Simulation Platform for the Evaluation of Robotic Swarm Algorithms

Michael HAMER and Cesar ORTEGA-SANCHEZ
Electrical and Computer Engineering Department, Curtin University
Perth, Western Australia
c.ortega@curtin.edu.au

Abstract—**One major problem in the development of robotic swarms is the slow process of testing. Testing different algorithms or variations of a single one using physical robots requires reprogramming every robot in the swarm before every run. Hence, the speed at which a robotic swarm can be tested is highly dependent on the time taken to reprogram the entire swarm and the physical speed at which the swarm operates. This paper details the development of a computer-based simulation platform for rapid development and testing of swarm-intelligence algorithms in an effort to mitigate the current bottleneck imposed by testing. The simulator uses an object-oriented programming environment to facilitate the implementation and modification of swarm algorithms. Simulation of food foraging in an ant-hill scenario is used to demonstrate the effectiveness of the simulator**

Keywords- Swarm intelligence, simulation tool, robotics

I. INTRODUCTION

SWARM intelligence (SI) is a field of research concerning the emergent intelligence that arises from the cooperation of unintelligent individuals. SI is an emergent behavior in many biological colonies, and through observation of these colonies, SI techniques have been successfully applied to modern problems, yielding impressive results [1].

The simplicity of individuals within a swarm makes the concept of swarm problem solving an attractive idea as it could simplify the development of highly complex systems. Furthermore, as swarms are inherently fault tolerant, swarm intelligence is highly applicable to domains with high rates of failure or that demand high availability. For these reasons, many universities and research centers are investigating swarm intelligence in the fields of robotics and artificial life. Over the past decade, many research projects have seen the implementation of small-scale robotic swarms with a focus on providing a collaborative solution to a wide array of problems [2], [3] and [4]. While these projects were successful, the process of developing, testing and debugging swarm behavior was arduous and time-consuming given the number of entities that were involved.

This paper details the development of a computer-based simulation platform for rapid development and testing of swarm-intelligence algorithms to mitigate the current bottleneck imposed by testing. The paper is structured as follows: Section II presents an overview of swarm intelligence. Section III presents the challenges in developing robotic swarms. Section IV outlines the requirements imposed on the proposed simulator. Section V provides details of the simulator's design. In section VI simulation of food foraging using an ant-hill algorithm is presented to demonstrate the effectiveness of the proposed tool. Finally, section VII gives some conclusions and discusses future work.

II. SWARM INTELLIGENCE

A. *Reductionism VS Holism*

Swarm intelligence derives its core concepts and principles from the study and observation of "social insects" such as termites, ants, bees and wasps [5]. The term "Swarm Intelligence" is used to describe the emergent behavior caused by interactions between independent entities within a swarm. Such intelligence is plentiful in nature, more specifically in insect colonies where thousands of individuals cooperate to achieve a common goal. There are two fundamental and largely exclusive schools of thought regarding the relationship between individual behavior and swarm behavior: Reductionism and Holism.

Ontological reductionism sustains that a system can be explained and implemented in terms of atomic components and their interactions [6] [7]; i.e. there is a direct mapping between macroscopic (swarm) behavior and microscopic (individual) behavior. Recently, the simplicity of reductionist thinking and its inability to explain complex systems has been criticized [6].

Contrary to reductionism, the holistic worldview proposes that a system is more than a sum of its constituent parts and intelligent action exists as a "between-relation rather than a within-relation" [8]. In Holism, behavior and intelligence are determined not only by the components within the system, but also by the interactions between components. Furthermore, swarms are treated as chaotic systems; i.e. they are "based on deterministic laws, but appear to be random", also they operate within a bounded domain, but are highly sensitive to initial conditions [9]. An implication of this sensitivity is that the progression, outcomes and behavior of the system diverge over time [10]. The work presented in this paper follows the Holistic approach to understand swarm intelligence.

