A Comprehensive Survey on Decentralized Modular Swarm Robotic systems and Deployment Environments

Tamer Abukhalil *tabukhal@bridgeport.edu*
Department of Computer Science and Engineering University
University of Bridgeport, 06604, Bridgeport, USA

Madhav Patil *mpatil@bridgeport.edu*
Department of Computer Science and Engineering University
University of Bridgeport, 06604, Bridgeport, USA

Tarek Sobh *sobh@bridgeport.edu*
Department of Computer Science and Engineering University
University of Bridgeport, 06604, Bridgeport, USA

Abstract: This paper presents the results of a comprehensive investigation of the current state of swarm robotics research, organizing and classifying that research into a preliminary taxonomy. We aim to provide an analysis of existing swarm systems in an attempt to define the starting point of potential algorithms leading to the development of a new swarm system platform and software design. In other words, we provide a detailed summary of systems that have been classified under four main categories of the general multi-robot system platforms, namely: self-reconfigurable, modular, self-replicating, and swarm systems. We present a preliminary taxonomy for swarm robotics and classify existing studies into this taxonomy. In later sections of this survey, we do not only address the fact that there is a shortage of available software packages and interfaces that are integrated with capabilities to distribute decentralized algorithms over the swarm system, but also we introduce the challenges of having such software/application for controlling multiple expandable and reconfigurable swarm agents.

Keywords: Decentralized swarm intelligence, Modular robotics, Swarm system behavior, Swarm robot interactive software, Decentralized robots control.

1. INTRODUCTION

Decentralized modular robotics is an emerging area that has attracted many researchers over the past few years. It has been proven that a single robot with multiple capabilities cannot necessarily accomplish an intended job whereas different robots, each one with its own configuration, are more flexible, robust and cost-effective. Moreover, the desired tasks may be too complex for one single robot, whereas they can be effectively done by multiple robots [1, 2]. Modular robotic systems have proven to be robust and flexible [3-7]; such properties are likely to become increasingly important in real-world robotics applications. However, there is a lack of software packages which provide control for various platforms of robots individually and allow concurrent control of heterogeneous robotic teams. Thus we will be interested in designing such control applications. Fig. 1 shows the break-down of the system hierarchy:

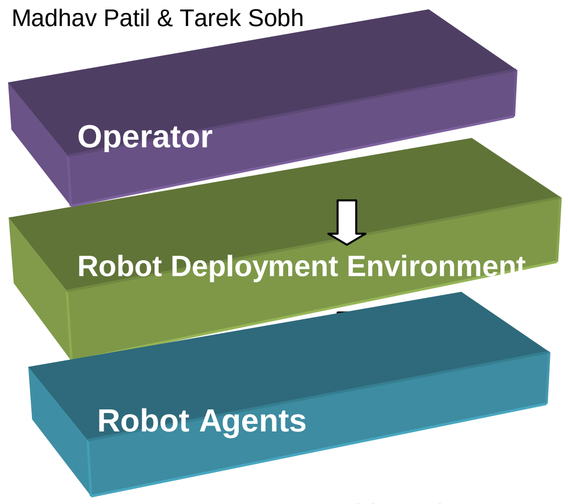


FIGURE 1. System hierarchy

Different research efforts have been carried out in the past decade that attempt to resolve coordination and decision making problems in swarm robotic systems. Such studies include simple models such as foraging [8, 9]. The multi-agent robotics system consisting of a number of identical robots proposed in [10] for a decentralized robot is yet another approach to swarms. In [11] Roderich and others proposed the concept of self-assembling capabilities of the self-reconfigurable S-bots as it is actually indicated in this particular research paper. The s-bots are in fact the “swarm-bot” robots that were developed by the Francesco Mondada et al. [12]. Swarm-bots can either act independently or self-assemble to form a swarm by using their grippers. In [13] Fukuda and Nakagawa proposed the concept of the DRRS (dynamically reconfigurable robotic system) based on a cell structure for removable parts. The implementation was then called CEBOT, the first cellular robotic system. CEBOT is a heterogeneous system comprised of agents with different locomotion functions. One of the critical aspects of this type of system is the communication between the members of the swarm [14], which is usually carried out using radio-links. In [15] Dumbar and Esposito studied the problem of maintaining communications among the robots performing tasks.

In conducting our survey we identified a criteria based on assumptions similar to the work presented in [16]. In other words, we are interested in systems that involve algorithms designed specifically to operate heterogeneous/homogenous robots performing various tasks. These assumptions can be summarized as follows:

- The systems examined are composed of an undetermined number of embodied robots;
- Robots are identical or heterogeneous with different capabilities;
- Robots have decentralized control;
- More robots may be added to the system at any time;
- Robots are multipurpose, not task specific;
- A coordination model should exist to operate the different robots.

We present a comprehensive study on the behavior of swarm systems dedicated to different tasks/applications with a new collective and mobile reconfigurable robotic system. The modules are fully autonomous mobile robots that, by establishing physical connections with each other, can organize into modular robots. We do not consider any particular hardware or infrastructure of each swarm agent, as our focus in our work is building control mechanisms that allow the system to operate several simple heterogeneous agents.

This survey is organized as follows: in Section II we provide a comprehensive survey of two primary swarm approaches, namely biologically inspired and functionally built robots. A comparison between existing re-configurable robots is presented in Section III. Section IV will provide some discussion about self-replicating robots. Section V provides an analysis of the software operating application systems that have been introduced.

2. RELATED WORK

Swarm behavior was first simulated on computers in 1986 with the simulation program Boids [17]. This program simulated simple agents (Boids) that are allowed to move according to a set of basic rules set by programmers. Those rules are simply algorithms usually called the PSA or Particle Swarm Algorithm. The model was originally designed to mimic the flocking behavior of birds, but it can be applied also to schooling fish and other swarming entities.

