Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2014
August 17-20, 2014, Buffalo, New York, USA

DRAFT PAPER – DETC2014-34659

TOWARD A DESIGN ONTOLOGY FOR SELF-ORGANIZING SYSTEMS

James Humann

IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, California 90089-1453
humann@usc.edu

Yan Jin*

IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, California 90089-1453
yjin@usc.edu (**Corresponding author*)

ABSTRACT

Many designers have demonstrated self-organizing systems of simple agents performing useful collective tasks such as shape formation or foraging. These systems draw inspiration from biology, engineering, economics and other diverse research areas. The great diversity of approaches has led to a large body of applications and simulations. The lack of unity among different approaches, however, leads to challenges in system integration, knowledge transfer, and modeling. With a shared vocabulary, designers could create a common conceptualization of the self-organizing systems design process and more easily share discoveries across application domains. To this end, we propose an ontology that covers the design and deployment of self-organizing systems with an emphasis on function, form, and behavior at the agent, group, and system levels of analysis, with particular emphasis on the behavioral design of agents.

INTRODUCTION

The theory of design of self-organizing (SO) systems is still in its nascent stages. Many designers are interested in the function, behavior, and form of such systems, but do not have a large bank of analogous systems to refer to when mapping between behavior and form, or between local and global levels of system behavior. Although various SO systems have been demonstrated, most researchers have taken an *ad hoc* approach to their design, demonstrating specific successes but not general methodologies [1]. Ontologies are useful tools for organizing a domain of knowledge. With a regimented ontology of the SO system design process, various successful systems can be categorized, and their important self-organizing mechanisms can be documented for possible adoption for other, similar functions.

An ontology is a knowledge structure applied to a domain of interest. Ontologies define formal languages, easing information transfer and clarifying semantics. They are abstract, detailing only the important (to specific users) details of a domain, and should be "explicit," giving a precise definition of the concepts and relationships contained within the ontology [2]. Ontologies are often used for consistency in knowledge transfer between cooperating entities, whether they be humans [3] or computer agents [4].

In our research, as an attempt to organize lessons learned by diverse studies on the use of self-organization, with a particular emphasis on the use of self-organization as a tool in engineering design, we are developing ontology for designing self-organizing systems. This paper reports our initial step in building a FBS (function, behavior and structure) based ontology in the self-organizing systems context. This ontology draws a distinction with the traditional design process by focusing on the three levels of system architecture and defining the key characteristics of function, behavior and structure at these levels. By enumerating the key concepts in the self-organizing system design process, we are able to identify the mappings between certain concepts where most the design work takes place, regardless of the system's application domain or the designer's field of expertise. A case study on the design of a self-organizing foraging system using the mappings described in the ontology is presented along with example applications of the terminology to self-organizing systems from the literature.

RELATED WORK

Researchers are currently making a similar effort in the biological sciences, and their work can inform our own due to similarities that self-organizing systems show to biological systems, and because their goals of knowledge capture and

transfer are similar to our own. In this section we present a brief review covering a small portion of the research on artificial SO systems that drives this work, inspiration from parallel efforts in the biological sciences, and an introduction of existing ontologies of engineering knowledge and their use in design.

Artificial Self-Organizing Systems

SO systems have the potential to display the adaptability of natural systems while performing useful engineering functions. Generally, they are composed of many interacting agents or robots. Their lack of fixed connections allows for architectural flexibility, and their redundancy at the agent level makes them resilient to partial system failures. The adaptability comes at a cost though, as these systems are not as easily controlled or efficient as monolithically engineered systems [1]. In highly regimented and repetitive tasks, the designer will probably not choose to trade off efficiency for adaptability, but in other more complex tasks, it may be advantageous, or even necessary.

Self-organization has been demonstrated as an effective mechanism for robots to collectively gather pucks on a field into a central location with no need for communication with one another or an *a priori* determination of the gathering point [5]. Werfel and Nagpal use self-organizing robots to assemble simple shapes with special bricks [6], and Shen [7] has designed a robot made of self-organizing modules capable of reconfiguration and locomotion. Other artificial SO systems that have been proposed or simulated include micro-scale construction [8], shape formation [9], and self-assembling origami [10]

In our previous work, we focused on the design, modeling, and simulation of Cellular Self-Organizing Systems, where the word cellular refers to the fact that these systems are built bottom-up from simple components that do not have great functionality of their own. We have analyzed these systems for their ability to form and reconfigure into shapes [11], organize to cooperatively forage or push objects toward goals [12], [13], and display fast changes in system-level behavior through parameter changes in agent behavioral models [14].

Design of SO systems is difficult because the designer only has direct control over the agents, and the higher level functionality emerges from agent interactions. The focus of the design is more about endowing the agents with fundamental capabilities and creating a rich search space of emergent behavior, from which a suitable behavior will be found [15]. Also, the analysis of SO systems can be extremely complex, even if its constituent agents are quite simple due to the massive interconnections among agents [16].