B. *Behaviors*

Through the study of biological systems, researchers have gained significant insights into the behaviors that promote the emergence of intelligence in social insect colonies. While much is known about these behaviors, very little work has been done on classifying and generalizing the behavioral patterns common to social insects. Garnier et al. classify behaviors of intelligent swarms into four main categories: Coordination, Cooperation, Deliberation and Collaboration.

Coordination- Coordination is "the appropriate organization in space and time of the tasks required to solve a specific problem". This definition highlights two major requirements of an intelligent swarm system:

a. Individuals within the system should act in a manner that solves a problem.

b. Individuals should be coordinated in both time, and space

An ant colony is one of the classic examples of coordination in the animal kingdom: hundreds of thousands of individual, unintelligent ants cooperate to build an intricate nest structure, each ant tunneling, excavating, pushing and pulling at the right time, in the right place and independent of any centralized control mechanism.

Cooperation- In a cooperative swarm, "individuals must combine their efforts in order to successfully solve a problem" [5]. In general, and given the simplicity of swarm entities, the problem being solved is outside the ability of a single individual, and hence cooperation is required if a solution is to be found.

Deliberation- When a colony is confronted with multiple options, a decision must be made. Deliberation refers to the process of making this decision. Contrary to decision making by conventional intelligence, deliberation and decision making in swarms is not a conscious decision (as the swarm has no centralized consciousness), but rather, a by-product of behavioral interactions [5]. An example of deliberation is the unconscious selection of food sources by an ant colony [1].

Collaboration- Collaboration is the coordination of multiple types of behavior to find the solution to a problem. In all social insect colonies, individuals are born with societal roles based on the requirements of the colony.

III. Challenges in Developing Robotic Swarms

A. *Design Challenges*

In [11] Beni and Wang present an analysis of distributed robotic systems (robotic swarms) with the objective of highlighting the difficulties and major problems that are inherent in the design of such systems. Although the paper appears dated (published in 1991), review of the literature shows that little has been done to address the problems highlighted and thus, these problems are still relevant today.

Beni and Wang divide the development of swarm robotic systems into three distinct areas of focus according to the unique challenges they present: Physical development, Communication development and Algorithms development.

Physical development refers to the task of developing robotic agents that are physically capable of realizing the required behaviors. In the years since the publishing of Beni and Wang's critique, the advancement of computers, computer assisted design (CAD) and computed aided manufacture (CAM) has simplified the construction of robotic agents to the point that physical development now forms a much smaller portion of the overall design process. This is reflected in the general societal view that "hardware is cheap, software is expensive". Furthermore, as swarm agents are simple by definition – consisting of few sensors and actuators, the physical development of such agents is correspondingly, simple.

Effective communication is core to swarm-based problem solving. Therefore, it is crucial that communication policies and procedures provide an effective means for the agents to form a cooperative system. Since the release of Beni and Wang's critique, the development of communication policies remains a significant software based problem. Many communication protocols exist to detail the syntax and semantics of communication, and many simulators and models exist to judge the effectiveness of these protocols. However, what should be communicated and when and why communication occurs, are still problems that are solved on an application-by-application basis, with little means to evaluate communication effectiveness without trial and error.

The development of algorithms leads to the implementation of procedures and rules by which agents solve tasks. As experience and knowledge increase in the areas of swarm robotics, swarm intelligence and artificial problem solving, algorithms will likely shift from a procedural step-by-step design (the basis of modern robotics), into a rule based design that leverages agent-based "planning" functionality, adopted from the artificial intelligence repertoire of solutions. Since Beni and Wang's critique very little has been done to ease the process of algorithm design and evaluation. On the contrary, the increased physical capabilities and computational power possessed by modern robotic agents has continuously reduced the constraints on algorithms, expanding the domain of solvable problems, but also increasing the relative difficulty of algorithm design.

