITALY*
[c]*Centro de Investigación de Teconologías Avanzadas de Tucumán, UTN, San Miguel de Tucuman, ARGENTINA*

Abstract

For more than 20 years, researchers have designed models in order to describe swarm intelligence and apply the resulting techniques to complex problems. However, there is still a gap between these models and current MAS methodologies. The goal of this paper is to propose a principled and methodological approach for the engineering of systems based upon swarm intelligence. The constraints are, on the one hand, to enable the analysis, design and implementation of such systems; and, on the other hand, to formally analyze and verify properties of resulting systems. The principles of the approach are based, on the one hand, on requirement driven activities that produce goals to be fulfilled by the system of interest and, on the other, hand on an ontological modeling of the problem domain. This ontological modeling conceptualizes the phenomenon one seek to imitate and thus allows it understanding. The produced ontology is refined through the methodology activities down to organizational models.

1. Introduction

For more than 20 years, researchers have designed models in order to describe swarm intelligence and apply the resulting techniques to complex problems. However, there is still a gap between these models and current MAS methodologies such as ASPECS [1], TROPOS [3] or GAIA [38].

The goal of this paper is to propose a principled and methodological approach for the engineering of systems based upon swarm intelligence. The constraints are, on the one hand, to enable the analysis, design and implementation of such systems; and, on the other hand, to formally analyze and verify properties of resulting systems.

Some works have contributed to this issue [6, 7, 8]. In [7] the authors build a library of patterns of self-organizing behaviors. However, even if it is an interdisciplinary and interesting attempt to define a framework for engineering systems based upon swarm intelligence a lot of work has still to be done in order to define principled and methodological approaches relying on swarm intelligence.

The author of [6] proposes general rules in order to engineer swarming systems and a formal framework for analysis and validation. These rules are not related to current methodologies or development platforms. In [8] the authors design a complete methodology based upon the theory of Adaptive Multi-Agent Systems. This methodology allows the analysis and design of swarming systems but impose a cooperative internal medium which maybe a strong constraint for some cases.
Swarming approaches are usually inspired by biology [9] or physics [10, 11]. These systems usually exhibit features such as self-organization, emerging phenomena, robustness and adaptability. One of the main problems with this kind of approaches is the small number of methodologies and guidelines which help the engineering of such systems. Indeed, to be fully adopted, methodologies must be able to decompose the underlying principles of these systems with abstractions in a principled way. Some experiments were done in the domain of swarming MAS architecture reverse-engineering [12] but they usually lack a systematic support. For instance, in [5] the authors give general principles in an informal way. However, it is a real problem and despite the interest risen by these architectures the claims that stripping away centralized control is enough to allow the emergence of interesting properties has never been proved [13].
In order to engineer such kind of MAS we propose to use a combined approach based, on the one hand, on requirement driven activities that produce goals to be fulfilled by the system of interest and, on the other, hand on an ontological modeling of the problem domain. The principle is to produce an ontology which conceptualizes the phenomenon one seek to imitate and thus allows it understanding. Indeed, swarming approaches [4] are based on the replication of existing behaviors, that have produced in certain experimental conditions the desired emerging properties. This statement was already observed in [5]:

> Swarming is a discovery, not an invention. It is a naturally occurring phenomenon that we seek to imitate in engineered systems. Design principles for effective artificial swarming systems must be developed from an understanding of why swarming works in natural systems.

The produced ontology is then refined down to organizational models. The analyst can then choose from a library of organizational models the ones that will satisfy the goals issued from the requirements analysis. This combined approach is associated with formal proofs techniques in order to formally verify properties of organizational models and fulfillment of goals.

It seems a sound principle as the ontology is an understanding of the modeled system and so describes at least partially how it works. The interacting entities the designer wants to replicate and their possible actions are part of this ontology. In order to introduce a systematic activity, some guidelines are provided in order to identify organizations and roles from this ontology. This activity is integrated in an existing methodology, namely ASPECS. The modified methodology is then able to decompose such systems in roles, interactions

and organizations which are the key concepts of the ASPECS analysis phase and which are identified, in the original methodology, by means of use cases. This kind of approach is advocated by the authors of [5] who state "we ought to be able to reverse-engineer the underlying mechanisms of swarming systems for use in synthetic systems".
The paper is organized as follows: section 2 presents the background related to the ASPECS methodology and the specific architecture that illustrates the presented approach. Section 3 presents the ontological identification approach. Section 4 is dedicated to related works and section 5 concludes.

2. Background

2.1. ASPECS

ASPECS is a step-by-step requirements to code software engineering process based on a metamodel which defines the main concepts for the proposed MAS and Holonic MAS analysis, design and development. It integrates design models and philosophies from both object- and agent-oriented software engineering (OOSE and AOSE) and is largely inspired by the PASSI [2] and RIO [14] approaches. The target scope of ASPECS can be found in complex systems and especially hierarchical complex systems. The main vocation of ASPECS is towards the development of societies of holonic (as well as not-holonic) multiagent systems.

The ideas underpinning the ASPECS design process can be described as follows:

1. The ASPECS design process explicitly deals with the design of open, dynamic and complex systems.
2. The adoption of an organizational approach. Functionalities to be realized are assigned to organizations. An organization is defined by a collection of roles that take part in systematic institutionalized patterns of interactions with other roles in a common context. A role is defined as an expected behavior (a set of role tasks ordered by a plan) and a set of rights and obligations in the organization context. The goal of each Role is to contribute to the fulfillment of (a part of) the requirements of the organization within which it is defined. A role can be instantiated either as a Common Role or Boundary Role. A Common Role is a role located inside the designed system and interacting with either Common or Boundary Roles. A Boundary Role is a role located at the boundary between the system and its outside and it is responsible for interactions happening at this border (i.e. GUI, Database, etc).
3. Domain related ontological knowledge is used as a tool for enhancing the quality of design. This has been already adopted in some previous methodologies [15] but it is lacking in most modern approaches. We think that in dealing with intelligent agents it is particularly important to explicitly catch an ontological model of the problem and solution domains; this allows an easy application of several AI techniques as well as the adoption of semantic-based communications among agents.

4. Three main levels of abstractions, called models according to the model-driven engineering terminology, are considered. Concepts of the problem domain are used to model system requirements in terms of organizations and interacting roles; concepts of the agency domain are the result of a set of transformations from the previous domain and are used to depict an agent-oriented solution; concepts of the solution domain are again the result of some transformations and are devoted to design a platform-specific solution at the code level.