Different studies of complexity have been carried out over these types of systems [7, 14, 15, 18-24]. There have been many interpretations of the understanding and modeling of swarming behavior. Some researchers have classified these behaviors into two primary types namely biologically inspired and functionally built robots [25], while others have proposed two fundamentally different approaches that have been considered for analysis of swarm dynamics. These are spatial and non-spatial approaches [26]. In the first approach, “biologically inspired”, designers try to create robots that internally simulate, or mimic, the social intelligence found in living creatures. The second approach, “functionally designed”, use functionally designed robots generally with constrained operational and performance objectives. Consequently, these “engineered” robots may only need to generate certain effects and experiences with the user, rather than having to withstand deep scrutiny for “life-like” capabilities [5].

2.1. Biology inspired robots:

Multiple researchers have shown some interest in the foraging and other insect inspired coordination problem and have investigated these behaviors and summarized them into algorithms. Others were interested in exploiting swarm robots in the tasks of localization [18], surveillance, reconnaissance [19], and hazard detection [20, 21]. Pheromone-trail-based algorithms sometimes have the ability to dynamically improve their path [22] and can adapt to a changing terrain [27]. Ant-inspired foraging has been implemented in robots by various groups. One major difficulty can be exhibited in implementing the pheromone itself. Others have resolved problems of how robots should interact in the swarm. There have been many approaches dedicated to this:

• *By means of physical markers*, where robots physically mark their paths in multiple ways, such as depositing of a chemical alcohol on the ground [22], drawing lines onto the floor using pen and paper [21], laying trails of heat [23], storing the pheromone values radio frequency identification tags RFID [24], or emitting ultraviolet light onto a phosphorescent paint [28].
• *Transmitting wireless signals when laying virtual landmarks in a localization space*. In the work of Vaughan et al., robots maintain an internal pheromone model with trails of waypoints as they move, and share it with other robots over a wireless network [27].
• *Virtual pheromones* that consist of symbolic messages tied to the robots themselves rather than to fixed locations in the environment. In their experiment [19], the virtual pheromone is encoded as a single modulated message consisting of a type field, a hop-count field, and a data field. Messages are exchanged between robots through infrared transmitters and receivers. It is assumed that the robots receiving the pheromone can measure the intensity of the IR reception to estimate their distance from the transmitter.
• *Foraging allocation ratio among robots*. In [29], Wenguo Liu et al, presented a simple adaptation mechanism to automatically adjust the ratio of foragers to resting robots (division of labor) in a swarm of foraging robots and hence maximize the net energy income to the swarm. Three adaptation rules are introduced based on local sensing and communications. Individual robots use internal cues (successful food retrieval), environmental cues (collisions with teammates while searching for food) and social cues (team-mate success in food retrieval) to dynamically vary the time spent foraging or resting.
• *Dynamic programmed deployable beacons.* The method described in [30] provides local rules of motion for swarm members that adhere to a global principle for both searching and converging on

a stationary target in an unknown and complex environment via the use of immobile relay markers.

The survey does not span the entire field of intelligent swarm behavior robotics. Instead, it focuses on systems for which new algorithms for communication between robots have been demonstrated. Such algorithms can be found in the work of the following researchers.

1. Algorithm for Self-Organized Aggregation of Swarm Robotics using Timer:

As a solution to self-organization among swarm agents, Xinan Yan, et al. [1] have proposed an aggregation algorithm based on some constraints for which neither central control nor information about locations of the agents are pre-given. The author's control strategy contains two states, Search and Wait for each individual robot as given in the model of probabilistic finite state automata (PFSA). Their algorithm assigns unique IDs to each robot. Knowing the total number of robots, randomly placed robots walk in the arena looking for other robots. Based on IR sensing and wireless connection capabilities installed on each robot, each can identify the others robot's ID. The group of encountered robots forms an aggregate, in which the robot with the larger ID defines the aggregate's characteristics and also insures that all robots in a particular aggregate must have the same timers. When the timer terminates, the robot tries to detach from its current aggregate. In the experiment, all the robots are identical. Each robot is mobile with limited ability of interaction including IR sensing for detecting objects and wireless communication for communicating with other robots.

2. Two foraging algorithms using only local communications

Nicholas R. Hoff et al. [31], have proposed two algorithms for searching the environment for an object of interest (food) and then returning this object to the base, keeping in mind that all robots do not have any prior information about the location of the food. Their algorithms are inspired by the foraging behavior of ants in which they mark paths leading from the nest to food by depositing a chemical pheromone on the ground. Ants use the distribution of the pheromone to decide where to move.

In their first algorithm, two simple floating-point values are used such that some robots will decide to stop their normal search and become 'pheromone robots' at any given point. Those robots will act like locations of virtual pheromones. Other robots can read the pheromone level by receiving a transmission from the pheromone robot, and they can "lay" the virtual pheromone by transmitting to the pheromone robot. So, if there were a network of pheromone robots, the walker robots could use the distribution of virtual pheromone they were able to sense in order to decide how to move. If integer values are used instead of floating-point values at each virtual pheromone such that the nearest robot to the nest stores a 1 and the other one close enough to hear the first robot stores and transmits a 2, then a walker robot can use these values to find a path to the nest by always moving to the lowest cardinality it can hear. This is the core idea of their second algorithm.

2.2. Functionally inspired robots:

Another line of swarm-based research can be found where robot agents are built to achieve specific tasks such as path finding using algorithms that are not necessarily based on imitating biological swarm organisms. In their previous work, Wang Bei, et al. [32] implemented what they call a robotic termite agent, which is able to simulate the wood-chip collecting behavior of termites. The authors have developed a software and hardware solution based on the simulation of collective building of a 2D termites' colony. The termites (swarm of robot agents) gather wood-chips into piles following a set of predefined rules. Boe-Bot Robots are used. The Boe-Bot is built on an aluminum chassis that provides a sturdy platform for the servomotors and printed circuit board and comes with a pair of whiskers and gripper. Their tasks include moving on smooth surfaces, detecting new objects, dropping the woodchips and then picking up such objects as they are encountered. The robot agent will keep on spinning left (360 degrees) until it detects an object. The robot will then carry the object and will keep holding it and moving around until it detects another object (wood-chip); it then releases it. After releasing the object, the robot moves backward, turns at an angle of 45 degrees, and the same procedure is repeated.

