On the continuity of the swarm robot design using MBSE method and simulation

Khalil Aloui, Moncef Hammadi, T Soriano, A. Guizani, M. Haddar

13the International Conference on Modeling, Optimization and Simulation - MOSIM'20 – November 12-14, 2020- Agadir – Morocco "New advances and challenges for sustainable and smart industries"

On the continuity of the swarm robot design using MBSE method and simulation

K. ALOUI, M. HAMMADI, T. SORIANO

QUARTZ Lab EA7393 - SUPMECA - 3 rue Fernand Hainaut 93400 Saint-Ouen, France
alouika95@gmail.com, moncef.hammadi@supmeca.fr, thierry.soriano@univ-tln.fr

A. GUIZANI, M. HADDAR

University of Sfax
LA2MP, Ecole Natioanle d'Ingénieurs de Sfax
km 4 route de la Soukra, Sfax 3038, Tunisia
amir.guizani@live.fr, mohamed.haddar@enis.rnu.tn

ABSTRACT: *Swarm robotics is an approach to collective robotics inspired by the self-organized behaviour of social animals. This approach aims to design robust, scalable and flexible collective behaviours for the coordination of a large number of robots using simple rules and local interactions. However, this design approach faces challenges, which are not present in other multi-robot systems: The strong decentralization, the continuity of methods, finding the simple behaviours, the local communication and action, the high number of individuals and the traceability are characteristics, which make a multi-robot system "too complex to be managed effectively".*
In this paper, we present a design approach based on the Property Driven Design method for the design of swarm robots. The specification and modeling phases are performed using SysML language. We used the SysML state-machines to describe the robot behaviours. The behaviour models of the robots described with SysML are then implemented in a multi-agent tool for the simulation phase. We applied our approach to a case study of a simple robot aggregation application. Simulation results show that our approach is able to manage easily a large number of robots while ensuring the design continuity process in terms of design traceability.

KEYWORDS: *Swarm robotics, Design methods, Property Driven Design, MBSE, Simulation.*

1 INTRODUCTION

Swarm robotics have been defined as "a novel approach to the coordination of large numbers of robots". It is an approach to collective robotics, which is inspired by the self-organized behaviour of social animals. Birds, ants, fish, and bees are some examples of the simple individuals who come together in groups to complete given tasks (Sahin, 2005).

The swarm robotics research studies how to use systems made up of multiple autonomous agents (robots) to accomplish collective tasks where tasks cannot be accomplished by an individual robot alone. This approach is inspired by the system of social insects which are characterized by: robustness, flexibility and scalability. Robustness is defined as the ability to adapt to the loss of individuals. In social animals, robustness is represented by redundancy and the absence of a leader. Scalability is defined as the ability to work with different group sizes. Adding or removing individuals does not change the performance of a swarm. In social animals, scalability is promoted by local sensing and communication. Flexibility is defined as the ability to adapt to a wide range of different environments and tasks. In social animals, flexibility is promoted by redundancy, simplicity of the behaviours and mechanisms such as task allocation (Camazine et al, 2001).

Swarm robots are capable of performing tasks impossible to accomplish with other classes of robots. For example, they are able to (i) move over terrain so rough that individual robots are unable to cross, (ii) overcome obstacles larger than individual robots, or (iii) carry objects too heavy to be transported by a single robot (Fukada T et al, 1998).
All of these characteristics allow them to disperse and perform surveillance tasks, detect dangerous events, such as a chemical leak, and focus on the problem and even act to prevent the consequences (Trianni V et al, 2007).

Unfortunately, in swarm robotics, there are still no formal and precise approaches to designing behaviours at the individual level that produce the desired collective behaviour. The intuition of the human designer is still the main ingredient in the development of robotic swarm systems. Many researchers have worked on the continuity of design approaches, emphasizing the relationship between the different phases of the approach (Brambilla et al, 2004). Others chose the swarm size as a problem to solve in the design of swarm robots. They proposed that the approach should adapt to the swarm size (Dudek et al, 1993). Nevertheless, until today there is no complete methodology for the design of these types of robots.

In this paper, we present a design approach based on the Property Driven Design method for the design of swarm robots. The specification and modeling phases are performed using SysML language. We used the SysML state-machines to describe the robot behaviours. The

behaviour models of the robots described with SysML are then implemented in a multi-agent tool for the simulation phase. We applied our approach to a case study of a simple robot aggregation application.

The paper is organized as follows: in the next section, we present the related works. Our design method of swarm robots will be detailed in the section three. A case-study of a simple robot aggregation application will be considered in section four to illustrate the advantage of our approach. Finally, the paper is concluded in section five.

2 RELATED WORKS

For a long time, designers have used the 'code-and-fix' approach to develop swarm robots: It is an iterative bottom-up process, the developer tests and improves the individual behaviour of the robots until the desired collective behaviour is obtained. This approach is not structured and it depends on the developer's expertise and ingenuity. However, the design of swarming robots faces challenges that are not present in other multi-robot systems. Indeed, the characteristics of robot swarms, such as a high number of individuals, simple behaviours, strong decentralization, local communication and action, are characteristics that complicate the multi-robot system (Wooldridge and Jennings, 1998). This has prompted many researchers to propose other design approaches.

(Spears et al, 2004) used the concept of artificial force to define a new distributed framework (called artificial physics) for the control of a large number of robots. This method is very similar to the potential field method used in single robot systems, but it performs all calculations at run time. No global map is generated. The calculations are also performed locally on each robot. It is assumed that other objects in the environment apply virtual forces to the selected robot. The robot calculates the average force calculated by its observations and heads towards the direction of the average force. The force exerted on a robot from an external object depends on two things: the bearing and the distance from the external object. These two parameters can be calculated from local observations. This framework is suitable for swarm robotic studies.

(Hamann and Wörn, 2008) have proposed a method inspired by statistical physics. The authors describe the individual behaviours of robots using Langevin equations and, by analytical means. They derive a Fokker-Planck equation describing collective behavior. Another similar approach has also been proposed by (Berman et al, 2011), which derived the individual behaviours of a swarm performing task distribution using a set of partial differential advection-diffusion-reaction equations. Both methods are based on the developer's ability to model robot interactions and advanced mathematical techniques, and these methods are based on ordinary or partial differential equations, which provide reliable results if the size of the swarm is infinite. In swarm robotics, this is very often not the case, because robot swarms are generally composed of around a hundred robots and often a few dozen robots (Brambilla et al, 2013).