The different activities of the System Requirements phase of ASPECS are represented by the SPEM diagram of figure 1. ASPECS software process is driven by requirements. Thus the first activity, Domain Requirement Description, of the first phase of ASPECS, System Requirements, deals with the analysis of system functional and non functional requirements. Functional requirements describe the functions the software has to exhibit [16] or the behavior of the system in terms of interactions perceived by the user. Non functional requirements are sometimes known as constraints or quality requirements [16]. The global objective of the Domain Requirements Description (DRD) activity is gathering needs and expectations of application stakeholders and providing a complete description of the behavior of the application to be developed. In the proposed approach, these requirements should be described by using the specific language of the application domain and an user perspective. This is usually done by adopting use case diagrams for the description of functional requirements; besides, conventional text annotations are applied to use cases documentation for describing non-functional requirements.
The global objective of the Problem Ontology Description (POD) is to provide an overview of the problem domain. Stakeholders naturally express requirements in their own terms and with implicit knowledge of their own works [17]. Therefore the aim of this activity is deepening the understanding of the problem by complementing the usual requirements description in terms of use cases with a description of the concepts that compose the problem domain. It describes concepts used in the specific language of the application domain and users. Results of this activity can sometime imply modifications in uses cases. The design of the domain ontology occurs very earlier in our process and this has a direct consequence in the organization and capacity identification activities. Problem ontology is modeled by using a class diagram where concepts, predicates and actions are identified by specific stereotypes. The ontology is inspired from the FIPA proposal [18]. The main point is that actions are distinguished concepts associated to the concept that act and the concept that is manipulated. This specific type of ontology is described in the next subsection.
The DRD and POD activities precedes Organization Identification. In this activity the objective is to assign to each use case an organization in charge of its fulfillment. Once the organizations have been identified the next activity, Interaction and Role Identification consists in refining organizations in terms of interacting roles. Roles use their capacities for participating to organizational goals fulfillment; a Capacity is a specification of a transformation of a part of the

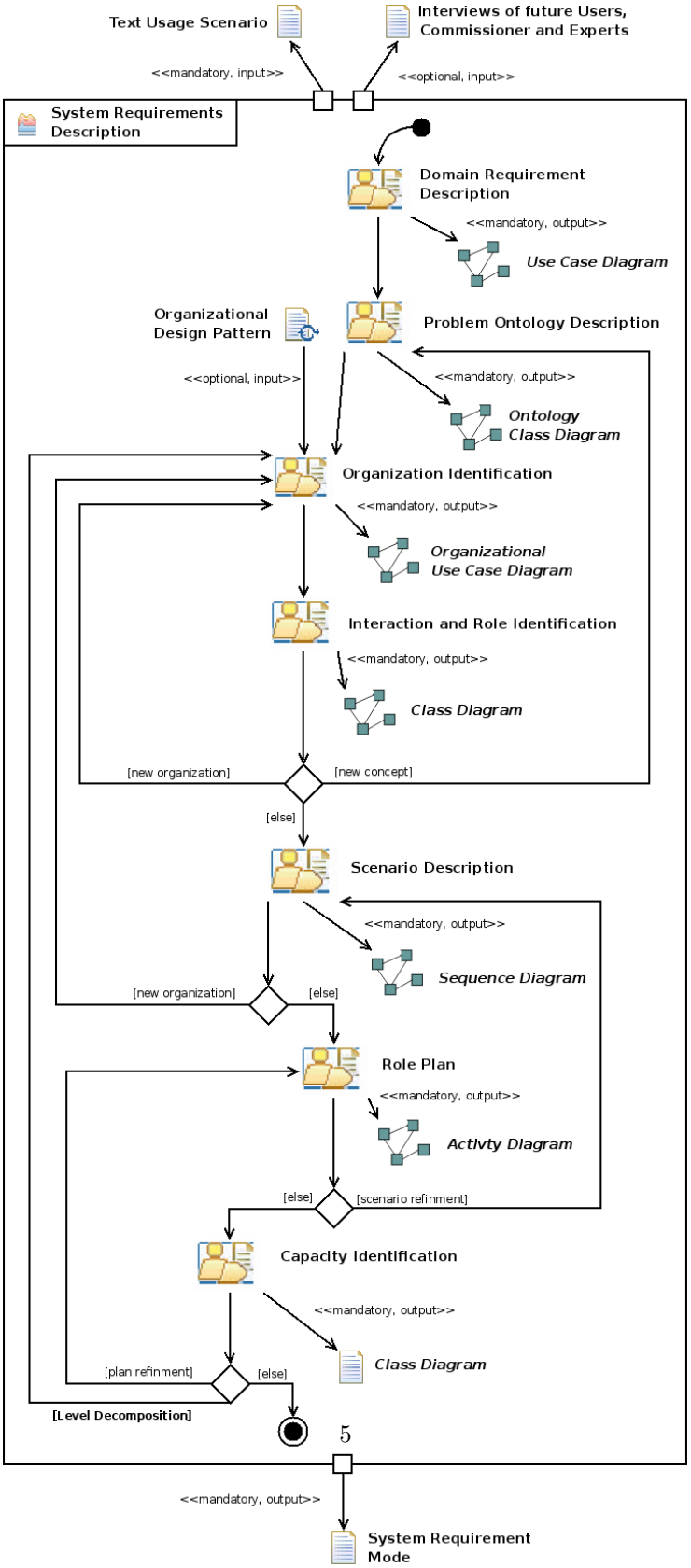

Figure 1: ASPECS System Requirements phase

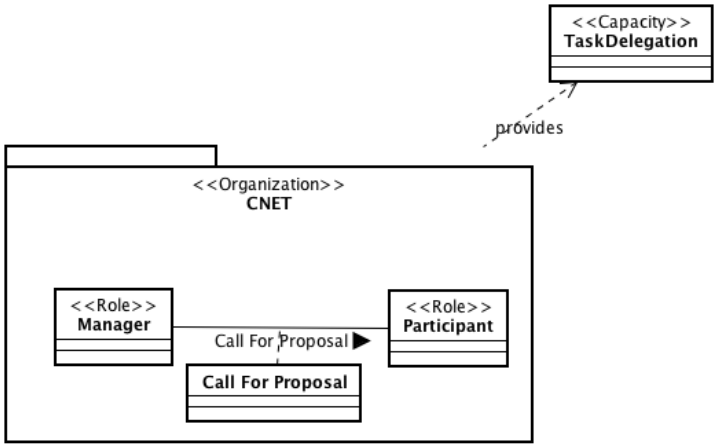

Figure 2: ASPECS example of organization

designed system or its environment. This transformation guarantees resulting properties if the system satisfies a set of constraints before the transformation. It may be considered as a specification of the pre- and post-conditions of a goal achievement. This concept is a high level abstraction that proved to be very useful for modeling a portion of the system capabilities without making any assumption about their implementations as it should be at the initial analysis stage.

A Capacity describes what a behavior is able to do or what a behavior may require to be defined. As a consequence, there are two main ways of using this concept:

- it can specify the result of some role interactions, and consequently the results that an organization as a whole may achieve with its behavior. In this sense, it is possible to say that an organization may exhibit a capacity.
- capacities may also be used to decompose complex role behaviors by abstracting and externalizing a part of their tasks into capacities (for instance by delegating these tasks to other roles). In this case the capacity may be considered as a behavioral building block that increases modularity and reusability.