Obtaining decentralized control that provides interesting collective behaviors is a central problem [16, 33-41]. Several algorithms have been developed to run on swarms of robots. The complexity varies between these algorithms. Some provided basic functionality, such as dispersion, while others exhibited complex interactions between the team of robots such as bidding on tasks according to some rules. Table 1 summarizes the most recent swarm robot systems with their corresponding algorithms. These are systems introduced in literature that only involve multiple agent teams with decentralized control.

TABLE 1: MULTI-ROBOT COORDINATION APPROACHES

Approach			Remarks
Approach 1 Knowledge-based coordination			
Symprion/replicator projects	What determines the behavior of either single or group of agents is HDRC (hormone driven robot controller) controller that contains a configuration for the robot itself, and a software controller called Genome. The Genome contains a set of rules that control each agent’s behavior and generates different actions according to the different environmental conditions. Agents keep learning about their environment using internal, external and virtual sensors. Agents also are supported with on-board computational power using approaches like Generic Programming (GP) and Genetic Algorithms (GA).	Kernbach et al., 2008 [41]	The most primary advantage of this approach is the huge number of units used in the experiment. Moreover, These modules are able to reassemble different shapes that could get the whole structure moving to desired locations.
iRobot	Authors suggest spreading pheromones in an ad-hoc way over the wireless network constituted by the robots. The primary communication component is an infrared inter-robot communication. Swarm software is written as behaviors that run concurrently. Each behavior returns a variable that contains actuator commands. Their goal is to spread robots throughout an enclosed space quickly and uniformly, that were identified by direct dispersion performed by two algorithms. The first one works by moving each robot away from the vector sum of particular positions from their closest neighbors. In the second one, robots move towards areas they have yet to explore. Once the robots know their positions the frontier robots issue a message. The trees created by these messages guide the swarm toward the	J. McLurkin and J. Smith, 2004 [33]	Their solution mainly focuses on path planning and routing protocols of messages transmitted between agents at their different positions. However, the cost of individual robots and the number of robots required to provide sufficient coverage to the environment are high. However, This particular system suffers from the fact that when the ad-hoc network of robots gets partitioned, pheromone trails automatically break down causing the robots to stop moving.

	frontier robots.		
Quadrotors	Authors attempt to design small light weight flying vehicles designed to operate in close ranges. The team of quadrotors is organized into groups. Vehicles within the group are tightly coordinated and centralized control and planning is possible. The inter-group coordination is not centralized. Each group is controlled by a dedicated software node, running in an independent thread. These control nodes receive vehicle pose data from a special node via shared memory.	A. Kushleyev, et al., 2012 [42]	Quadrotors rely on an external localization system for position estimation and therefore cannot be truly decentralized
Approach 2: Auction-based coordination			
Layered architectures coordination	Authors propose auctions in which a bidding process takes place among the agents to determine who will be 'foreman' and will be in-charge for a given task and to secure teammate participation in subtasks. Tight coordination is implemented using an inexpensive reactive approach. Each robot consists of a planning layer that decides how to achieve high-level goals, an executive layer that synchronizes agents, sequences tasks and monitors task execution, and a behavioral layer that interfaces with the robot's sensors and effectors. Robots execute plans by dynamically constructing task trees.	R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith, 2000 [34]	The three robots used in this experiment are coordinated by a manipulation manager which means this is a centralized system.
ASyMTRe-D and Market-Based Task Allocation	The authors' approach is based on schemas such as perceptual and motor schemas. Inputs/outputs of each schema create what it is called semantic information that is used to generate coalitions. Tasks are assigned to the robot with the highest bid. Bids are calculated according to the costs of performing different tasks. A set of tasks is allocated to coalitions. Coalition values are calculated based on the task requirement and robot capabilities. Execution of tasks is monitored and the process of allocation repeats itself until each individual task is completed. During run-time their novel protocol ASyMTRe-D takes place. This protocol manipulates calculated coalition values to assist in completing tasks.	Tang and Parker, 2007 [43]	The advantage of this approach is that it enables robots to adopt new task solutions using different combinations of sensors and effectors for different coalition compositions. However, that solution is mainly related to computational performance where tasks are static. The authors do not mention the dynamical tasks and ways of task reassignment. Additionally, they do not discuss fault tolerance, flexibility, robustness, and how the system reacts to any robot failure.
RoboCup 2002 (Sony legged league)	Authors used wireless communication between robots in a 4-player soccer team. Each robot broadcasts a message to its teammates. This message contains the	D. Vail and M. Veloso,. 2003 [35]	Communications between robots is critical for successful coordination between robots. Local information about the