IV. Design Requirements

To define the scope of the proposed system, three existing robotics swarm development platforms were analyzed: Swarm Robot [2], Swarm [12] and Repast Simphony [13]. To improve functionality and overcome limitations of existing platforms the following requirements for the proposed swarm simulator were defined:

- Design and development of algorithms should be straight forward, reduce effort and involve minimal learning curve.
- Should support multiple entities, i.e. simulation of swarms, rather than isolated individuals.
- Should support entity-entity communication because communication and interaction between swarm individuals causes the emergence of swarm behavior.
- Entities in the simulator should have a construction similar to real robots. This enables algorithm portability and supports the direct translation of algorithms from the simulator, to a hardware platform.

- The simulation platform must run on a standard desktop computer.
- Given the variation in computer operating systems and computer hardware, it was decided that the simulator be designed to allow for cross platform execution and to minimize ties to particular hardware.

As the simulator was conceived as a platform upon which others could build, language selection was a crucial aspect of the design process. The development language strongly influences the learning curve and has an effect on the design, implementation and testing of behavioral algorithms. Python was chosen as the language of implementation because its syntax is highly expressive, understandable and easy to learn. It provides flexibility and support for multiple programming styles while maintaining a simple syntax, akin to pseudo-code implementation. Simplicity and expressiveness are core to the design of the Python language. This is readily observed when reading well-written Python applications. Finally, as Python is an interpreted language, it is portable to many platforms and operating systems without requiring recompilation.

V. SYSTEM DESIGN

The following sections discuss the design of the swarm simulation platform. Focus is given to the design of the two core elements in the system: Entities and Worlds.

A. *Entity Design*

Entities are the core component of a swarm. Also referred to as Agents, they are the individuals that compose a swarm, and through their communications and interactions, facilitate the emergence of swarm behavior.

One of the requirements states that "entities in the simulator should have a construction similar to real robots". This implies that each entity should contain sensors, actuators and communication systems, all tied together through a central algorithm. Fig. 1 presents the implementation of this design.

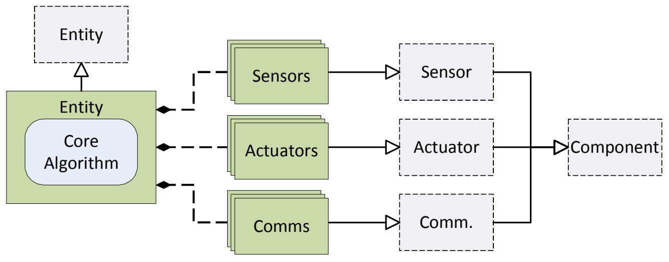

Figure 1. Entity Design.

Based on this implementation, the designer must implement the entity, the entity's sensors, actuators and communicators and finally, the core logic that ties the entity's components together. Implementation of these elements uses classes to promote reusability, abstraction of complexity and information hiding. Behaviors fundamental to all sensors, actuators, communicators and entities were abstracted into relevant abstract super classes, further promoting a simple and consistent design.

B. *World Design*

The environment or world is the second core component of the swarm simulator. While not active in controlling swarm behavior, the environment provides a medium through which swarm individuals can communicate and interact. Through environmental features sensed and acted upon by entities, the features of the environment passively direct the behaviors of entities, thereby passively contributing to the overall behavior of the swarm.

Conceptually, an environment contains many different types of information, e.g. a position within the environment may contain information about (to list a few): temperature, humidity, brightness, pheromones, and bacterial levels. While these layers of information may be related, for example, humidity is often correlated with temperature; each layer of information is isolated from the others.

The diagram in Fig. 2 represents the architecture of the object world in the swarm simulator. Classes inheriting from the Layer abstract class, represent each layer of information. Through the object-oriented principles of information hiding and data encapsulation, this class-based implementation promotes the characteristics of good software design [14].