Figure 2 shows an example of organization drawing from the well-known Contract NET protocol [19]. This organization is sketched by the CNET package stereotyped Organization. Within this organization there are two roles depicted by classes stereotyped Role. These two roles are Manager and Participant. The association class between roles depicts an interaction. Here it is the Call for Proposal aiming to delegate a task that starts the CNET protocol. An example of provided capacity is included in the diagram. This capacity,

namely TaskDelegation, represents the fact that the CNET organization is able to manage the delegation of a task. the requires part of this capacity includes the fact there exist such a task and that there is participants. The ensures part of the capacity states that the task will be delegated to the participant with the best bid

For a complete view of ASPECS the reader can see [1].

2.2. Ontology definition

A definition of ontology is given by the author of [20] : in the context of knowledge sharing, an ontology means a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. There are several ways for describing an ontology such as: semantic networks, concept lattices and logic for example. A frequently used ontology is OWL advocated by the W3C [21]. It is an object-oriented language based on class and properties. For clarity reasons, the examples of ontologies will be presented as UML class diagrams in this paper.
The ontology we use for this paper is inspired by the FIPA RDF content specifications [18] and the one used in [2]. An UML class diagram is used to represent the ontology, as it is the case in other works such as [22, 23]. In order to distinguish the different ontology categories, the classes representing them are stereotyped by their category name. These categories are: concept, predicate and action. A concept describes a set of individuals or instances of the domain of interest. A predicate is an assertion on properties about concepts. An action is something that is performed in the domain of interest. These classes have relationships between them. Indeed, a predicate is related to the concerned concept and an action is related to an actor (the concept that acts) and a receiver (the concept that is acted on). To these existing categories the Capacity, as described in previous subsection, is added. A capacity is described by a set of properties required (pre-conditions), a set of properties ensured (post-conditions) and two sets that are the inputs and outputs of the capacity. An example of such ontology is given in figure 3. In this example, there are two concepts: Seller and Buyer and one action: Sell. The principle of the action is that a Seller sells to a Buyer. The predicate hasProductsToSell establishes whether or not a Seller has some products to sell. Eventually, the capacity TriggerPayment of a Buyer states that a Buyer must be able to pay.

2.3. Localization and Tracking MAS

2.3.1. Context

Localization, with mobile or fixed sensors, is a very difficult but required task to control mobile robots in an indoor dynamic and uncertain environment. This task can be defined as finding the position of an object, mobile or not, in a well known referential system. The localization problem may adopt two methods: localization with on board sensors (also called self localization) and localization with external sensors. The algorithms used generally stem from

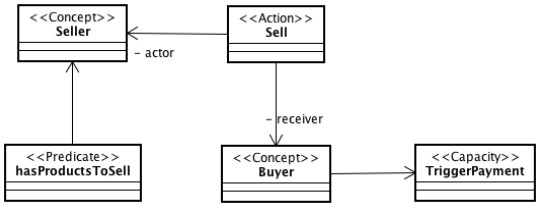

Figure 3: Example of ontology diagram

signal or image processing, or from the stochastic methods based on Markov Models [24]. So, the standard localization algorithms are extremely dependent on the nature of the used sensors and deal generally only with one single target. The existing multi-agent based localization and tracking devices are generally based on specialized cognitive agents and are closely tied to specific task [25]. In this way, tracking is considered to be a collection of temporally and spatially coherent localizations. As a means of localization, the tracking algorithms stem from the signal processing. Among the most spread out we can point out the Kalman filter, the optical flow algorithms and the particle filtering [26]. The main difficulty in designing such systems for localization and tracking is to take into account the characteristics of the used sensors while obtaining properties such as robustness and adaptation to the variation in the targets' kinetics. Considering these required properties, using a reactive multi-agent system to solve this problem seems to be adapted.
The approach presented in this paper is based on the fact that the environment can also be considered as an active element in MAS problem solving processes. Indeed, it has been emphasize in litterature [27], [4] that it can plays an important role in MAS especially when agents have limited abilities. As exposed in [28], in the presented approach, the environment can be considered as the interface between the real world where the problem evolves (i.e. where the targets appear, move and disappear in our case) and the resolution process, composed of agents and their interactions, which computes the solution (i.e. localization and tracking of targets). This principle has already been used practically in target tracking and localization [29] but also in relation with intelligent mobile vehicle abilities such as obstacle avoidance [30] and driving assistance [31].

2.3.2. Approach

As previously exposed, localization and tracking are based on the use of sensors that are spread out in the target evolution area. The association sensor/processing algorithms is then called Perceptive Unit (PU). The environment is depicted using an *occupancy grid* that represents an abstraction of the observable areas of the real world according to the sensors' range. Dynamics of the problem depend on dynamics of the targets, which can (i) *appear*, i.e. they arrive in the observation field of the sensors, (ii) *move*, i.e. they go from one observable point of the real world to another observable point, (iii) *disappear*,

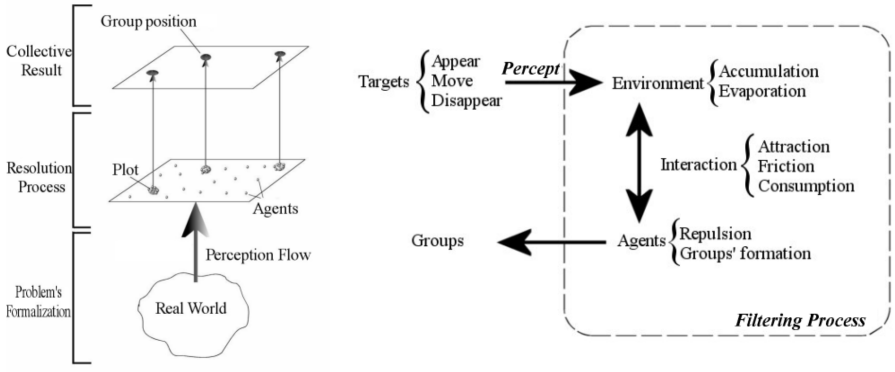

Figure 4: Architecture of the Physics based reactive model for the localization and the tracking (left) and representation of the solving process as a filter (right).

i.e. they go out of the observation field. These dynamics have been accounted for using two main trends. First, *accumulation* of the sensing information deals with the appearance of the targets. This accumulation leads to the construction of a plot(namely, a local probability distribution) that represents a possible position for a target. This construction can be considered as a deformation of the environment that has to be perceived by the agents. Second, there is *attenuation* of the plot in order to deal with target disappearance. Together, these two trends take into account the targets' movements.
The *perceptions of the agents* have then to be defined. The agents perceive the plots through the environment by means of an *attraction* force (formulated to account for the nearsightedness of the agents). This force is induced by the appearance of a plot and depends on its size.
As for the interaction mechanisms, they have to be defined taking into account individual and collective points of view. Moreover, regulation mechanisms are required. A *repulsion* mechanism has been defined between agents in order to spread them in the information-less areas of the environment. This mechanism is inhibited when the agents are on a plot. There, the agents cooperate by amplifying the attenuation of the plot, in order to limit the size of the resulting group. Finally, the emergent organization is characterized by both a gathering of the agents on the plots, which leads to group construction, and an homogeneous distribution of them in the information-less areas. Each group can thus be considered as a localized target. The output of the system is stable when equilibrium is established between refreshing and solving dynamics. Figure 4 summarizes the architecture of our problem-solving model.