	current position of the robot and some other information about the ball in that position. All of the robots use the same set of functions to calculate real valued bids for each task. Once each robot calculates the bids for itself and each of its teammates, it compares them. If it has the highest bid for the role being assigned, it assumes that role. If it was not the winner, it assumes that the winning robot will take up the role and performs calculations for the next role in the list.		field will not be enough. This approach does not coordinate a large scale of robots.
Another application of soccer robots.	Authors use dynamic role assignment as in Robocup basing on information gathered from best behavior. Two intermediate levels have been provided to allow robot individuals to communicate. The lower level implements stigmergy (indirectly stimulating the performance of the upcoming action to provide coordination between agents) whereas, the higher one deals with the dynamic role exchange. Authors use schema-based methodology. They discuss all perceptual schemas with the required sensing, also feeding the C-implemented motor schemas which demand immediate sensor data Robots are equipped with unidirectional cameras.	E. Pagello, A. D'Angelo, and E. Menegatti,. 2006 [36]	
M+ scheme for multi robot allocation and corporation	Each robot considers all currently available tasks at each iteration. For each task, each robot uses a planner to compute its utility and announces the resulting value to the other robots. Robots negotiate which one will be in charge of performing the task. For these tasks, robots create their own individual plans and estimate their costs for executing these tasks. The robots then compare their costs to offers announced by other robots. The robot selects the task of the lowest cost that it can perform that is better than the cost announced by any other robot. Upon receipt of the other robots' utilities, each robot executes a greedy task-selection algorithm.	S. Botelho and R. Alami, 1999 [37]	Relying on Negotiation Protocols, may complicate the design of the coordinating system. Furthermore, such negotiation scenario can drastically increase communication requirements/overhead.
MURDOCH, a general task allocation system	The coordination system works using an auction protocol that allocates tasks via a sequence of first-price one round auctions. Every auction is issued by agents in five steps: task announcement, metric evaluation, bid submission, close of auction, progress monitoring/contract renewal. For each task auction, each available robot broadcasts its bid. Because of the asymmetric nature of MURDOCH's auctions, the running time varies between the bidders and the auctioneer. Authors	Brian P. Gerkey and Maja J. Mataric, 2002 [16]	M+ and MURDOCH systems assume that each robot has a single task. Each task may be performed by a single robot. This assumption proves to be oversimplified as many task domains require simultaneous work from multiple robots.

	two main testing domains were a long-term scenario consisting of many loosely coupled single-robot tasks, and a cooperative box-pushing task requiring tight coordination among the robots.		
Market-economy Approach	Authors define three strategies for exploring unvisited regions. In the first strategy namely random goal point selection the goal points are chosen at random and discarded if the area surrounding the goal point has already been visited. In the second one, the goal point is centered in the closest unexplored spot as a candidate exploration point. In the last strategy, the region is divided into its four children if the fraction of unknown space within the region is above a fixed threshold. Robots are initially placed into known positions. While running, each robot will try to sell each of its tasks to all robots with which it is currently able to communicate via an auction. If two robots lie in the same region, the robot with the highest bid wins that region's task.	R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, 2003 [38]	Authors consider regions of potential target locations for each robot and distribute tasks using bid auctions. According to some experiments performed in [44], this approach could be useful if the number of robots is small compared to the number of frontier cells. However, in the case of multiple robots this approach can be disadvantageous since a robot discovering a new frontier during exploration will often be the best suited to go on it. This can lead to an unbalanced assignment of tasks and increased overall exploration time.

3. RECONFIGURABLE ROBOTS

Reconfigurable robots automatically rearrange and change their shape accordingly to adapt themselves to different environments of application. Reconfigurable robots exhibit some features that make it possible for the robots to adapt to different tasks. For example shape shifting robots could form a worm-like shape to move through narrow spaces, and reassemble into spider-like legged robot to cross uneven terrain. Another important feature of modular robots is their potential for self repair. As the modules making a unit up are usually identical, it is possible to eliminate the damaged module and substitute it using another one, if available. Modular robots are usually composed of multiple building blocks of a relatively small repertoire, with uniform docking interfaces that allow transfer of mechanical forces and moments, electrical power, and communication throughout the robot.

According to M. Yim et al. [39], modular self-reconfigurable robotic systems can be generally classified into three architectural groups based on the geometric arrangement of their units. The first group consists of lattice architectures where robot units are arranged and connected in some regular, three-dimensional pattern, such as a simple cubic or hexagonal grid. The second group consists of chain/tree architectures where units are connected together in a string or tree topology. Finally, the third group consists of mobile architectures where units use the environment to maneuver around and can either hook up to form complex chains or lattices or form a number of smaller robots that execute coordinated movements. A respectable number of self-reconfigurable robot systems have been proposed in the last decade. Table 2 shows comparisons between the most recent ones.

TABLE 2: COMPARISONS BETWEEN EXISTING MODULAR RECONFIGURABLE ROBOT SYSTEMS

Robot	Author	Learned Pros and Cons	Software	Units	Communication	sensors
SuperBot (2006)	Shen et al.[40]	Decentralized control. Reliable Mechanical design. Limitations: Infrared sensors limit the search range and require line-of-sight between SuperBots. SuperBot architecture lacks extra actuators, grippers, and sensors for gathering information about the working environment.	Low-level programs written in C and Real-time java-based operating system	3D Modules	Infra-red and a wireless capability limited to some functions	Joint position and orientation
Molecubes (2005)	Zykov et al.[45]	Molecubes are low cost, small lattice based swarm robot with 3 DOF. Limitation: Unable to provide heavy object transport. Limited sensors. Lacks actuator mechanism.	2-D simulation	Cubes with 120 swiveling	None	None
YaMor (2006)	R. Moeckel et al. [46]	Each module comprises an FPGA for more computational power. Limitation: Uses onboard low-capacity batteries that limit the usefulness of modules. Limited sensors limit ability to sense surroundings. Only two controllable degrees of freedom.	Java-based GUI connected to robots via wireless connections	3D Chain of modules	Bluetooth	Joint position and orientation
Swarm-bot (2006)	Groß et al. [11]	Robot swarms consisting of 2 to 40 S-bots have been successfully demonstrated. S-Bots are fully autonomous mobile robots capable of self-navigation, perception of the environment and objects. Capable of communicating other S-Bots and transporting of heavy objects over very rough terrain. Limitations: Initial cost is high. Images and sound are the only way of communicating with other S-Bots. Large number of sensors and actuators consumes power, reducing functionality and operating time.	Neural Networks	S-bots with grippers	No communications occur between individual robots (S-bots) however each s-bot connects wirelessly to the PC.	Joint position and orientation
Catom (2005)	Goldstein et al. [47]	Largest actuated modules (many electromagnets on modules) Limitations: Limited sensors that have limited ability to sense	NA	3D Massive volume of agents (m$^{3)}$	This papers only presents a principle so no actual	Joint position and orientation