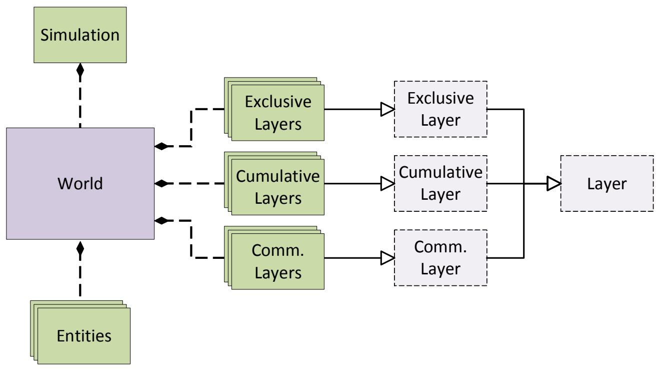

Figure 2. World Design.

Fig. 2 shows there are three different types of layer: exclusive, cumulative and communication.

An exclusive layer is a layer of information where each position within the world may contain only one value. The physical layer, which holds information about the physical composition of the world, is a good example of an exclusive layer. If a position is occupied, it is considered full, that is, it cannot be occupied by any other object. In exclusive layers only one object may contribute to the value of a position.

In a cumulative layer the combination of all effects and influences on a position, yields the final value for that position. A pheromone layer is an excellent example of a cumulative layer, whereby the summation of all pheromones contributing to a point determines the final pheromone count for that point.

In a communication layer every position in the layer is affected by the same information. A communication layer may contain, for example, radio, microwave or sound based communication. Due to the relative speed of these mediums when compared with the speed of algorithm execution, it was assumed that every position within the world would be affected by the communication instantaneously. However, properties of the communication channel such as power and attenuation can be defined so that positions far from the point of origin may not receive the communication.

C. Integration

An example of integration of entities in an environment to create a swarm simulation is presented in Fig. 3. This figure describes the relationships and interactions of objects within an ant-hill simulation. Special note should be made of the object-oriented nature of the simulator. By using abstract class hierarchies the underlying complexity of the simulation platform is abstracted from the user-level simulation design, thereby improving the stratification of the system, and promoting an extensible and maintainable system. The design and implementation of the ant-hill will be explained in more detailed in Section VI.

In every simulation, a deterministic environment is assumed, i.e. a system will always produce the same output for a given set of inputs. However; a swarm system is a chaotic system and small variations in the initial state can yield vastly different and unexpected results in simulation.