3. Architecture analysis principles

The approach proposed in this paper in order to analyze swarming system modifies the ASPECS process as defined by the figure 1. The Domain Requirement Description is no longer the only first activity. In fact, there are two possible ways to begin the analysis process as described in the figure 5. The first possible way consists in starting from phenomena to be imitated and conceptualize them with the Problem Ontology. From the ontology, the subsequent activities identify organizations, roles and interactions and then refine roles with concrete behaviors. The result of this path is the production of an organizational diagram composed of interacting roles with detailed behaviors that exhibit known overall properties. The second way consists in starting from the domain requirement description analyzing text usage scenarios and interviews at hand in order to produce a goal diagram for the system to be. The DRD activity may be quite complex and involves a lot of tasks before producing a goal diagram as described in [32]. However, it is not in the scope of this paper to describe these tasks.
These two paths reunite in the Goals assignment activity which intend to associate to identified goals to organization forms (coming from the ontological analysis) that are able to fulfill them.

3.1. Problem ontology description

The approach proposed in this paper consists in using the result of the Problem Ontology Description to identify organizations and roles from specific phenomena we seek to imitate. As a consequence the ASPECS process is modified to take into account this methodological point of view. POD can be done before in order to identify organizations and roles from the phenomenon that is to be imitated. In order to do so, the POD must describe the concepts of the phenomenon of interest and the actions that take place. Moreover, if the expected results of an action are known the action has to be described by a capacity and if there exist some known properties about concepts or actions then it must be described by predicates.
This approach leads to the definition of two elements usable in the verification process. On the one hand, proof obligations for the candidate organizations are generated by each capacity. Indeed, a capacity is a kind of abstraction of the behavior of an organization that needs to be verified by the role behaviors. The property to be verified can be formulated as follows :

$$requires \Rightarrow ensures$$

with, respectively, requires (resp ensures) the pre-condition (resp post-condition) of the capacity. On the other hand, each known property expressed as a predicate can be used as a lemma to help the verification process.

3.2. Organisation and Role Identification

The guidelines proposed in order to identify organizations and roles from the Problem Ontology are the following. Organizations are logical units composed of

interacting roles. Each *Action* of the Problem Domain Ontology will correspond to an interaction between two roles. Indeed, acting concepts exhibit behaviors when acting and concepts that are acted upon are either roles or parts of the environment. It is then reasonable to associate (at least) a *CommonRole* or a *BoundaryRole* to these concepts. For each *Action* two Roles and one interaction are then identified. In a first approximation, we can define an organization for each of these role couples. This process gives birth to many organizations, one for each *Action*. A role may be duplicated in several of these organizations if it takes part in several actions. The next step is now to merge the created organizations in order to define coherent organizations. The proposed guideline suggests to check if all duplicated roles are really different. If duplicated roles exhibit the same behavior or if their behaviors do not make sense when considered separately then their encapsulating organizations must be merged.
The following algorithms formalize the preceding guidelines. The initial identification of organizations is done by the initial Organization and role identification algorithm. After this step each role is defined by a statechart in the Role Plan Activity. Once the Role Plans are defined one can refine the organizations issued from the initial Organization and role identification algorithm using the Organization and role refinement algorithm. This process is repeated until a stable set of organizations is defined.

Input: POD the Problem Ontology Description
Output: A set of Organizations O composed of interacting roles
1 O:=$\varnothing$;
2 **foreach** *action* $\in$ *POD* **do**
3 | O:=O $\cup$ createOrg(action.actor,action.receiver,action);
4 **end**

Algorithm 1: Initial Organization and Role Identification

The *consistencyCheck* test asserts if the two roles in parameter can be described separately or not. If the description of one of the roles needs information coming from the other then the roles are not consistent and must be described by a single merged role.

The *consistencyCheck* test can be realized using the different role plans resulting from the Role Plan activity. Each Role Plan is described with a statechart. If the statecharts can not be fully defined because of the absence of informations present in another incomplete statechart then the roles are inconsistent and should be merged. Another case is when a predicate states a property over two or more actions linked to roles. Such a predicate, in order to be consistent, needs to be wrapped in a single organization.

The *mergeOrgs* function takes as input two organizations, O_1 and O_2, and produces as output an organization which is composed by the union of the two organizations O_1 and O_2 roles and interactions. This function algorithm is given