		surroundings.			implementation	
M-TRAN (2002)	Murata et al. [48]	Very small actuated modules, highly-robust, miniature, and reliable. Quick self-reconfiguration and versatile robotic motion. Limitations: Connection mechanism works on an internally balanced magnetic field that is not strong enough to hold the other modules. Single M-TRAN module does not have enough DOFs for switching from one posture to another form. Lack of sensors leads to mapping and control problems. Power consumption is more as it uses servo motor and electromechanical force for connectivity.	OpenGL Library, M-TRAN simulator	3D Double-Cubes	Serial bilateral communications to the PC.	Joint position and orientation
ATRON (2004)	E. H. Østergaard et al. [49]	Each module is equipped with its own power supply, sensors and actuators, allowing each module to connect and communicate with a neighbor module. Able to sense the state of its connectivity and relative motion. Limitations: Since each module includes two-axis accelerometers only, a module cannot tell if it is turned upside down or not. When two modules are connected, it's very difficult for them to move themselves, which requires cooperation from its neighbor. They are not mechanically stable and due to this mechanical instability, their electronic performance is poor.	On-board system	Lattice type units	Infra-red diodes	Joint position, orientation and proximity
PolyBot (2002)	Yim et al. [50]	First system to demonstrate the ability of self-reconfiguration with most active modules in a connected system. Each module fits within the 5cm cube. They are versatile in nature. Each module contains a Motorola PowerPC 555 processor with 1MByte of external RAM, and DC brushless motor with built in hall effect sensors. Limitations: Insufficient sensory	NA	Lattice	Infra-red Interface	Joint position, docking aid and orientation

		unit for mapping of environment. Cannot work in unknown environment with rough surface or when obstacle avoidance is not possible.				
Replicator/Symbrion(2008)	7th framework program project, European Communities [41]	Multiple processors for different tasks. Limitations: Limited to a specific task. Lack actuators and connection mechanisms for physically attaching to other modules.	On-board system	Lattice/chain	N/A	Joint position, docking aid and orientation

4. SELF-REPLICATING ROBOTS

Designing fully autonomous replicating systems did not come true until the early 2000's. An attempt to design semi-autonomous self-replicating robots that demonstrated the LEGO Mindstorm kits as a prototype capable of replication under human supervision was introduced in [51]. An autonomous self-replicating robot consisting of four low-complexity modules was presented in [52]. The authors proposed a system composed of a parent robot, four unassembled modules, and an environment in which the self-replication takes place. The authors defined two operations namely expansion and separation in which the parent robot grows itself by attaching the resource modules onto itself until it doubles its physical size, and then splits in the middle thereby returning the parent to its original state and producing one more robot. The parent robot is made of four cube-like modules connected to each other with electromagnets (EMs) installed in female and male couplers.
In [53], similar work has been done, also using unassembled components placed at certain locations on a track. The authors presented a robot that can assemble exact functional self-replicas from seven more basic parts/subsystems. The robot follows lines on the floor using light sensors and a simple control circuit without any onboard memory

5. SWARM CONTROL SOFTWARE SYSTEMS

Trifa V. et al., [54] have proposed a methodology to support standardized interfaces and communication protocols that connects robots produced by different manufacturers. To achieve this goal, the authors have used the so-called service oriented architecture (SOA) in which different software components exchange data over HTTP and then create web services (WS). The authors proposed a system that consists of four parts namely, the physical layer which contains the actual e-puck robots, the gateway layer which acts like a connection between the physical devices and the system, the logic layer containing a server that runs on J2EE, and the interface layer which provides services to the end users. In their system, any physical device or program capable of running HTTP such as PDAs, Tablet PC, and mobile phones can interact with the interface regardless of the operating system on the device. (No further explanation about control modules or how the interface looks like was given in the article). The e-puck robot –the standard one -- has eight infrared proximity and light sensors, a triangular microphone array, a speaker, a three-axis accelerometer, and a Bluetooth interface for programming. The e-puck platform can be upgraded with custom pluggable modules such as the short-range radio communication turret which provides a subset of the 802.15.4 and ZigBee protocols and is fully interoperable with the MicaZ nodes used in the physical gateway layer. However, using SOA has some performance limitations as it requires a sophisticated messaging infrastructure that would restrict the capabilities of software running on robots.

Kulis et al., [55] have proposed a software framework for controlling multiple robot agents by creating what they have named the Distributed Control Framework (DCF). DCF is an agent-based software architecture that is entirely written in Java and can be deployed on any computing architecture that supports the Java Virtual Machine. DCF is specifically designed to control interacting heterogeneous agents. DCF uses a high-level platform-independent programming language for hybrid control called MDLE. The DCF architecture consists of two distinct agents: a Robot Agent and a Remote Control Agent (RCA). The RCA lies within the human interface shown in Fig 2. Robot Agents process data from onboard hardware and from other agents, and react to perceived stimuli by selecting an appropriate behavior which is a sequence of control laws with embedded state transition logic according to a mission plan. Using the RCA, the end user can select tasks for either a robot agent or a group of agents using simple drag and drop operators. When agents are in place, a popup menu appears prompting the user to select a task. Relevant tasks for a team mission are defined in an XML configuration file which is loaded by the RCA at startup. The XML file also specifies which tasks can be performed by each agent. Authors also added a simulating feature to their RCA agent which provides a flexible numerical solving integrating system that solves differential equations for simulating a robot's kinematics/dynamics. Another feature of this system, it provides automatic updating of sensors and actuators to be distributed across multiple computing resources. The DCF currently provides drivers for a variety of robots (e.g., iRobot Creates, Pioneers, Amigobots, FireAnt, LAGR), and a wide range of sensors (e.g., digital encoders, sonars, stereo cameras, GPS receivers, inertial navigation systems, LIDARs, and cameras). Multiple efforts have been conducted as part of enhancing the DCF system. Other versions of the DCF called JAUS and TENA are being developed and tested [56].