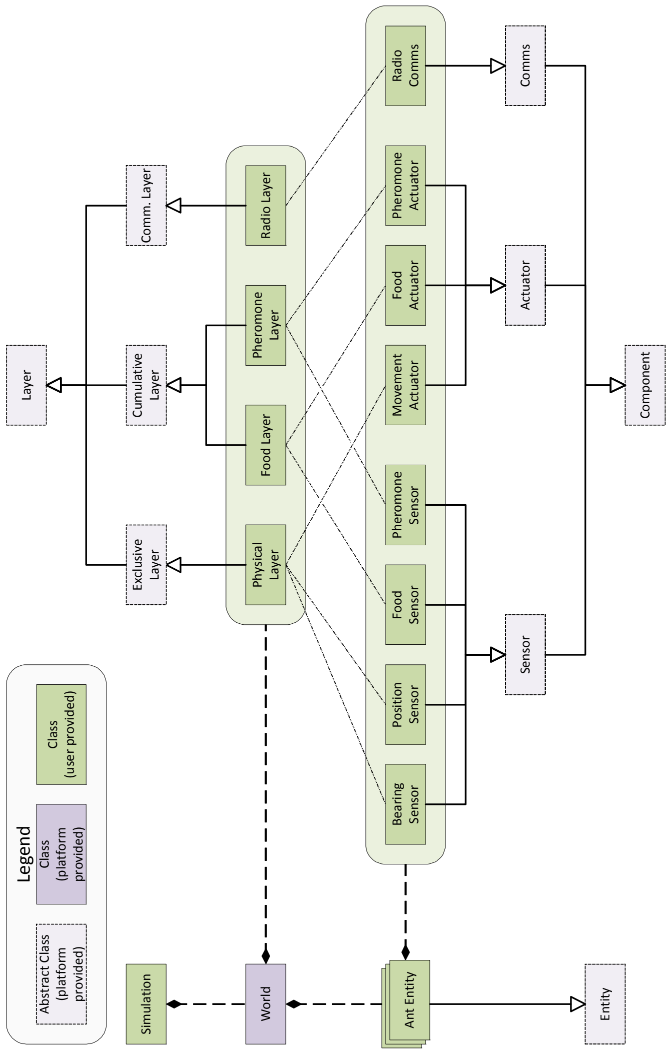

Figure 3. Object-oriented ant-hill simulation

D. Logging Subsystem

The key reason for simulation is to evaluate performance. For that purpose, the swarm simulator includes a logging subsystem that allows rapid evaluation and comparison of algorithms. The logging subsystem is built on top of the core simulation platform. While not required for simulation the logging tool provides the mechanism by which entities and layers can record custom messages to indicate behavior patterns and events.

On every step of the simulation process, the logging subsystem performs the following sequence:

1. Print custom output messages when synchronizing the layer's buffers.
2. Save the state of the world (layer states + entity states) prior to entity execution.
3. Print custom output messages during execution of each entity.

The output from this sequence is stored in XML format. XML was chosen, as it is both human readable as well as structured. Furthermore, to aid in automatic parsing, a serialized version of the XML tree used to construct the file is appended at the end of the file as an XML comment.

VI. Ant-Hill Simulation

Core to any design is an understanding of the problem being solved. For this purpose, the simulation of an ant-hill was broken down into four key behaviors:

- Movement around the environment searching for food.
- Collection of food and returning it to the ant-hill.
- Laying pheromone trails to mark the route to food.
- Following pheromone trails to find a possible food source.

From these four, simple behaviors, both the world and the swarm entities were developed.

A. Ant-Hill World Design

The first stage of entity design is to highlight the types of information present in the simulation, as they dictate the layers of the world, as well as the sensors and actuators required by the entities. In the set of ant-hill behaviors mentioned above, two core information types can be observed: Food and Pheromones. Additionally, the entities must have a physical presence within the world to enable movement. Based on these observations, it was decided that the world required three layers: food layer, pheromone layer and physical layer.

The physical layer is core to most simulations and is therefore provided as part of the simulation platform. However, both the food layer and the pheromone layer required custom implementations.

In the food layer multiple entities accumulate food at the home location. Therefore, the food layer should behave as a cumulative layer.

In the pheromone layer many entities can lay pheromone trails and multiple trails can contribute to the same location, hence this layer was also implemented as a cumulative layer. However, unlike the food layer, the pheromone layer must implement additional functionality to allow a gradual decay of pheromone trails.

B. *Ant-Hill Entity Design*

According to the original set of requirements, interaction of entities with the world should mimic the mechanisms used by a real robot. For this reason, the following virtual sensors and actuators were designed to simulate entity-to-entity and entity-to-world interactions:

Food sensor and food actuator- The food sensor indicates to the entity whether food exists in the food layer at the current location. The food actuator allows an entity to manipulate and control the current location within the food layer by picking up and putting down food.

Pheromone sensor and pheromone actuator- The pheromone sensor was designed to allow the entity to look for food in a 3x3 area within the food layer. This mimics the concept of smells wafting and allows the entity to determine and follow a more optimal path to food, as it has a wider perception of the environment. The pheromone actuator is designed as a passive device to mimic the way ants passively exude pheromones while moving. The entity maintains control over whether pheromones are exuded or not; however once activated the process itself is passive.

Position sensor, bearing sensor and movement actuator-As entity movement is core to most simulations, the Position sensor, Bearing sensor and Movement actuator are provided as part of the simulation platform and do not need custom implementations. These three elements interact with the physical layer.