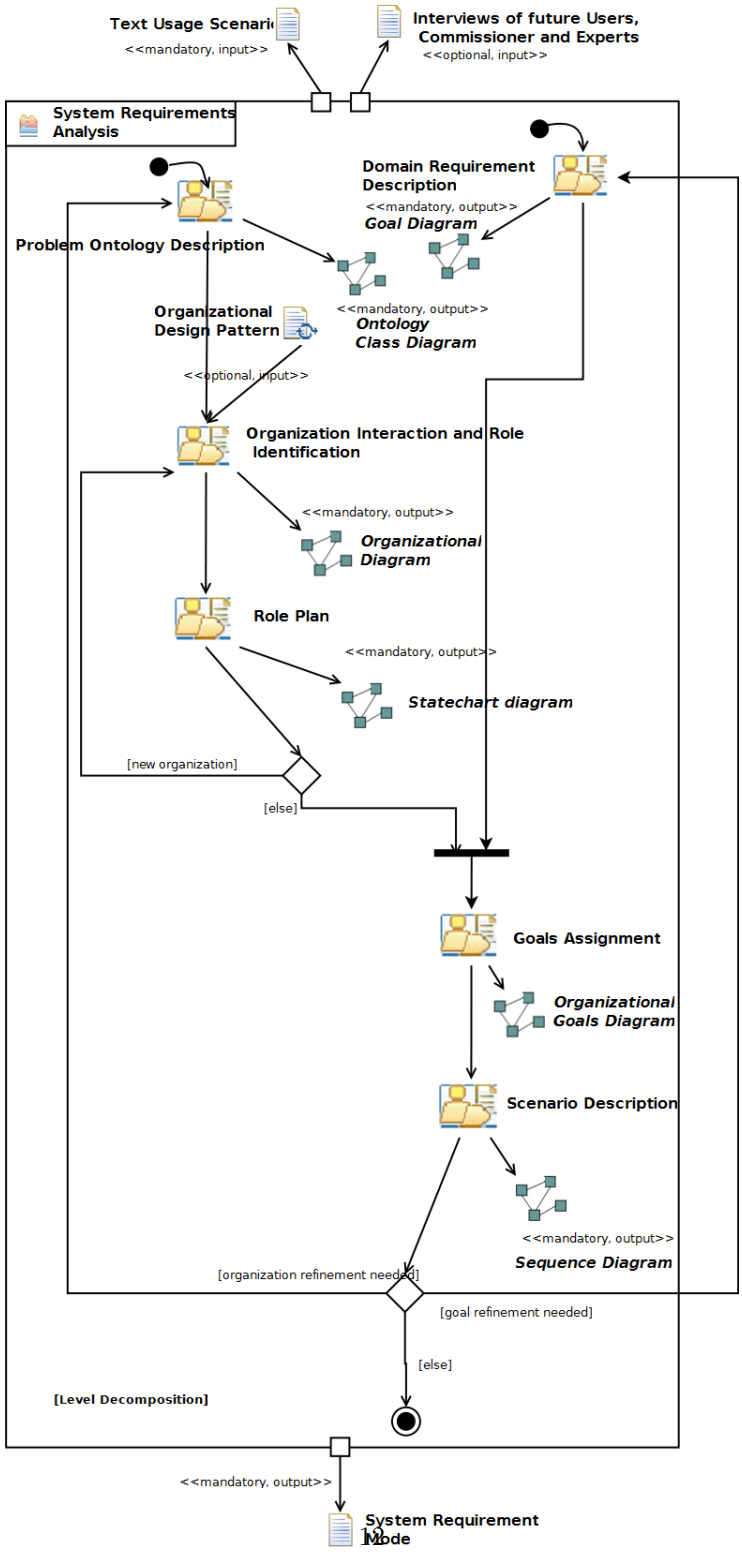

Figure 5: Modified sequence of activities for swarming system analysis

Input: A set of Organizations O composed of interacting roles
Output: O refined to eliminate inconsistent roles

```
1 while There are still unchecked duplicates do
2 |   foreach O1,O2 containing the same concept c do
3 |   |   if not consistencyCheck(O1.c,O2.c) then
4 |   |   |   O:=mergeOrgs(O1,O2);
5 |   |   |   O:=O-O1;
6 |   |   |   O:=O-O2;
7 |   |   end
8 |   end
9 end
```

Algorithm 2: Organization and role refinement

by algorithm 3.

Input: O_1 and O_2 two organizations
Output: O_3 resulting of the merge of O_1 and O_2

```
1 O3:=∅;
2 O3.R:=O1.R ∪ O2.R;
3 O3.I:=O1.I ∪ O2.I;
```

Algorithm 3: mergeOrgs algorithm

An example of merge of two organizations is given in figure 6.

3.3. Verification

One of the possible approaches for requirements elicitation is based upon goals analysis see [3, 33] for example. The general principles of these approaches is to identify the global goals of the system and then to decompose these goals in sub-level goals that contribute to the realization of the upper-level goals until reaching a satisfying level of decomposition. If such an approach would be chosen within ASPECS the result would be a hierarchical, tree-like, decomposition of goals.

The verification process proposed in this paper is composed of two steps. First, each organization is associated to a set of proved capacities. It means that the identified roles and interactions of the organization verify the property defined by the capacity. This verification relies on the description of roles and interactions. These descriptions are transformed into transition systems and used as input for specific software tools such as SAL [34]. The principle of the verification using SAL is to prove that the system composed of interacting roles satisfies the capacities defined property.

The second step consists in verifying that for each goal there is a capacity (or a set of capacities) that verify it. This kind of proof maybe very complex. Indeed, when goals cannot be related by well defined constructs, such as *logical and* or *logical or* for example, to capacities, then there is the need for a theoretical framework such as the one presented in [35]. Presenting such a framework

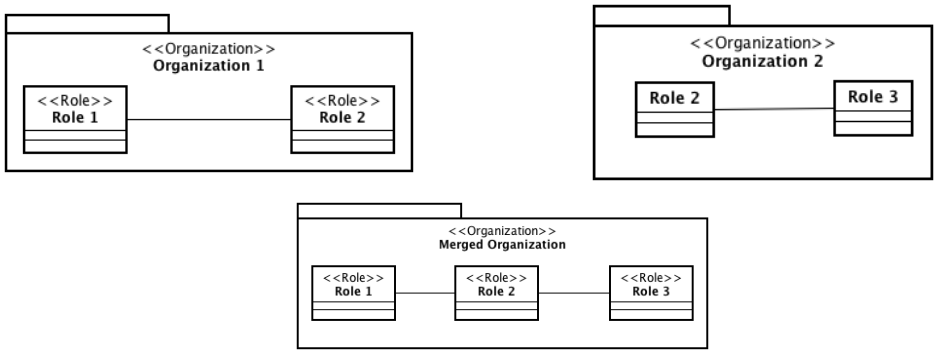

Figure 6: Example of organizations merge

would be out of the scope of this paper and the case study of the next section allows a verification without this kind of framework. However, it is possible to apply such a technique and use it for the purpose of the here described goal analysis.

4. Case study

4.1. Problem Ontology Description

The domain ontology for the analysis of the localization and tracking MAS is described in figure 7. The *SituatedElement* concept describes all concepts that are situated in the environment. In order to do this, each of these elements is associated with a location in a three dimensional coordinate system. The *Target*, *PerceptionUnit* and *Tracker* inherit from *SituatedElement*. The *Target* concept represents the targets that are to be localized and tracked. The *PerceptionUnit* is a concept that represents sensors able to perceive targets in the real world environment. This action is represented by the *Perceive* concept (stereotyped Action). According to the MAS architecture described in section 2.3 these sensors modify the environment and specifically the highness (z component) of the grid. This action is represented by the *ModifyZ* concept (stereotyped Action). The *Agent* concept represents the agent that behave as described in section 2.3. It means that they are repulsed by each others and attracted by high altitude spots. The repulsion is materialized by the *Repulsion* concept (stereotyped Action) and the attraction is materialized by the *Attraction* concept (stereotyped Action).

4.2. Organisation and Role Identification

The O_1, O_2 and O_3 organizations of figures 8, 9 and 10 are generated from respectively the *Perceive* Action, *IncrementZ* Action and *DecrementZ* Action.

The Target Moving Organization of figure 11 is the result of the merging of organizations O_1, O_2 and O_3. Indeed, the *PerceptionUnit* role behavior is not