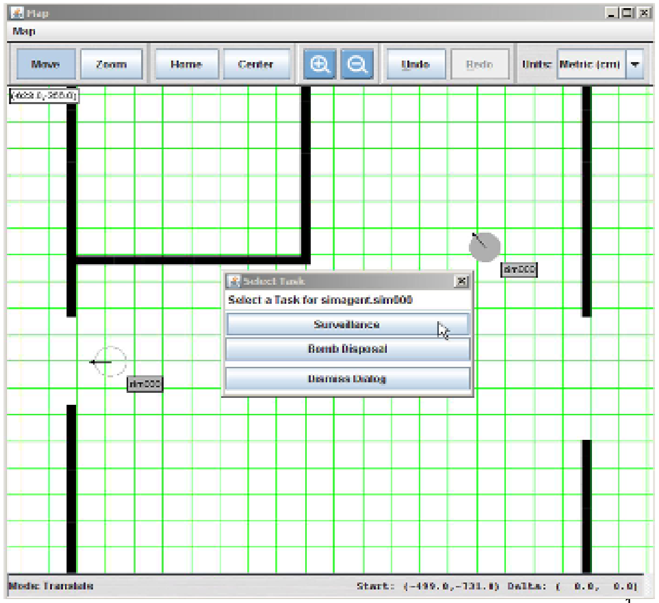
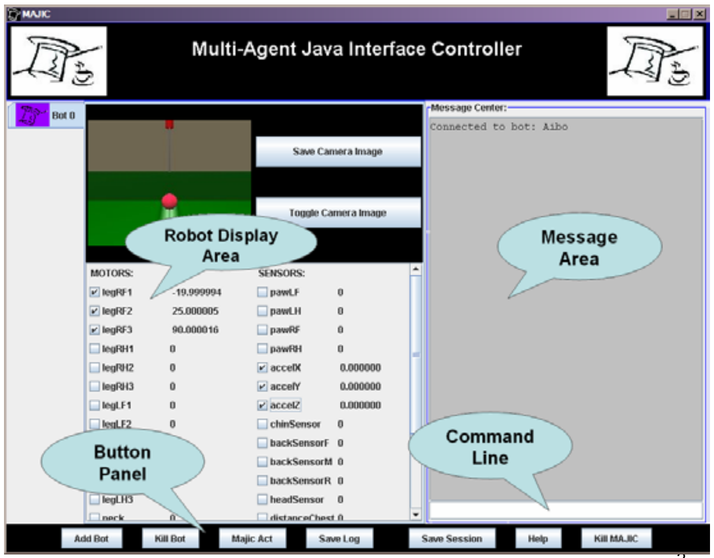


Figure 2: The DCF human interface (© 2008 IEEE)[1]

Gregory P. Ball G. et al. [8], have proposed application software built in JAVA to operate heterogeneous multi-agent robots for the sake of educational purposes named MAJIC. The system provides basic components for user interaction that enables the user to add/remove robots change the robotic swarm configuration, load java scripts into robots and so on as shown in Fig 3. The system establishes communications with built-in robot servers via a wireless connection that uses the client/server relationship. Authors described their architecture as components, consisting of one higher level component that is the GUI manager, two application logic components that consist of a Logic System to parse input into valid commands, and a Robot Server, which receives commands from the Logic System and communicates those commands to the appropriate robot. Local components communicate using direct procedure calls.

[1]© [2008] IEEE, Permission granted by Mr. Babak Sadjadi [55].



```
Connected to bot: Aibo
```

MOTORS:		SENSORS:	
✓ legRF1	-19.999994	pawLF	0
✓ legRF2	25.000005	pawLH	0
✓ legRF3	90.000016	pawRF	0
legRH1	0	pawRH	0
legRH2	0	✓ accelX	0.000000
legRH3	0	✓ accelY	0.000000
legLF1	0	✓ accelZ	0.000000
legLF2	0	chinSensor	0
		backSensorF	0
		backSensorM	0
		backSensorR	0
legLH3		headSensor	0
neck	0	distanceChest	0

FIGURE 3: The MAJIC Control Platform (© 2008 IEEE)[2]

In order to operate robots, the operator needs to write Java-embedded programs that use either the MAJIC library or Java libraries. Once a robot is connected to MAJIC, the user can immediately communicate with it from the Command Line. However, repeating this process for a team of heterogeneous robots can be impractical. The MAJIC system does not allow the user to specify the types of sensors a robot is equipped with or the type of motion model a robot's move command will utilize that could allow the user to develop more intricate behaviors with greater precision.

In [57], Patricio Nebot et al., were more interested in developing cooperative tasks among teams of robots. Their proposed architecture allowed teams of robots to accomplish tasks determined by end users. A Java-based multi-agent development system was chosen to develop their proposed platform. The authors used *Acromovi* architecture which is a distributed architecture that works as a middleware of another global architecture for programming robots. It has been implemented by means of the MadKit (Multi-Agent Development Kit) multi-agent systems framework. The graphical interface is built around pure Java Swing components, thus resulting in a cross platform application, capable of running in any operating system running the Java virtual machine.

Tao Zhang et al. [58], proposed a software platform comprised of a central distributed architecture that runs in a network environment. Their system is composed of four parts namely, user interface, controlling center, robot agent, and operating ambient making up the platform top-down. The user interface is deployed on a terminal anywhere as long as it can connect to the server where the control center is deployed. The control center provides APIs for users. The user interfaces basically communicate with the control center via a network, using TCP/UDP protocol. The authors' platform was mainly developed using Java. On the other hand, a non-distributed system architecture is used in Mobile-R [4] were the system is capable of interacting with multiple robots using Mobile-C library [59], an IEEE Foundation for Physical Agents standard compliant mobile agent systems. Mobile-R provides deployment of a network of robots with off-line and on-line dynamic task allocation. The control strategy structure and all sub-components are dynamically modified at run-time. Mobile-R provides some packages to enhance system capabilities like artificial neural networks (ANNs), genetic algorithms (GAs), vision processing, and distributed computing. The system was validated through a real world experiment involving a K-Team Khepera III mobile robot and two virtual Pioneer2DX robots simulated using the Player/Stage system.