C. *Core Logic Design*

Having defined the construction of the world and the components through which the entity interacts with the world, the final stage of entity design was to develop the core logic that ties inputs to outputs and defines the overall entity behavior. Abstracting complexity through the usage of sensor and actuator classes, allowed for a very simple behavioral definition. It can be summarized as a series of if-then-else statements:

- If standing on food and not at home, then collect food and return home;
- else, if standing on a pheromone trail, then follow it;
- else, move randomly.

D. *Simulation Development*

Having developed all components of the simulation (layers, sensor, actuators, behavior), the final stage of development is the integration of these components into a simulation. This process involves defining the world, defining the initial state, creating the entities and finally, running the simulation. Fig. 4 shows the implementation of the ant-hill simulation in Python.

```
#import world
from SwarmIntelligence.World import World

#import entities
from Ant import Ant

#import layers
from SwarmIntelligence.Layers.Physical import
PhysicalLayer
from FoodLayer import FoodLayer
from PheromoneLayer import PheromoneLayer

#set size
Size=(40,40)
World.set_size(*Size)

#initialise layers
World.layer_register(PhysicalLayer(*Size))
World.layer_register(FoodLayer(*Size))
World.layer_register(PheromoneLayer(*Size))

#load initial layer data
World.FoodLayer.load_static("fooddata")

#initialise 20 entities
for i in range(0,20):
    World.entity_register(Ant())

#run the world for 500 cycles
World.run(500)

print "Simulation Complete. Saving Results"

#save the replay only log
World.Log.save("replay.xml",World.Log.REPLAY)

#save the debug (and replay) information
World.Log.save("replay.xml",World.Log.DEBUG)
```

Figure 4. Simulation implementation

The usage of Python 2.6 as the implementation language provided a simple, expressive syntax that is above all, easy to learn. Furthermore, the simulation platform provides layers of abstraction that allow the designer to focus on the problem at hand, rather than the complexities of simulation.

The diagram in Fig. 5 summarizes the object-oriented implementation of the ant-hill simulator.

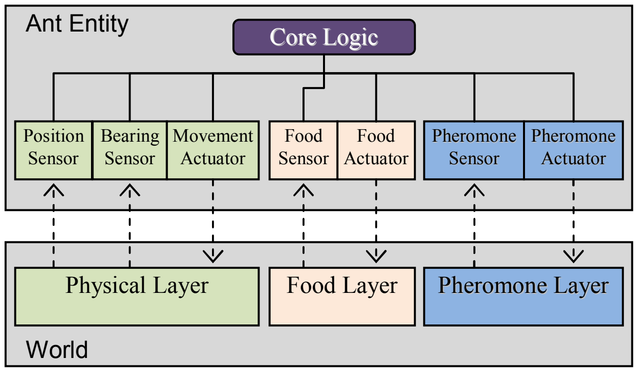

Figure 5. Ant-Hill Structural Design

E. Results

The implementation discussed in previous sections successfully simulated the ant-hill behaviour: Food foraging was achieved through stigmergic cooperation facilitated by the marking of pheromone trails.

For visualisation, a graphics library was used to display layers and entities on the screen. Ants were presented as little circles, food sources were represented by light-coloured squares and pheromone trails as sequences of light-coloured squars resembling paths. Fig. 6 shows sequential screenshots of the swarm behaviour simulation: Ants locate food and bring it to the ant-hill, located in the center of the screen.