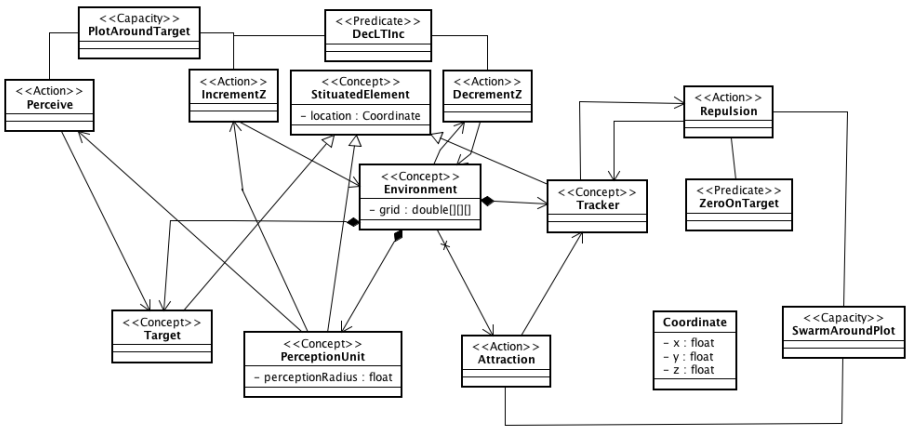

Figure 7: Problem Domain Ontology for the localization and tracking MAS

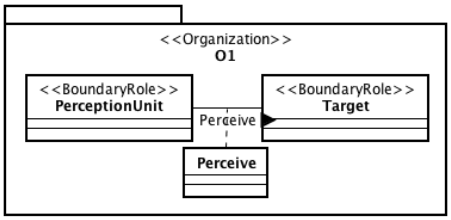

Figure 8: Organization generated from Perceive

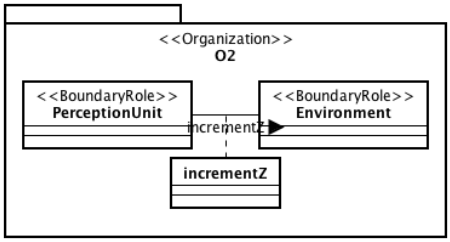

Figure 9: Organization generated from IncrementZ

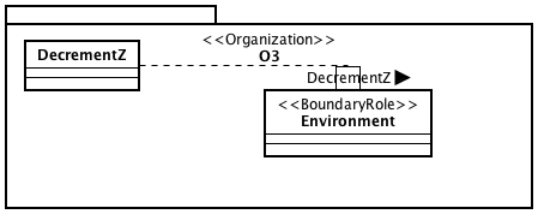

Figure 10: Organization generated from DecrementZ

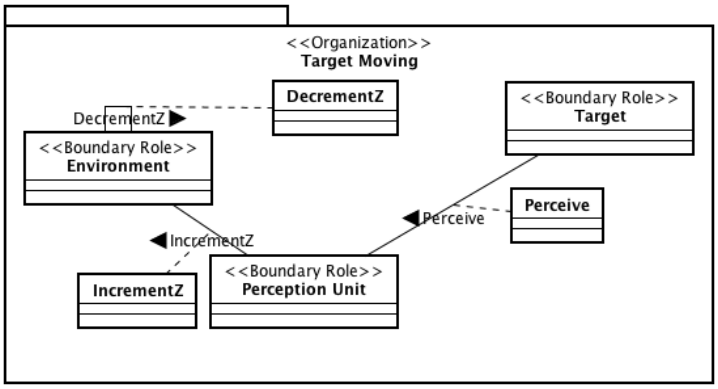

Figure 11: Target Moving Organization

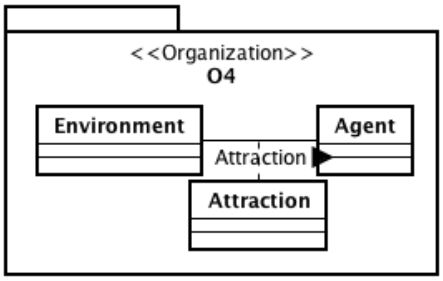

Figure 12: Organization generated from Attraction

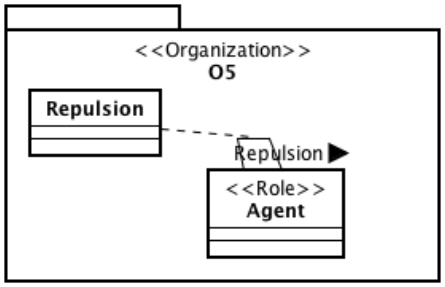

Figure 13: Organization generated from Repulsion

decomposable. It consists in perceiving targets and according to this perception modifying the z component of the environment grid. This fact merge O_1 and O_2 organizations. Since the predicate *DecLTInc* is expressed over the *IncrementZ* and *DecrementZ* actions, and could not be expressed otherwise, O_2 and O_3 must be merged.

The O_4 and O_5 organizations of figures 12 and 13 are generated from respectively the *Attraction* Action and *Repulsion* Action. The Localization and Tracking organization is the result of the merging of organizations O_3 and O_4. In this case, the behavior of the Agent role consists in movements which are computed following equations of section 2.3 that combine both attraction and repulsion. It thus makes no sense to separate this behavior in two organizations. At this point there is still one duplicated role (the Environment role) which belongs to the Target Moving and to the Localization and Tracking Organizations of figure 11 and 14. The behaviors associated to this role are different in the two organizations. In the Target Moving organization the environment is passively

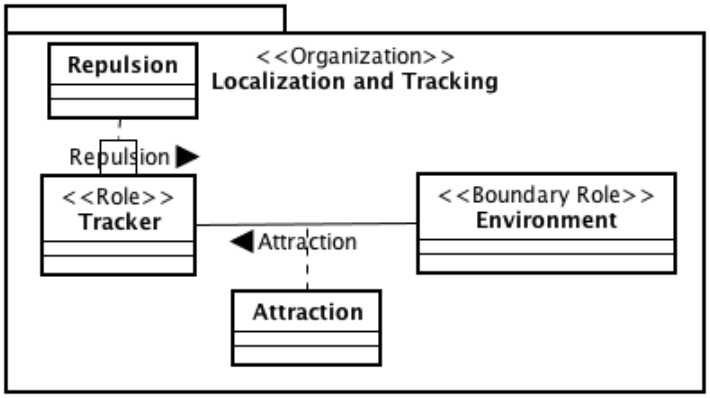

Figure 14: Localization and Tracking Organization

modified by its PerceptionUnit. In the Localization and Tracking organization the behavior of the Environment consists in attracting agents to high altitude spot. These two organizations are thus coherent.

4.3. Role plan and proofs

After Organization Interaction and Role Identification activities the analyst defines the role plan using statechart diagrams. These statecharts detail the behaviors of the roles. From these statecharts, using the operational semantics defined in [36] one can generate transition systems that can be given as input of softwares such as SAL [34] thus allowing automatic verifications. For our system, the generated proof obligations concern, respectively, the organizations of figures 11 and 14 and the capacities *PlotAroundTarget* and *SwarmAround-Plot*. The behavior of the organization defined in figure 11 must satisfy the capacity *PlotAroundTarget* taking as hypothesis the predicate *DecLTInc*. The capacity requires property is $perception(x, y)$ representing the perception of a target on a (x, y) position and the ensures is $\bigcirc(highness'(x, y) > highness(x, y))$ representing the fact that in the next step the highness of (x, y) position is greater than the highness in the current step. The property to be proven is thus $perception(x, y) \Rightarrow \bigcirc(highness'(x, y) > highness(x, y))$ with the lemma *DecLT-Inc* stating that the increment due to a target perception is greater than the decrement due to evaporation. This property was proven by using the transition system issued from the behaviors of the Target Moving organization with the SAL software using induction and the bounded model checker.
The behavior of the organization defined in figure 14 must satisfy the capacity *SwarmAroundPlot* taking as hypothesis the predicate *ZeroOnTarget*. The capacity requires property is $\exists(x, y, z) \bullet z > 0$, this means that there is at least one position in the environment with a non-zero highness. The ensures property

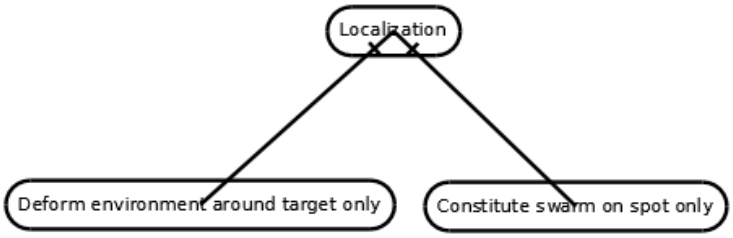

Figure 15: Goal analysis for the localization and tracking problem

is

$$\forall\, i, j : \mathit{Tracker}, (i \neq j) \wedge (i.pos.x = j.pos.x) \wedge (i.pos.y = j.pos.y) \Rightarrow i.pos.z > 0$$
$$\wedge\, \forall\, i : \mathit{Tracker}, \exists\, s : \mathit{Plot} \bullet \bigcirc(d(i,s) > d'(i,s))$$

The first line states that if two Trackers are on a same position then it must be on a Plot. The second line states that a Tracker is always attracted by a Plot. This property was proven by using the transition system issued from the behaviors of the Localization and tracking organization with the SAL software using induction and the bounded model checker.