[2]© [2008] IEEE, Permission granted by Dr. Craig Martell [8].

6. CONCLUSION

After reviewing the previous research on swarm systems we have created a basic objective to develop collective intelligence which uses small non-intelligent or slightly-intelligent robots that collectively perform complex tasks. Such smaller agents would each have location sensors, simple communication modules, and vision capability to be able to move away from each and start painting their little part of the wall in parallel. The paper provides an analysis of existing swarm systems that adhere to our criteria in an attempt to define the starting point of potential algorithms leading to the development of a new software application that aims to facilitate rapid deployment of multiple robotic agents, helps configuring these robots with dynamic task allocation and assigns the group of robots a particular task from a set of available tasks. Related work on ten swarm coordination systems were presented, compared and discussed. We also found that current swarm systems lack supporting performance data on how well the whole swarm system that is composed of either homo or heterogeneous agents will behave in performing the different tasks when varying the number of robots.

Based on this survey, design considerations leading to a new design of a swarm system platform will be presented in our future work. We will attempt to build a software environment to utilize robots that have different modular design and configuration of sensory modules, and actuators that will be implemented as a GUI interface to reduce efforts in controlling swarm robotic systems. The future application will have to offers customization for robotic platforms by simply defining the available sensing devices, actuation devices, and the required tasks. The main purpose for designing this framework is to reduce the time and complexity of the development of robotic software and maintenance costs, and to improve code and component reusability. Usage of the proposed framework prevents the need to redesign or rewrite algorithms or applications due to changes in the robot's platform, operating systems, or the introduction of new functionalities.

7. REFERENCES

1. Xinan, Y., L. Alei, and G. Haibing. *An algorithm for self-organized aggregation of swarm robotics using timer*. in *Swarm Intelligence (SIS), 2011 IEEE Symposium on*. 2011.
2. Bayindir, L. and E. Sahin, *A review of studies in swarm robotics.* Turkish Journal of Electrical Engineering, 2007. **15**(2): p. 115-147.
3. D. Rus, Z.B., K. Kotay, and M. Vona, *Self-reconfiguring robots.* Communications of the ACM, 2002. **vol. 45**(no. 3): p. 39-45.
4. Nestinger, S.S. and H.H. Cheng. *Mobile-R: A reconfigurable cooperative control platform for rapid deployment of multi-robot systems*. in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011.
5. Fong, T., I. Nourbakhsh, and K. Dautenhahn, *A survey of socially interactive robots.* Robotics and autonomous systems, 2003. **42**(3): p. 143-166.
6. Sugawara, K. and T. Watanabe. *Swarming robots-foraging behavior of simple multirobot system*. in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. 2002. IEEE.
7. Szu, H., et al. *Collective and distributive swarm intelligence: evolutional biological survey*. in *International Congress Series*. 2004. Elsevier.
8. Ball, G.P., et al. *MAJIC: A Java application for controlling multiple, heterogeneous robotic agents*. in *Rapid System Prototyping, 2008. RSP'08. The 19th IEEE/IFIP International Symposium on*. 2008. IEEE.
9. Dai, H., *Adaptive Control in Swarm Robotic Systems.* The Hilltop Review, 2011. **3**(1): p. 7.

10. Sayouti, A., H. Medromi, and F. Moutaouakil, *Autonomous and Intelligent Mobile Systems based on Multi-Agent Systems.*
11. Groß, R., et al., *Autonomous self-assembly in swarm-bots.* Robotics, IEEE Transactions on, 2006. **22**(6): p. 1115-1130.
12. Mondada, F., et al., *SWARM-BOT: A new distributed robotic concept.* Autonomous Robots, 2004. **17**(2): p. 193-221.
13. Fukuda, T. and S. Nakagawa. *Dynamically reconfigurable robotic system*. in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. 1988. IEEE.
14. Mohan, Y. and S. Ponnambalam. *An extensive review of research in swarm robotics*. in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. 2009. IEEE.
15. Dunbar, T.W. and J. Esposito. *Artificial potential field controllers for robust communications in a network of swarm robots*. in *System Theory, 2005. SSST'05. Proceedings of the Thirty-Seventh Southeastern Symposium on*. 2005. IEEE.
16. Gerkey, B.P. and M.J. Mataric, *Sold!: Auction methods for multirobot coordination.* Robotics and Automation, IEEE Transactions on, 2002. **18**(5): p. 758-768.
17. Reynolds, C.W., *Flocks herds and schools: A distributed behavioral model.* Computer Graphics, July 1987(Volume 21 Issue 4): p. 25-34.
18. Hayes, A.T., A. Martinoli, and R.M. Goodman. *Swarm robotic odor localization*. in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. 2001. IEEE.
19. Payton, D., et al., *Pheromone robotics.* Autonomous Robots, 2001. **11**(3): p. 319-324.
20. Mayet, R., et al., *Antbots: A feasible visual emulation of pheromone trails for swarm robots.* Swarm Intelligence, 2010: p. 84-94.
21. Svennebring, J. and S. Koenig, *Building terrain-covering ant robots: A feasibility study.* Autonomous Robots, 2004. **16**(3): p. 313-332.
22. Russell, R.A. *Ant trails-an example for robots to follow?* in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999. IEEE.
23. Russell, R.A. *Heat trails as short-lived navigational markers for mobile robots*. in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. 1997. IEEE.
24. Mamei, M. and F. Zambonelli. *Spreading pheromones in everyday environments through RFID technology*. in *2nd IEEE Symposium on Swarm Intelligence*. 2005.
25. Psaier, H. and S. Dustdar, *A survey on self-healing systems: approaches and systems*. Computing, 2011. **91**(1): p. 43-73.
26. Gazi, V. and K.M. Passino, *Stability analysis of social foraging swarms.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2004. **34**(1): p. 539-557.
27. Vaughan, R.T., et al. *Blazing a trail: insect-inspired resource transportation by a robot team*. in *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS), Knoxville, TN*. 2000.