(Kazadi et al, 2009) have proposed a design approach based on Hamiltonian vector fields called the Hamiltonian method. From a mathematical description of collective behaviour, the method can be used to derive microscopic rules which minimize or maximize a selected numerical value (for example, the virtual potential energy of a particular state of the swarm). However, this method only deals with spatial organization behaviours such as pattern formation.

(Berman et al, 2009) have developed a top-down approach for the design of task allocation behaviour. The authors present the system as a Markov chain in which the states represent tasks and the edges represent the possibility of a robot to move from one task to another. Using a stochastic optimization method, it is possible to derive the probabilities that govern how robots change tasks in order to minimize the time required to converge on the desired allocation. This approach is specific only for the distribution of tasks and has not been extended to other collective behaviours.

Recently, (Konur et al, 2012) have proposed a different approach. The authors used model verification on a macroscopic model of a swarm of foraging robots. They specified the desired properties of the system using the Probabilistic Computation Tree Logic (PCTL), a temporal logic which includes probabilistic aspects. This approach is capable of going beyond the limits of linear temporal logic. In addition, the use of a macroscopic model, instead of a microscopic model, allows this approach to deal with systems made up of dozens of robots.

(Brambilla et al, 2012) have proposed a new top-down design method for swarm robots based on normative modeling and model verification called "Property Driven Design". The method consists of four phases: In the first phase, the developer specifies the requirements of the swarm of robots by specifying the desired properties. For the second phase, the developer creates a normative model of the swarm and uses model verification to verify that this model satisfies the desired properties. In the third phase, the developer implements a simulated version of the desired robot swarm and validates the prescriptive model developed in the previous steps. Finally, the developer implements the desired swarm of robots.

In this paper, we propose to develop a design approach of swarm robots based on the Property Driven Design method. The specification and modeling phases are performed using SysML language. The behaviour models of the robots described with SysML are then

implemented in a multi-agent tool for the simulation phase.

In 2006, OMG published the initial standard for SysML (Systems Modeling Language), an extension of UML designed to support systems engineering in general, but specifically system modeling. SysML is a graphical language for building models of large-scale, complex, and multi-disciplinary systems. It re-uses a subset of UML, and adds some new diagrams specifically designed to support systems engineering (OMG 2006).

In the following section, we detail our design approach of swarm robots. This approach is based on the different phases of the property Driven Design method using SysML diagrams.

3 DESIGN APPROACH OF SWARM ROBOTS

The principle of our design approach is that a robotic swarm system can be described through a series of properties. The specification of these requirements is carried out through SysML diagrams. These properties are the characteristics of the system that the developer wishes to realize.

In the first phase, the swarm developer uses the requirements diagram to specify the requirements representing the customer/user needs of the swarm. The SysML use-case diagram and sequence diagram are then used to describe the missions of the robots.

For the second phase, the developer creates the model describing the structure of the swarm using the Block Definition Diagram (BDD). The swarm architecture showing the composition, the links and the interactions between the robots is described with the Internal Block Diagram (IBD). The swarm behaviour is modelled using the SysML state-machines, for both the individual and swarm levels. At the end of this phase, a second requirement diagram describing the technical specification of the swarm is generated. Model verification step allows the swarm developer to verify that the model satisfies the desired properties by tracing the technical specification with customer needs and the functional/behavioural attributes generated.

In the third phase, the robot behaviour models described with SysML state-machines in the second phase are then implemented in a multi-agent tool for the simulation phase.

Finally, in the fourth phase, the developer implements the simulated version of the swarm model developed in real robots.

In figure 1, we present the steps of our approach with the different SysML diagrams used in the different phases.

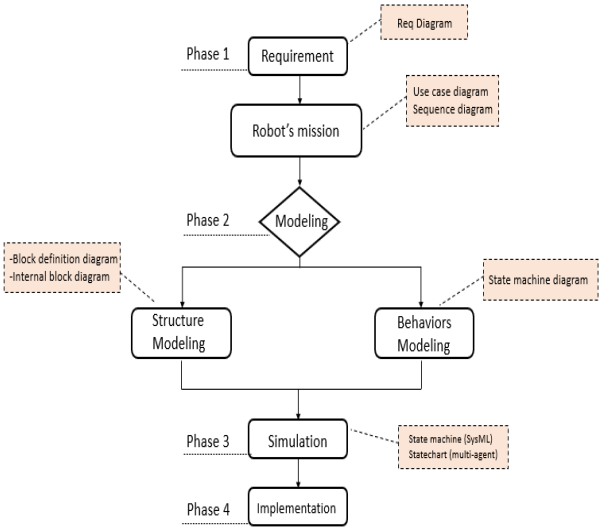

Figure 1 : Design approach process

The detail of these phases is as follows:

Phase 1: Specification of properties
The first phase consists informally specifying the system requirements and identifying the desired properties using a requirements diagram, which graphically describes a capacity or a constraint, which must be satisfied by a system. The clearer and more complete these properties, the more the system developed will comply with the requirements. The first phase of requirements analysis aims to build all the requirements of stakeholders (users, customers, etc.). It is interesting to note that the analysis phase identifies the system boundary and characterizes the interfaces with other systems. This step consists in producing the specification of requirements, mainly as requirement diagrams, allowing the traceability between the design levels, but also in in textual format. During this phase, the developer describes the main mission of the system with its various secondary tasks using the use-case diagrams and sequence diagrams to present and detail the different possible scenarios.