4.4. Goals analysis

The result of the goal analysis for the problem of localization and tracking is shown in figure 15. The main goal for this problem can be reduced, as discussed in section 2.3, to a localization goal with dynamic properties. This main goal is decomposed into two sub-goals by an AND type decomposition. The first of these sub-goals is to deform the environment around targets, and only around targets, so as to define high altitude spots and maintain them as targets move. The second goal consists in building swarms around high altitude spots and only around these spots.
The proof that the two sub-goals satisfy the main goal is quite obvious. Indeed, if the environment is deformed around targets only and agents are grouped on high altitude spots then the only existing groups of agents (swarms) are necessarily constituted around identified targets. Now let us suppose there exists a target not surrounded by a swarm. It means that either the environment is not deformed around it (which is contradictory with the first sub-goal) or there are no swarm on this spot (which is contradictory with the second sub-goal). So, if the two sub-goals are satisfied, and supposing there are enough agents to cover all possible targets then all targets will be surrounded by a swarm.

4.5. Discussion

A first interesting property of the presented heuristic is that generated organizations are correct in the sense that it is impossible to produce an organization

with a role that has no interactions with other roles. This can be easily proven by induction. The initial state produced from the ontology defines one organization, composed of two or more interacting roles, for each action. In this state the roles interact as each one participates in an action. Subsequently, the transformations that are applied to organizations are of the merge type. Each merge of organization is based upon union of roles and interactions. There are no suppressions of interactions. It is so impossible to create an isolated role.
The second interesting property is that the set of generated organizations is the smallest possible respectively to the consistency test. It means that for a given problem ontology there are no smaller set of consistent organizations than the one produced by the heuristic. It can be proven by contradiction. If it were not true then among the generated organizations there would be at least two duplicates that could be merged. This is in contradiction with the loop (line 3 of Algorithm 2) that tests whether there are unchecked duplicates. The generated set of organizations is then the smallest possible.

5. Related works

For many methodologies, such as [37, 38, 39], the identification of roles and organizations is left to the designer choices or to an external requirements process. However, it is recognized that requirement elicitation is not an easy task and the assignment of such requirements to organizational structures is not a trivial task, any guidelines maybe helpful.
For goal-oriented methodologies, like TROPOS [3] or MOISE [33], there is a strong relationship between the organizational structure and the goals since each goal is associated to an organization (for MOISE) or actor (for TROPOS). The problem lies in the fact that goal-oriented approaches are not well fitted for swarming MAS as it is difficult to define goals and decompose them for this kind of system. Indeed, the principle is to replicate existing behaviors.
Other methodologies do not take into account organizational concepts, as for example Prometheus [40] and Adelfe [8].
In [5] the authors propose definitions for swarming and self-organized MAS and give general principles on how to engineer swarming MAS. These principles are illustrated through several case studies. Even if it is a very important work both on theoretical and practical aspects, the given principles are not well formalized and may sometimes be ambiguous. Their application may lead to several different results. Moreover, the approach presented in this paper is based on organizational concepts which enable the definition of ODP from swarming metaphors. These ODP can be easily reused as many MAS methodologies and development platforms integrate organizational concepts.
The authors of [13] deal with the engineering of emergence as it appears in many biological systems. The underlying assumption is that there must exist several levels and/or timescales to explain and describe emergence. Their proposition consists in a three elements architecture to be refined for any specific problem. These elements are: the System of System (SoS) model that describes the high level system, the local model that describes the lower level and the the integrated

model that describes an integration environment between these two levels. This approach has some similarities with the one presented in this paper. Indeed, the Problem Ontology Description acts as an integration model describing the system at different levels. The local model is described by organizations which are projection of agents behaviors in a specific context. Eventually, the definition of agents and their behaviors, not described in this paper as it comes later on in the ASPECS methodology, can be related to the SoS model. However, the work presented in [13] only sketch the architecture and do not propose a methodology to apply it.

6. Conclusion

This paper presents an approach for the engineering of swarming MAS. This approach is based simultaneously on the analysis of the Problem Ontology Description and the Goal-Requirements analysis of the problem. The result of the Problem Ontology analysis is the decomposition in terms of organizations composed of interacting roles. The construction of these organizations is described by an heuristic which exhibits interesting properties and the approach generates proof obligations from the ontology that should be proven by the concrete behaviors when defined. These properties once proven can be used to determine which organization is able to fulfill the identified goals from the goal-requirement analysis. Moreover, the heuristic that produces organizations only produces organizations that are syntactically correct since each role interacts with at least another role and the produced set of organizations is the smallest possible.
The described activity of Organization, Role and Interaction identification is integrated in an existing MAS methodology, named ASPECS. The ASPECS initial flow of activities is modified in order to take into account the engineering of swarming MAS. It is a minor modification and the two subsequent phases, agency domain (a sort of design) and implementation remains unchanged. The result is then an entire methodology from analysis to implementation dedicated to swarming MAS.
The interest of the presented approach is that it introduces a systematic way to engineer swarming systems that relies on the description of the system by an ontology. This description is not complex to obtain as it is the conceptualization underlying the chosen swarming system.
Future works will be concerned by the analysis of well known swarming MAS in order to constitute a library of Organizational Design Patterns library.

References

[1] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, A. Koukam, ASPECS: an agent-oriented software process for engineering complex systems, Autonomous Agents and Multi-Agent Systems 20 (2) (2010) 260–304.

[2] M. Cossentino, From requirements to code with the passi methodology, in: Agent-Oriented Methodologies, Idea Group Inc., Hershey, PA, USA., 2005.

[3] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, Autonomous Agents and Multi-Agent Systems (8) (2004) 203–236.

[4] H. V. D. Parunak, Go to the ant: Engineering principles from natural multi-agent system, Annals of Operations Research 75 (1997) 69–101.

[5] H. V. D. Parunak, S. A. Brueckner, Engineering swarming systems, in: METHODOLOGIES AND SOFTWARE ENGINEERING FOR AGENT SYSTEMS -The Agent-Oriented Software Engineering Handbook, Kluwer academic publisher, 2004, pp. 341–376.