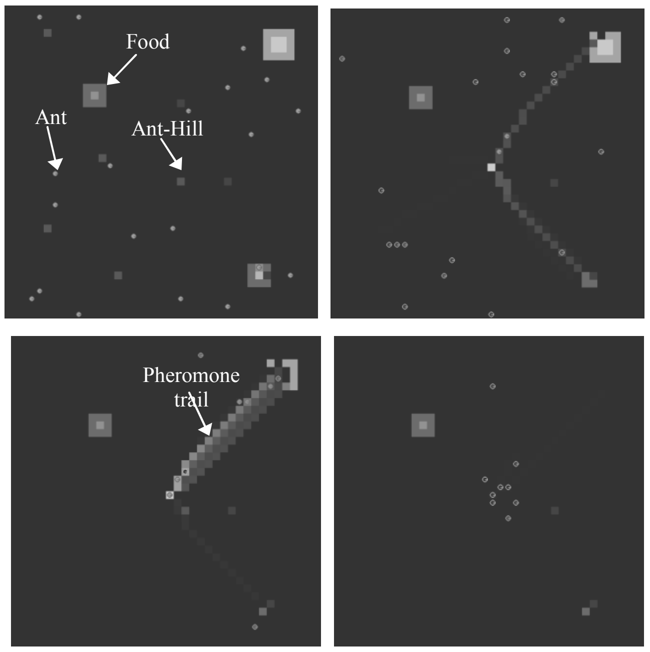

Figure 6. Screenshots of ant-hill simulation

Successful simulation of the ant-hill demonstrates the capability of the simulator to facilitate the development and evaluation of swarm algorithms. Furthermore, it should be noted that this simulation is purely an example usage of the simulation platform. An unbounded number of unique swarm applications could be developed and simulated with the proposed tool.

VII. Conclusions and Future Work

This paper presented the design and implementation of a simulation platform specifically designed to aid in the development of multi-robot swarm systems. The objective of the platform is to remove the bottleneck of swarm algorithm development caused by the need to reprogram the complete swarm every time a new version of the algorithm is to be tested.

The simulator uses an object-oriented approach to implement swarm algorithms. Worlds and Entities are created as independent objects that interact with each other, resembling the way real robots interact with their environment. Python 2.6 language was used to describe the objects required by each swarm algorithm. The current version of the simulator is fully functional as demonstrated by the simulation of an ant-hill.

A logging system allows the comparison and quantification of algorithms' performance. However, while functionally sound, direct comparison between algorithms requires manual parsing of the XML log to extract required data. Future work includes the development of a tool to aid in the automatic comparison of algorithms' performance. Optimally, this tool should automate the testing process and be capable of comparing results over a large number of algorithm versions through the automatic generation of graphs and testing reports.

References

[1] Dorigo, M and Gambardella, L M. "Ant colonies for the traveling salesman problem", BioSystems,1997.

[2] Institut für Parallele und Verteilte Systeme, University of Stuttgart and Institute for Process Control and Robotics. Open-source micro-robotic project. [Online] 2008. www.swarmrobot.org.

[3] Dorigo, M and Future and Emerging Technologies program, European Commission. "Swarm-bots: Swarms of self-assembling artifacts" [Online] 2005. www.swarm-bots.org.

[4] Institute for Process Control and Robotics, Universität Karlsuhe. "I-SWARM". [Online] 2008. http://i-swarm.org/.

[5] Garnier, S, Gautrais, J and Theraulaz, G. "The biological principles of swarm intelligence", Swarm Intelligence, Vol. 1, pp. 3-31, 2007.

[6] Uttal, W. Toward A New Behaviourism: The Case Against Perceptual Reductionism. Lawrence Erlbaum Associates, Inc., 1998.

[7] Sheehy, P. The Reality of Social Groups. Ashgate Publishing, 2006.

[8] Parkin, D and Ulijaszek, S. Holistic Anthropology: Emergence and Convergence. Berghahn Books, 2007.

[9] Haefner, J W. Modeling Biological Systems. 2nd ed. Springer Science, 2005.

[10] Kampis, G. Self-Modifying Systems in Biology and Cognitive Science. Pergamon Press, 1991.

[11] Beni, G and Wang, J. "Theoretical Problems for the Realization of Distributed Robotic Systems". Proceedings of IEEE International Conference on Robotics and Automation, 1991.

[12] Swarm Development Group. SwarmWiki. [Online] 2009. www.swarm.org.

[13] Argonne National Laboratory. Repast Simphony. [Online] 2009. http://repast.sourceforge.net.

[14] McConnell, S. Code Complete 2. Microsoft Press, 2004.