Phase 2: Creation of the model
In this phase, there are two steps: the first step is architectural modeling using structural diagrams (block definition diagrams and internal block diagrams) to describe the architecture of the system. The second step is behaviour modeling using state-machine diagrams.
In our approach, we choose the probabilistic method to describe the transitions between states and the actions that the system or its parts perform in response to an event. This allows the developer creating a normative model of the robot swarm that describes how robots change state over time. Each state is a simplified abstract description of the actions of a robot. The normative model should be detailed enough to capture the behaviour of the robots and their interactions, but should not be too detailed to avoid unnecessary complications. The desired properties indicated in the first phase are checked using model checking techniques to verify that

all the requirements are related to the structure and behaviour elements.
Initially, it is possible that the normative model does not satisfy all the desired properties. Thus, through an iterative process, the developer improves the model until the properties are satisfied. The result of this process is a normative model of the collective behaviour of the swarm robots that satisfies the declared properties.

Phase 3: Simulation of the model
The emergent swarm behaviour requires the simulation to check if the model developed in the second phase achieves the global need of the swarm robots. The simulation also is used to identify some optimal value of properties such as the number of robots to be used in a swarm mission. In our approach, we used the multi-agent simulation technic to capture and verify the emergent swarm behaviour.

In this phase, the swarm developer uses the model created in the second phase to guide the process of simulation model implementation.
Generally, the transition from a descriptive macroscopic model to a simulation microscopic implementation is difficult. It depends on the developer's expertise and ingenuity. In our approach, this transition has been eased because we are talking about a transition from the SysML state-machine diagrams to Statechart models to be implemented in a multi-agent simulation tool.

Even if we represented a linear process for our approach in figure 1, the practice shows that the process is iterative. Indeed, the simulation implementation can allow the swarm developer to identify some modeling faults that requires a return to the previous step. Simulation is also used to validate the swarm model by checking if the required properties are coherent with the simulation results. Therefore, the model in the phase two can be improved with values obtained by simulation, or in some cases the model is modified according to the simulation results found.

Phase 4: Real Implementation
In the last phase, the developer deploys the system on real robots. As for the transition between the previous phases, the developer can modify the simulated model or the descriptive model if certain assumptions are not verified to keep all the levels consistent.

In the next section, we will apply our approach to a case study of a simple robot aggregation application in order to illustrate that our approach is able to easily manage a large number of robots while ensuring the design continuity process in terms of design traceability.

4 CASE STUDY: AGGREGATION PROBLEM

The aggregation problem is studied either as an independent problem or as a part of more specialized tasks involving the grouping of a certain number of agents. An example of aggregation of autonomous robots is illustrated in Figure 2.

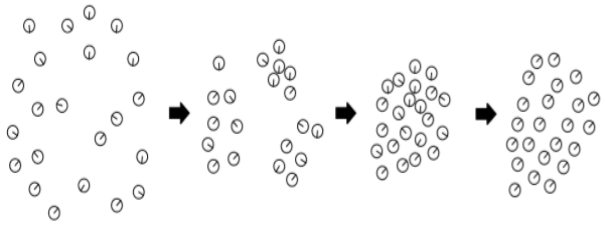

Figure 2 : An example of aggregation of autonomous agents

Among the most effective approaches for aggregating a swarm of robots are the virtual force method, the evolutionary method and the probabilistic method which we use in our approach to build the simulation model.

In this aggregation case study, we consider a white area called C with two black spots of the same size called area A and area B as illustrated in Figure 3. Each of the black spots is large enough to accommodate all the robots. We consider two sizes of swarms: 10 and 20. We use two different areas for the two different groups, of 4.91 m^2 and 19.63 m^2 respectively.

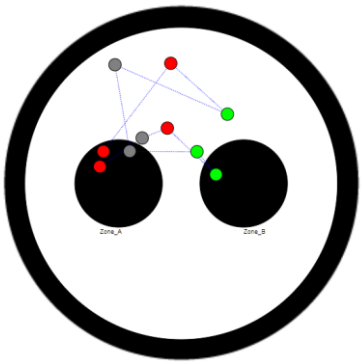

Figure 3: A screenshot of the simulated version of a swarm of robots with 10 robots

In what follows, we apply the 4-phase process of our approach

4.1 Phase one: Properties specification

The two main properties that the swarm robot must satisfy are:

- The robots must aggregate as quickly as possible in zone A or zone B. We have set a delay of 1000 seconds for the mission duration.
- The aggregate must remain stable for a specific period of time. For this, we have set a duration of 10 seconds.

Using the different SysML diagrams, we can build our model. We need to design a swarm robotic system that meets the properties specified in the first phase. The robots must regroup in zone A or B of the environment and remain stable for a period *t*. The requirements diagram illustrated in figure 4 shows two requirements

necessary to build the system. These requirements represent the customer need.

• **Choice of zone:** contains the power supply and the programming of the robots to detect the different zones and avoid obstacles. The user should check the battery level of each robot before using. In addition, the developer must program the robots to be able to avoid obstacles and choose the available area.

• **Stability in the area:** after its formation, the swarm must remain stable.

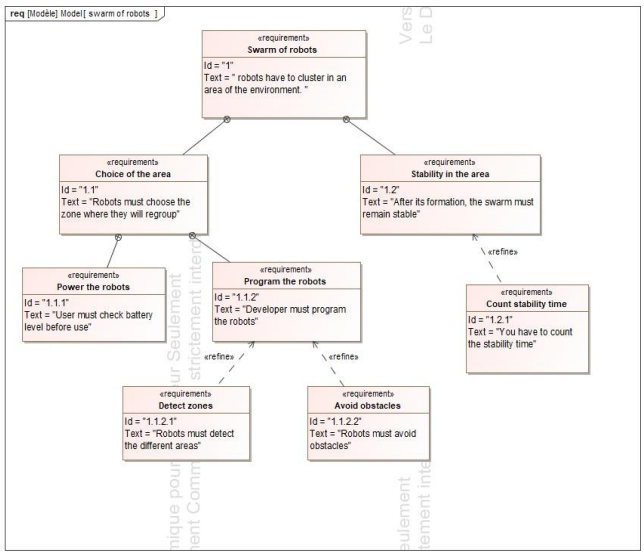

Figure 4: Requirements diagram (customer need)

Once the requirements are specified, we propose the possible use-cases that show the functional interactions of the actors and the study system. The developer describes the main mission of the system with its different secondary tasks using the sequence diagrams and use-case diagrams to present the different possible scenarios.