[6] M. naturwissenschaftlichen Fakult At Ii, H. D. inf, S. A. Brückner, D. H. Meyer, D. Mathematisch-naturwissenschaftlichen, F. Ii, P. Dr, P. Dr, B. Krause, Return from the ant - synthetic ecosystems for manufacturing control (2000).
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.7907; http://www.anteaters.net/ sbrueckner/publications/2000/thesis.pdf

[7] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm intelligence: From natural to artificial systems, J. Artificial Societies and Social Simulation 4 (1).
URL http://jasss.soc.surrey.ac.uk/4/1/reviews/kluegl.html

[8] C. Bernon, V. Camps, M. Gleizes, G. Picard, Engineering adaptive multi-agent systems: The adelfe methodology, in: B. Henderson-Sellers, P. Giorgini (Eds.), Agent-Oriented Methodologies, Idea Group publishing, 2005, Ch. VII, pp. 172–202.

[9] V. D. Parunak, "go to the ant": Engineering principles from natural multi-agent systems, in: Engineering Principles from Natural Agent Systems. Annals of Operation Research, 1997.

[10] C. W. Reynolds, Flocks, herds, and schools: A distributed behavioral model, in: Computer Graphics, 1987, pp. 25–34.

[11] M. C. Marco, M. Mamei, F. Zambonelli, L. Leonardi, Co-fields: A physically inspired approach to distributed, IEEE Pervasive Computing 3 (2004) 2004.

[12] V. Hilaire, O. Simonin, A. Koukam, J. Ferber, A formal framework to design and reuse agent and multiagent models, in: J. Odell, P. Giorgini, J. Muller (Eds.), Agent Oriented Software Engineering, no. 3382 in LNCS 3382, Springer, 2005.

[13] S. Stepney, F. Polack, H. R. Turner, Engineering emergence, in: ICECCS, IEEE Computer Society, 2006, pp. 89–97.
URL http://doi.ieeecomputersociety.org/10.1109/ICECCS.2006.55

[14] V. Hilaire, P. Gruer, A. Koukam, O. Simonin, Formal driven prototyping approach for multi-agent systems, International Journal of Agent Oriented Software Engineering 2 (2) (2008) 246–266.

[15] C. Iglesias, M. Garijo, J. Gonzalez, J. Velasco, Analysis and design of multi-agent systems using mas-commonkads, Vol. 1365 of LNAI, Springer-Verlag, 1998, Ch. Intelligent agents IV: Agent theories, architectures, and languages, pp. 313–326.

[16] Software Engineering Body of Knowledge, IEEE Computer Society, 2004.

[17] I. Sommerville, Software Engineering, seventh edition Edition, International Computer Science Series, Addison Wesley, Pearson Education, 2004.

[18] FIPA, Fipa rdf content language specification, Tech. Rep. XC00011B (2001).

[19] R. G. Smith, The contract net protocol : High-level communication and control in a distributed problem solver, Morgan Kaufmann (1988) 357–366.

[20] T. R. Gruber, A translation approach to portable ontologies, Knowledge Acquisition 5 (2) (1993) 199–220.

[21] W3C OWL Working Group, OWL 2 web ontology language — document overview, World Wide Web Consortium, Working Draft WD-owl2-overview-20090611 (Jun. 2009).

[22] F. Bergenti, A. Poggi, Exploiting uml in the design of multi-agent systems, in: A. Omicini, R. Tolksdorf, F. Zambonelli (Eds.), Engineering Societies in the Agents' World, Lecture Notes in Artificial Intelligence, Springer Verlag, 2000.

[23] S. Cranefield, M. Purvis, UML as an ontology modelling language, in: Proc. Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.

[24] F. Gechter, F. Charpillet, Vision based localisation for a mobile robot, in proceedings IEEE International Conference on Tools with Artificial Intelligence (ICTAI) (2000) 229–236.

[25] F. Ealet, B. Collin, G. Sella, C. Garbay, Multi-agent architecture for scene interpretation, SPIE'00 on Enhanced and synthetic vision, Orlando, USA.

[26] D. Fox, W. Burgard, F. Dellaert, S. Thrun, Particle filters for mobile robot localization, Sequential Monte Carlo Methods in Practice Springer Verlag, New York.

[27] Environments for multi-agent systems, first international workshop, e4mas 2004, new york, ny, usa, july 19, 2004, revised selected papers, in: D. Weyns, H. V. D. Parunak, F. Michel (Eds.), E4MAS, Vol. 3374 of Lecture Notes in Computer Science, Springer, 2005.

[28] D. Weyns, H. V. D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems II, Second International Workshop, E4MAS 2005, Utrecht, The Netherlands, July 25, 2005, Selected Revised and Invited Papers, Vol. 3830 of Lecture Notes in Computer Science, Springer, 2006.

[29] F. Gechter, V. Chevrier, F. Charpillet, A reactive agent-based problem-solving model: Application to localization and tracking, TAAS 1 (2) (2006) 189–222.

[30] F. Gechter, J.-M. Contet, P. Gruer, A. Koukam, A reactive agent based vehicle platoon algorithm with integrated obstacle avoidance ability, in: SASO, 2011, pp. 129–137.

[31] F. Gechter, J.-M. Contet, P. Gruer, A. Koukam, Car-driving assistance using organization measurement of reactive multi-agent system, Procedia CS 1 (1) (2010) 317–325.

[32] V. Seidita, M. Cossentino, S. Gaglio, Adapting passi to support a goal oriented approach: a situational method engineering experiment, in: Proc. of the Fifth European workshop on Multi-Agent Systems (EUMAS'07), 2007.

[33] J. F. Hübner, J. S. Sichman, O. Boissier, Developing organised multiagent systems using the MOISE, IJAOSE 1 (3/4) (2007) 370–395.
URL http://dx.doi.org/10.1504/IJAOSE.2007.016266

[34] L. de Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, A. Tiwari, SAL 2, in: R. Alur, D. Peled (Eds.), Computer-Aided Verification, CAV 2004, Vol. 3114 of Lecture Notes in Computer Science, Springer-Verlag, Boston, MA, 2004, pp. 496–500.

[35] P. Giorgini, J. Mylopoulos, R. Sebastiani, Goal-oriented requirements analysis and reasoning in the tropos methodology, Eng. Appl. of AI 18 (2) (2005) 159–171.
URL http://dx.doi.org/10.1016/j.engappai.2004.11.017

[36] P. Gruer, V. Hilaire, A. Koukam, Heterogeneous formal specification based on object-z and state charts: semantics and verification, Journal of Systems and Software 70 (1-2) (2004) 95–105.

[37] A. Drogoul, A. Collinot, Applying an agent-oriented methodology to the design of artifical organizations: A case study in robotic soccer, Journal of Autonomous Agents and Multi-Agent Systems 1 (1) (1998) 113–129.

[38] F. Zambonelli, N. Jennings, M. Wooldridge, Developing multiagent systems: the gaia methodology, ACM Transactions on Software Engineering and Methodology 12 (3).

[39] S. DeLoach, Multiagent systems engineering: a methodology and language for designing agent systems, in: Agent Oriented Information Systems '99, 1999.

[40] L. Padgham, M. Winikoff, Prometheus: A methodology for developing intelligent agents, in: AOSE, 2002.