28. Blow, M. *'stigmergy': Biologically-inspired robotic art*. in *Proceedings of the Symposium on Robotics, Mechatronics and Animatronics in the Creative and Entertainment Industries and Arts, The Society for the Study of Artificial Intelligence and the Simulation of Behaviour*. 2005.
29. Liu, W., et al., *Towards energy optimization: Emergent task allocation in a swarm of foraging robots.* Adaptive Behavior, 2007. **15**(3): p. 289-305.
30. Barth, E.J. *A dynamic programming approach to robotic swarm navigation using relay markers*. in *American Control Conference, 2003. Proceedings of the 2003*. 2003. IEEE.
31. Hoff, N.R., et al. *Two foraging algorithms for robot swarms using only local communication*. in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. 2010.
32. Wang, B., et al. *An Experimental Collective Intelligence Research Tool*. in *proceedings of the Fourth International ICSC Symposium on Engineering in Intelligent Systems (EIS 2004), Madeira, Portugal*. 2004.
33. McLurkin, J. and J. Smith, *Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots.* Distributed Autonomous Robotic Systems 6, 2007: p. 399-408.
34. Simmons, R., et al., *First results in the coordination of heterogeneous robots for large-scale assembly*. Experimental Robotics VII, 2001: p. 323-332.
35. Vail, D. and M. Veloso, *Multi-robot dynamic role assignment and coordination through shared potential fields*. Multi-Robot Systems, 2003: p. 87-98.
36. Pagello, E., A. D'Angelo, and E. Menegatti, *Cooperation issues and distributed sensing for multirobot systems.* Proceedings of the IEEE, 2006. **94**(7): p. 1370-1383.
37. Botelho, S.C. and R. Alami. *M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement*. in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999. IEEE.
38. Zlot, R., et al. *Multi-robot exploration controlled by a market economy*. in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. 2002. IEEE.
39. Yim, M., et al., *Modular self-reconfigurable robot systems [grand challenges of robotics]*. Robotics & Automation Magazine, IEEE, 2007. **14**(1): p. 43-52.
40. Salemi, B., M. Moll, and W.M. Shen. *SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system*. in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. 2006. IEEE.
41. Kernbach, S., et al. *Symbiotic robot organisms: REPLICATOR and SYMBRION projects*. in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*. 2008. ACM.
42. Kushleyev, A., D. Mellinger, and V. Kumar. *Towards a swarm of agile micro quadrotors*. in *Robotics: Science and Systems (RSS)*. 2012.
43. Tang, F. and L.E. Parker. *A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation*. in *Robotics and Automation, 2007 IEEE International Conference on*. 2007. IEEE.
44. Burgard, W., et al., *Coordinated multi-robot exploration.* Robotics, IEEE Transactions on, 2005. **21**(3): p. 376-386.

45. Zykov, V., et al., *Evolved and designed self-reproducing modular robotics.* Robotics, IEEE Transactions on, 2007. **23**(2): p. 308-319.

46. Moeckel, R., et al., *Exploring adaptive locomotion with YaMoR, a novel autonomous modular robot with Bluetooth interface.* Industrial Robot: An International Journal, 2006. **33**(4): p. 285-290.

47. Goldstein, S.C., J.D. Campbell, and T.C. Mowry, *Programmable matter.* Computer, 2005. **38**(6): p. 99-101.

48. Murata, S., et al., *M-TRAN: Self-reconfigurable modular robotic system.* Mechatronics, IEEE/ASME Transactions on, 2002. **7**(4): p. 431-441.

49. Østergaard, E.H., et al., *Design of the ATRON lattice-based self-reconfigurable robot.* Autonomous Robots, 2006. **21**(2): p. 165-183.

50. Yim, M., Y. Zhang, and D. Duff, *Modular robots*. Spectrum, IEEE, 2002. **39**(2): p. 30-34.

51. Suthakorn, J., Y.T. Kwon, and G.S. Chirikjian. *A semi-autonomous replicating robotic system*. in *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*. 2003. IEEE.

52. Lee, K. and G.S. Chirikjian. *An autonomous robot that duplicates itself from low-complexity components*. in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2010. IEEE.

53. Liu, A., et al. *A memoryless robot that assembles seven subsystems to copy itself*. in *Assembly and Manufacturing, 2007. ISAM'07. IEEE International Symposium on*. 2007. IEEE.

54. Trifa, V.M., C.M. Cianci, and D. Guinard. *Dynamic control of a robotic swarm using a service-oriented architecture*. in *13th International Symposium on Artificial Life and Robotics (AROB 2008)*. 2008.

55. Kulis, Z., et al. *The distributed control framework: a software infrastructure for agent-based distributed control and robotics*. in *American Control Conference, 2008*. 2008. IEEE.

56. Lenzi, N., B. Bachrach, and V. Manikonda, *DCF (Registered)-A JAUS and TENA Compliant Agent-Based Framework for Test and Evaluation of Unmanned Vehicles*, 2011, DTIC Document.

57. Nebot, P. and E. Cervera. *Agent-based application framework for multiple mobile robots cooperation*. in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2005. IEEE.

58. Tao Zhang, X.L., Yi Zhu, Xiaqin Li, Song Chen, *Coordinative Control for Multi-Robot System through Network Software Platform.* iConcept Press, 2010(28): p. 51-59.

59. Chen, B., H.H. Cheng, and J. Palen, *Mobile-C: a mobile agent platform for mobile C/C++ agents.* Software: Practice and Experience, 2006. **36**(15): p. 1711-1733.