The use-case diagram illustrated in Figure 5 gives an overview of the functional behaviour of the swarm robot system. This mission includes 3 necessary tasks: robot programming, robot power supply and swarm self-formation.

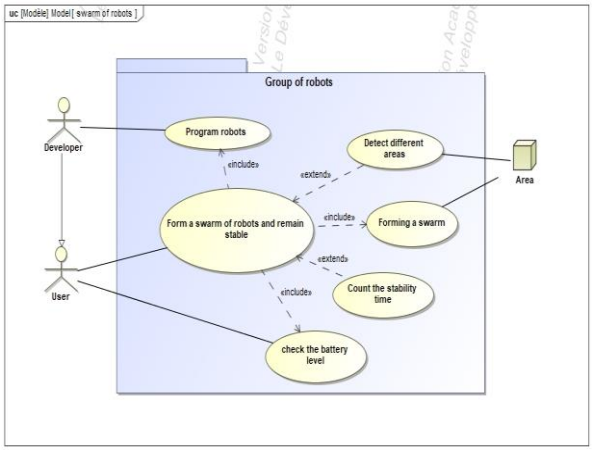

Figure 5: Use case diagram

Figure 6 shows that the developer must program the robot: This program allows the robot to distinguish between different areas of the environment and decide where to stay. Before starting operation, the user must check the robot battery level. This operation must be applied to each robot. Once the robot group is ready, two scenarios are possible:

• If the batteries are charged: the user begins to use robots to form the aggregate.

• If the batteries are low: the user must charge the batteries before starting to use them.

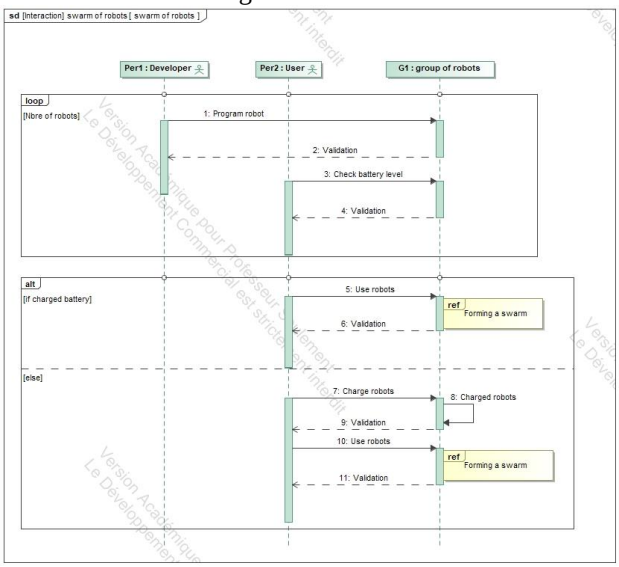

Figure 6 : Generalized sequence diagram

We use a referential block called "Forming a Swarm" that allows the swarm to perform the main function. In Figure 7 we present an instance of 5 robots to explain the "Forming a Swarm" block.

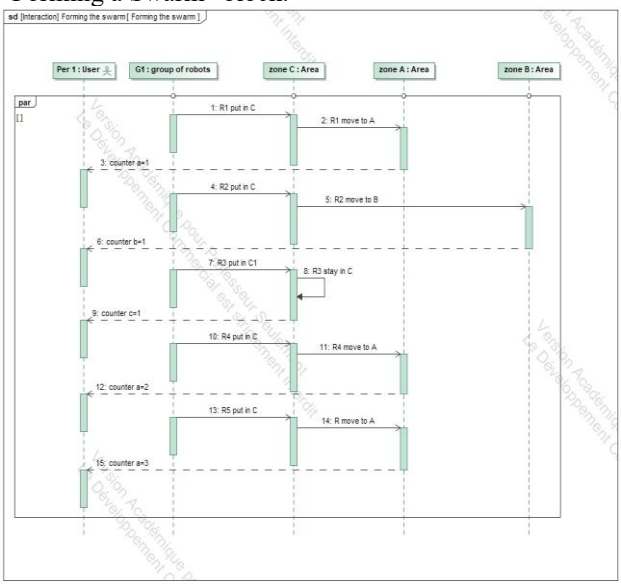

Figure 7 : Sequence diagram: an instance of 5 robots

In this example we use 5 robots: We assume that 3 robots (R1, R4 and R5) have chosen zone A, one robot (R2) has chosen zone B and the last robot (R3) has decided to stay in zone C. We define a counter that returns the number of robots detected in each zone to return this information to the user.

After the functional analysis and the mission definition, the second requirements diagram illustrated in figure 8, is used to specify the technical requirements of our system:

- **Swarm formation:** autonomy, speed and precision.
- **The swarm shape:** the general swarm shape, the geometric parameters.
- **Swarm stability:** formation time, stability time.

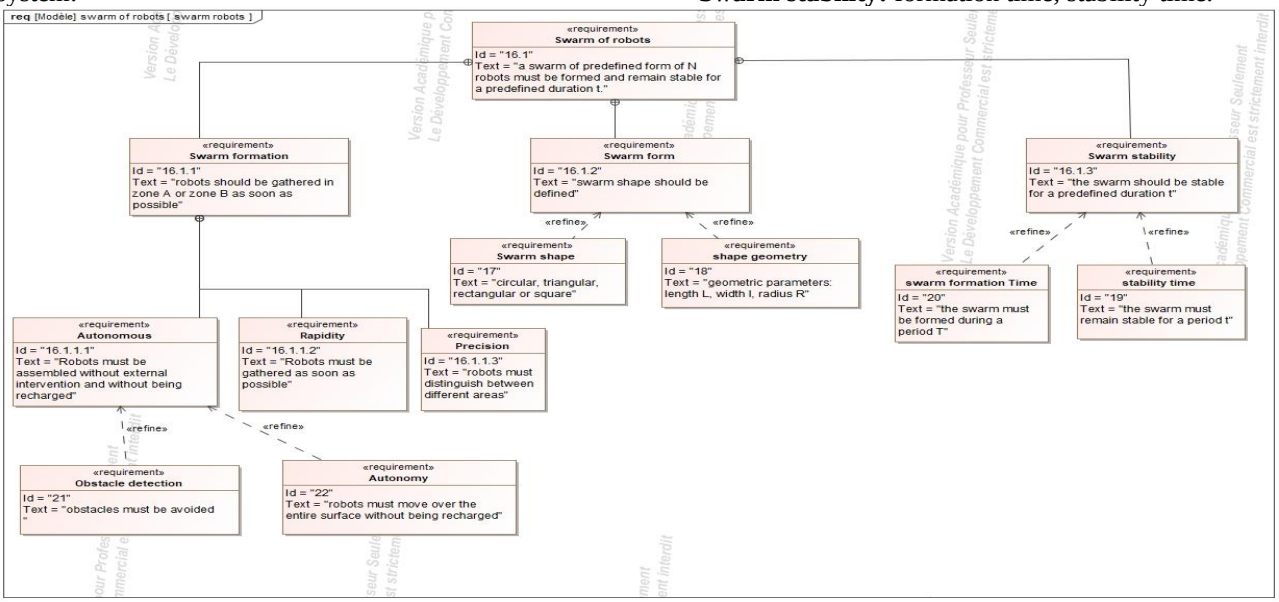

Figure 8: Requirements diagram of technical specifications

As indicated in figure 8, the formation of a swarm of robots must be autonomous without any external intervention. Indeed, the developer must design a program that allows robots to detect obstacles. In addition, the robots must move over the entire surface without being recharged each time. Each robot in the group randomly moves into the search space from its initial position. The goal of this behaviour is searching for a shelter site while avoiding static and dynamic obstacles (other robots), while the environment of the robot is completely unknown.
Obstacle avoidance is a basic behaviour found in almost all mobile robots. It is essential to allow the robot to operate in a dynamic environment. It makes it possible to avoid obstacles that appear in the robot's field of vision. For this, the robot must determine another path (via the path selection behavior). Obstacle avoidance behaviour is determined by the robot agent according to the environment in which it operates and according to its position in relation to the obstacles encountered. The method we are going to use is effective as long as we have a correct perception of the environment.

Before starting the formation, we must specify the swarm shape: circular, rectangular, each time specifying the necessary geometric parameters (length, width, radius, …)

First, we need to specify the execution time T. Once the swarm is formed, the robots must be stable for a specific period of time t.
Before going to the next step, we need to check if the technical requirements are going with the requirements describing stakeholder's needs. This verification is only the first step in a two-step process (verification, then validation) which is essential to lead to the acceptance of the system by the customer: This will be done through tests (called acceptance tests) based on the requirements identified with the customer.

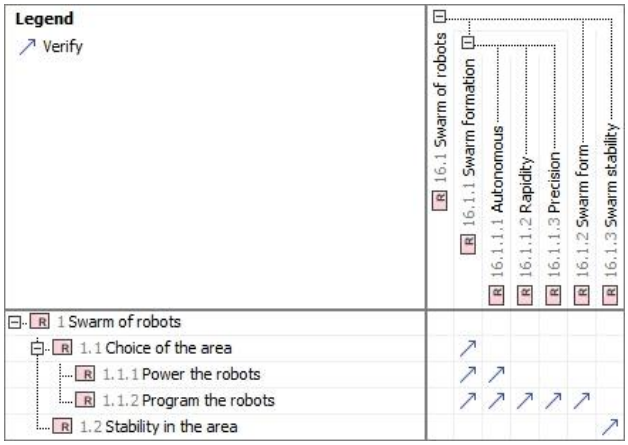

Table 1: Requirements check table

Table 1 illustrates the checking if all requirements specifying the customer needs are traced to the technical requirements of the swarm (and vice-versa).
We note in this case that all the technical requirements described in figure 8 are well linked to customer requirements described in figure 4.

4.2 Phase two: Swarm modeling

Figure 9 describes the robot development process: First, a group of autonomous robots is programmed to accomplish the mission (avoid obstacles, choose the area where will stay and communicate with other agents).

Then, the developer will check the battery levels of each robot: if the batteries are charged, we start using the robots. Else, we charge the batteries.

The design process consists of the following operations: Programming the robots, checking the batteries and recharging of the robots.

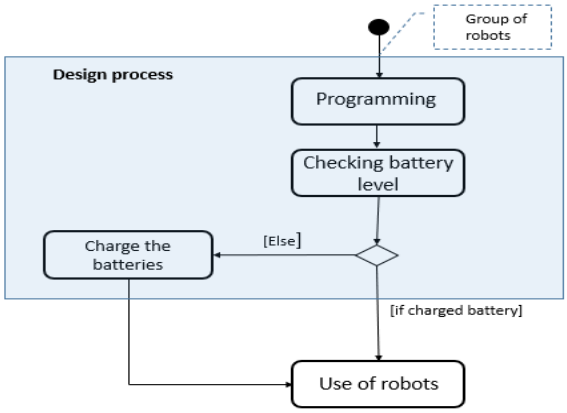

Figure 9: Robot development process

Figure 10 shows the action process that was a collection of predefined behaviours, such as displacement, aggregation, obstacle avoidance. These behaviours are necessary for moving a robot. While respecting an order of priority: in the case where the robot agent is confronted between two cases: reaching the goal (aggregate area) and avoiding an obstacle, in this case the robot must first avoid obstacles, then continue moving.

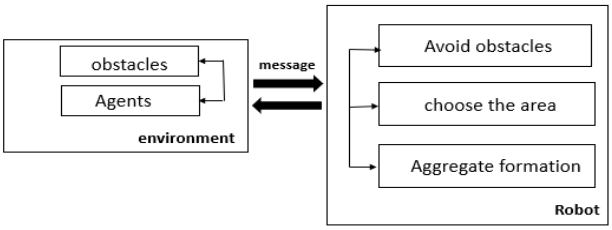

Figure 10: Action process

To understand the process, we apply in figure 11 an instance of 5 robots: 5 Robots (R1, R2, R3, R4, R5) are put in zone C, three robots have chosen zone A, one robot has chosen zone B and the latter decided to stay in zone C.

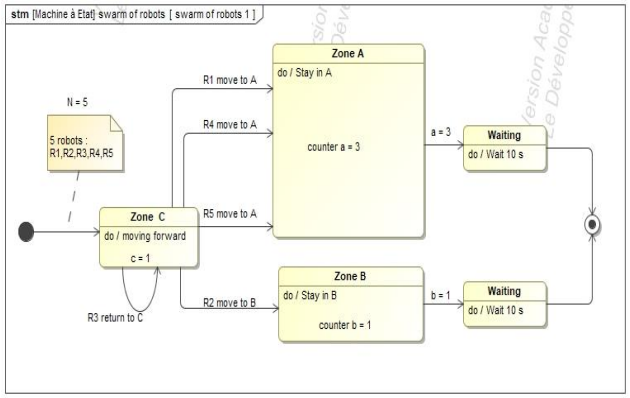

Figure 11: Design process for an instance of 5 Robots

To develop the normative model of the aggregation case study, we use the probabilistic method presented in Figure 12. We consider the three areas in which the environment is divided. We define three states: Sa, Sb and Sc. A robot in zone A or B is in state Sa or Sb, respectively. The robot outside zone A or B is in the state Sc.

A robot in zone C can move either in zone A, or in zone B or stay in zone C. This means that a robot in zone C has a probability of moving from zone C to zone A equal to $P_{ca} = \frac{\mathcal{A}A}{\mathcal{A}\ arena}$, to move from zone C to zone B equal to $P_{cb} = \frac{\mathcal{A}B}{\mathcal{A}\ arena}$, and stay in zone C equal to $P_{cc} = \frac{\mathcal{A}C}{\mathcal{A}\ arena}$ Note that $P_{ca} = P_{cb}$, because the two zones have the same size and Arena is the total zone : $\mathcal{A}_{\text{ arena}} = \mathcal{A}_A + \mathcal{A}_B + \mathcal{A}_C$

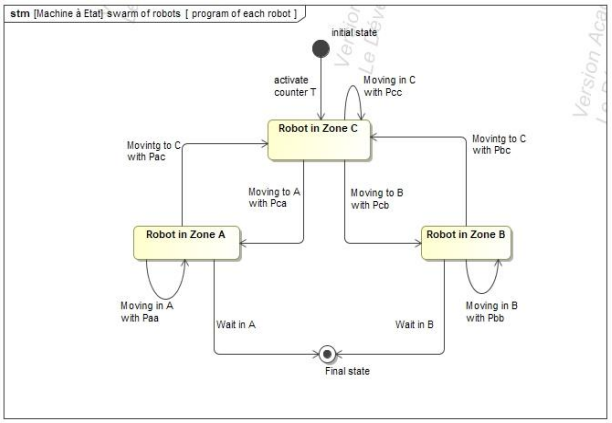

Figure 12: Swarm behaviour with probabilistic Model

The remaining probabilities depend on the behaviour of the robots. The aggregate can be obtained in zone A or zone B, so we fix the probabilities of leaving these two equal zones: Pac = Pbc.

A robot in zone A can only go to zone C or stay in zone A, therefore Paa = 1 - Pac. The same goes for zone B. According to the previous description, it follows that Paa = Pbb. The only remaining independent probability is Pac. With the check model, we can find the value of Pac which maximizes the probability of property 1. Using the simulation model (next step), we can find the best values for the parameter Pac.

Finally, with this current model, we are also able to define the specifications of the hardware capacities of the robots using the block definition diagram illustrated in Figure 13:

- A power supply system made up of batteries and conductive wires.
- A movement system consisting of DC motor, wheels and movement transmission system.
- A communication system between the robots.
- A control system made up of time counters, position sensors and an electrical control card.
- A PLC to install the program.

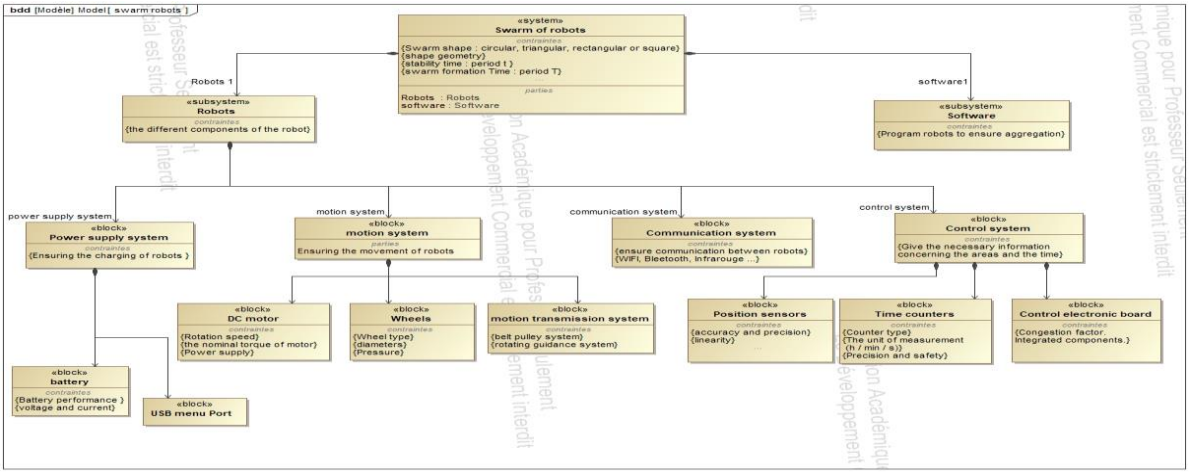

Figure 13: Block Definition Diagram (BDD)

To show the internal organization of the system, we use the Internal Block Diagram. It's a static diagram used to describe the hardware architecture of the system. It represents the instances of the shares of a block. The IBD is framed within the boundaries of the block concerned. The flows of flow (MEI) between the parts are carried out thanks to the connectors which connect their ports. The IBD is defined from the corresponding BDD. A flow enters or leaves on the one hand via a port.

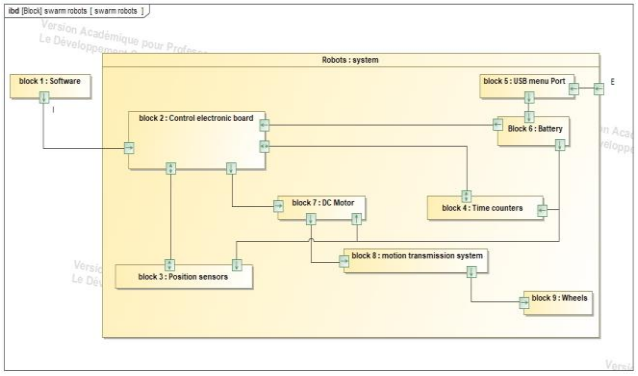

Figure 14: Internal Blocks Diagram (IBD)

Finally, we must verify that the system components described in the BDD respond to the different missions specified in the first phase (sequence diagrams and state machine diagrams). Table 2 links each component of the system with their functions.

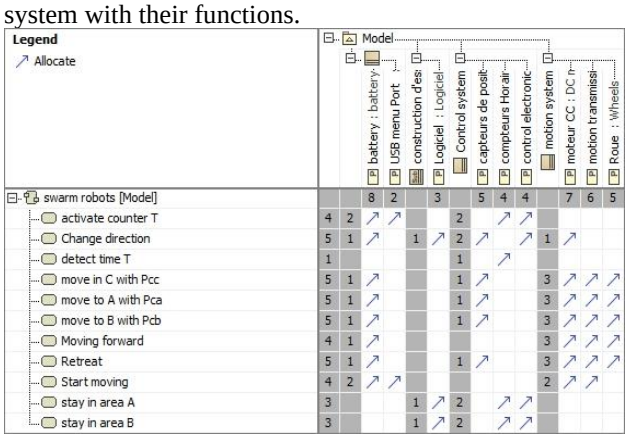

Table 2: Allocation Matrix

4.3 Phase three: Simulation

In this step we used AnyLogic™ as a multi-agent software tool to implement the swarm simulation model. We do two different sets of experiments, one for each group size. To validate our model, we measure the time required to form a complete aggregate. The robots are deployed in a random position at the start of each experiment. Each experiment is interrupted when a complete aggregate is formed or after 1000 seconds of simulation.

Experiment 1

In this example, we want to obtain a probability value for having an aggregate in area A or B (Pr1) equal to 0.8 and the probability of remaining stable for 10 seconds (Pr2) is equal to 0.6.

- P (Pr1) = 0.8
- P (Pr2) = 0.6

For a group of 10 robots, we took the value of Pac = 0.05.

In Table 3, we have calculated the different probabilities.

The parameter		Value
Probability to stay in C	P_{cc}	0.85
Probability to stay in A	P_{aa}	0.95
Probability to stay in B	P_{bb}	0.95
Probability of moving from C to B	P_{cb}	0.08
Probability of moving from C to A	P_{ca}	0.08
Probability of moving from A to C	P_{ac}	0.05
Probability of moving from B to C	P_{bc}	0.05
Waiting time	t	10 s
Execution time	T	1000 s

Table 3: the values of different probabilities for P_{ac}=0.05

We noticed the formation of two aggregates in the two zones A and B: 7 robots are placed in zone A and 3 robots are placed in zone B. In addition, the stability time is equal to 5s.

From the results obtained, we observed that a fixed P_{ac} does not favor the formation of a single aggregate. A better solution is to let a robot decide to leave based on the number of robots detected around it. We set

$P_{ac} = 1 - P_{min-ac} * (Ns + 1)$, where P_{min-ac} is the minimum probability of stay that we want for a robot and Ns is the number of other robots detected. We add 1 to the number of robots detected, because we also include the robot that chooses its next action (M. Brambilla et al, 2014)
We set the value P_{min-ac} in the interval [0.19, 0.24].

The parameter		Value
Probability to stay in C	P_{cc}	0.85
Probability to stay in A	P_{aa}	0.96
Probability to stay in B	P_{bb}	0.96
Probability of moving from C to B	P_{cb}	0.08
Probability of moving from C to A	P_{ca}	0.08
Probability of moving from A to C	P_{ac}	0.04
Probability of moving from B to C	P_{bc}	0.04
Waiting time	t	10 s
Execution time	T	1000 s

Table 4: values of different probabilities for $P_{ac} = 0.04$

We have noticed in figure 15 the formation of an aggregate in zone A: 9 robots are placed in zone A. In addition, the stability time is equal to 7s. So, the two properties (Pr1 and Pr2) are verified.

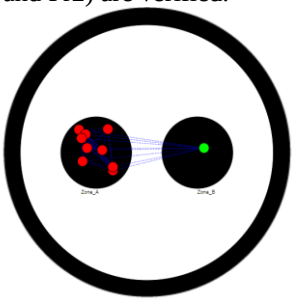

Figure 15: A screenshot of the simulation results of the swarm robot with 10 robots

This figure 16 gives us an idea on the time of start of aggregate formation depending on Pmin-ac.

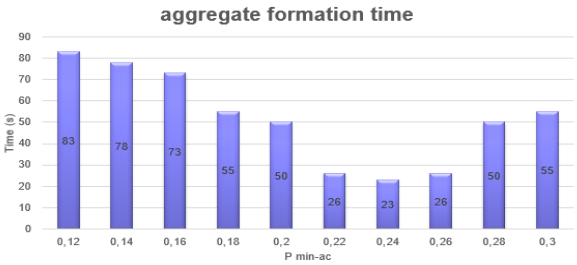

Figure 16: Time to start forming an aggregate for a group of 10 robots

We notice that the best value of P_{min-ac} is in the interval [0.18, 0.28] to obtain an aggregate as quickly as possible.

Results verification

Table 5 shows a comparison between the results obtained and the desired properties.

	Property 1 P(Pr1) = 0.8	Property 2 P(Pr2) = 0.6
Experience 1 (10 robots)	P(Pr1) = 0.9	P(Pr2) = 0.7

Table 5: Verification table of the results obtained

Experiment 2

In this example, we want to obtain a probability of having an aggregate in area A or B (Pr1) equal to 0.4.

- P (Pr1) = 0.4

For a group of 20 robots, we took the value
$P_{min-ac} = 0.12$

In Table 6, we have calculated the different probabilities.

The parameter		Value
Probability to stay in C	P_{cc}	0.88
Probability to stay in A	P_{aa}	0.6
Probability to stay in B	P_{bb}	0.6
Probability of moving from C to B	P_{cb}	0.06
Probability of moving from C to A	P_{ca}	0.06
Probability of moving from A to C	P_{ac}	0.4
Probability of moving from B to C	P_{bc}	0.4
Waiting time	t	10 s
Execution time	T	1000 s

Table 6: Values of different probabilities for $P_{ac} = 0.4$

We have noticed the formation of an aggregate in zone A: 11 robots are placed in zone A.
So, property 1 is verified: P (Pr1) = 0.55

Figure 17 presents the start time for aggregate formation. We notice that the best value of P_{min-ac} is in the interval [0.12, 0.24] to obtain an aggregate as quickly as possible.

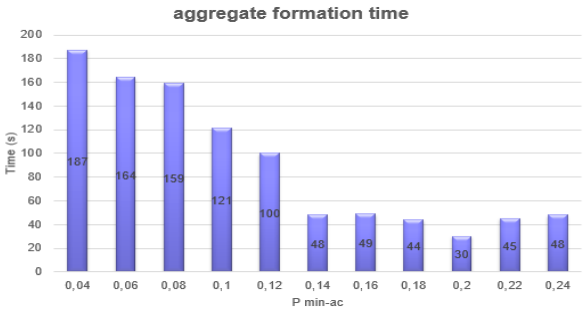

Figure 17: Time to start forming an aggregate for a group of 20 robots

4.4 Phase four: Real robots' implementation

In the last phase, the swarm developer realizes the final swarm robots. In this paper, we have not yet implemented the real robots. This task is undergoing and it could be a subject of a future paper.

5 CONCLUSION

Differently from bottom-up approaches such as code-and-fix technique, our top-down design approach, which is based on the Property Driven Design method and the use of SysML language, offers a systematic methodology towards the development of swarm robotic systems. With our approach, we are able to specify the system requirements to avoid the risk of developing the "wrong" system, that is, a system that does not satisfy the requirements. Our approach facilitates also the

development of a set of simulation and hardware independent models that can be easily reused in future applications.

The simulation results of the aggregation application show that our approach is able to manage easily a large number of robots while ensuring the design continuity process in terms of design traceability.

In the near future, we plan to validate our approach with a real implementation of a swarm robotic system. After that we can apply our design approach to more complex applications. A problem to work on is how to derive the individual behaviour of the robots from a collective behaviour. Several possibilities can be studied, such as the integration of spatial calculation and artificial evolution.

REFERENCES

Batishchev D.I and Isayev S.A, 1997. Optimization of multi functions using genetic algorithms. Interuniversity collection of scientific papers "High technologies in engineering, medicine and education". (Voronezh: VGTU) pp. 4–17.

Bayindir L and Sahin E, 2007. A review of studies in swarm robotics. Turkish Journal of Electrical Engineering, 15(2):115–147.

Bowden N et al, 1997. "Self-Assembly of mesoscale objects into ordered two-dimensional arrays," Science 276(5310), 233– 235.

Blickle T and Thiele L, 1995. A Comparison of Selection Schemes used in Genetic Algorithm, 2 Edition. TIK-Report. 67 p.

Brambilla M , Pinciroli C, Birattari M, and Dorigo.M 2012. Property-driven design for swarm robotics. In Proceedings of the AAMAS 2012. IFAAMAS, 139–146.

Brambilla M, Ferrante E, Birattari M, and Dorigo M, 2013. Swarm robotics: A review from the swarm engineering perspective. Swarm Intelligence 7, 1, 1–41.

Brambilla M, Dorigo M, and Birattari M, 2014. Property-driven design for robot swarms: Supplementary material.

Brooks R.A, 1986. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, 2(1):14–23.

Brooks R.A, 1990. Elephants don't play chess. Robotics and autonomous systems, 6(1-2): 3–15.

Cao Y, Fukunaga S, and Kahng A, 1997. "Cooperative mobile robotics: antecedents' directions and," Autonomous Robots, vol.4, no.1, pp.226–234.

Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, 2001. Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ.

Dudek G, Jenkin M, Milios E and Wilkes D, 1993. A taxonomy for swarm robots, in: Intelligent Robots and Systems' 93, IROS'93. Proceedings of the IEEE/RSJ International Conference on, Vol. 1, IEEE, 1993, pp. 441–447.

Dudek G, Jenkin M, Milios E and Wilkes D,1996. A taxonomy for multi-agent robotics, Autonomous Robots 3 (4) 375–397.

Francesca G, Brambilla M, Brutschy A, Trianni V and Birattari M, 2014. Auto Mo De: a novel approach to the automatic design of control software for robot swarms. Swarm Intell. 8(2) 89– 112.

Fukada T et al, 1998. "Self-Organising Robots Based on Cell Structures," IEEE Int. Workshop on Intelligent Robotics (IEEE Computer Society Press, Los Alamitos.

Iocchi L, Nardi D and Salerno M, 2001. "Reactivity and deliberation: a survey onmulti-robot system," in Balancing Reactivity and Social Deliberationin Multi-Agent Systems. From RoboCup to Real-World Applications, pp. 9–32, Springer, Berlin, Germany.

Jan Carlo Barca and Ahmet Sekercioglu Y, 2012. ''Swarm robotics reviewed'', Robotica (2013) volume 31, pp. 345–359. Cambridge University Press.

Kernbach S, Thenius R, Kernbach O and Schmickl T, 2009. Re-embodiment of honeybee aggregation behaviourin an artificial micro-robotics system. Adapt. Behav. 17(3) 237–259.

Minsky M, 1967. Computation: Finite and Infinite Machines. Prentice-Hall, Upper Saddle River, NJ.

Spears W, Spears D, Hamann J and Heil R, 2004. "Distributed, Physics-Based Control of Swarms of Vehicles", Autonomous Robots, Volume 17(2-3).

Trianni V et al, 2007. "From Solitary to Collective Behaviours: Decision Making and Cooperation," In: Proceedings of the 9th European Conference on Artificial Life (Springer-Verlag, Berlin, Germany).