Ontology in the Biological Sciences

Over 16,000 new species are described in the biology literature annually [17], overwhelming the ability of any single researcher to comprehend them all. These data are often recorded in terms with definitions that may vary between countries, or even between research groups. Biologists are increasingly turning to the use of ontologies to categorize important aspects of the species being described for sharing with other researchers and future data mining [3]. One example is the anatomical ontology of the genus *Hymenoptera* (ants, wasps, bees, etc.) that gives distributed researchers a language to describe their subjects in a such a way that everyone agrees on the interpretation [18]. This also allows comparisons between different research groups, and between living and extinct species of insects. Although not mentioned, from an engineering perspective it is interesting to note that such structural ontologies could also enable the "design" of new, hypothetical hybrid insects, since they are a store of knowledge about insect parts and rules for composing them.

The Gene Ontology Consortium [19] is an ambitious plan to record knowledge about the location and structure of genes on the DNA strands of various organisms, the biochemical processes that these genes govern, and the ultimate effect of these processes on the higher-level needs of the organism. These three aspects of DNA were chosen with the intention of identifying the role of proteins in model organisms and thus inferring its role if found in others [20]. This knowledge of the relationships between structure and higher-level functions is also necessary in an ontology of artificial self-organizing systems.

Ontology in Engineering Design

Engineering design is fundamentally an exercise in information processing. Thus, having the information structured in ontological form can be very valuable to the process. Various researchers have proposed ontologies for concepts important to engineers. Gero has proposed an ontology of artifacts described by their function, behavior, and structure [21], which can be used to describe the various states and transformations of the design process [22]. The Functional Basis [23] is a detailed exploration of the definition of function, trying to identify a "minimal set" of terms for engineers to use during functional design, to ensure common levels of specificity between different designers. With common decompositions of function, and design repositories of successfully engineered structures that fulfill those functions, automated search can be used to generate structural designs for engineering functions [24], [25]. If the same engineering terms are used to describe natural systems, these knowledge stores can even be used as aids to generate design concepts by analogy to biological systems [26].

A DESIGN ONTOLOGY FOR SO SYSTEMS

Function, behavior, and structure (FBS) are often used to classify engineered systems. Gero's FBS ontology [21] of artifacts defines function as the purpose of an object, what it is for. Structure describes the form of the object, what it is. Behavior describes what the object actually does, and it can be determined from the structure and application of engineering analysis. A designer specifies the structure of an artifact, with the intention that its behavior will fulfill its function. We use the familiar terms function, behavior, and structure in our ontology, but with modified definitions given in the following sections to account for the unique scenario of self-organizing systems design.

The goal of the design process is to specify a product for an end user. For our work, we define "product" as the output of the design process, a specified arrangement of self-organizing agents with defined behavior. The arrangement includes both the number and initial conditions of the agents. The form and behavior of the agents are specified during the design, but the final form and behavior of the product are the result of self-organizing processes and are only evident at runtime.

Three Levels

Self-organizing systems derive their power from their ability to self-assemble from primitive components. The ontology captures this defining characteristic by recognizing three different levels of system architecture: the overall system level, the atomic agent level, and an intermediate group level.

- The *system level* is familiar to engineers. It includes the product in its entirety. The function at this level is the end user's ultimate concern.
- The *agent level* describes the function and behavior of the smallest units within the system, such as a robot in a robot swarm.
- The *group level* describes an organization of several agents whose collective behavior accomplishes some function that an agent could not accomplish individually.

The group level cannot be defined without a decomposition of the system function. The group's function does not entirely fulfill the system's function, but does fulfill some sub-function decomposed from the group function. Certain sub-functions must be fulfilled by other groups or individual agents. A group (by definition) must contain more than one agent, and should not include every agent, as this would make it indistinguishable from the system.

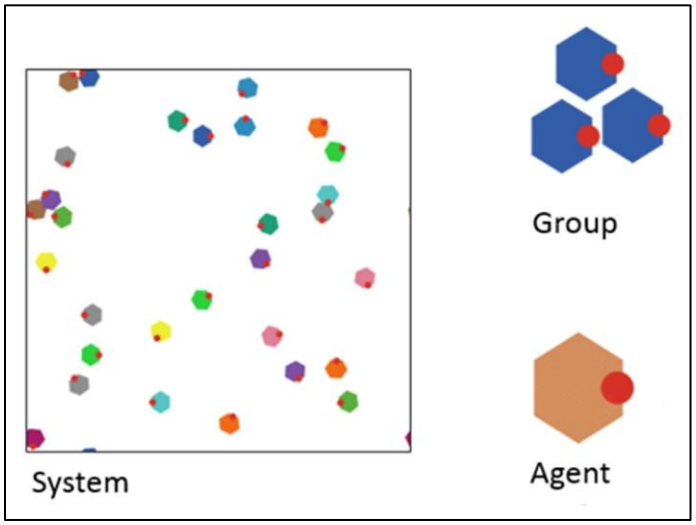

Figure 1: Three architectural levels in self-organizing systems, the agent, group, and system levels

FBS in Self-Organizing Systems

The product is designed to perform a function or functions. As in [27], we define "function" as an "intended change between two scenarios" that is the result of the implementation of the design. Intent is important to the definition, as products may produce changes that are unintentional, and these are not described as the function of the system. Intentions can originate with customers as requirements, or with the designer as expectations.

This definition is not unique to self-organizing systems, but definition of functions at lower levels of decomposition will require more refinement to account for the peculiarities of SO systems. For example, in SO systems, behavior that was unintentional may in fact show emergent effects that are highly beneficial to the system. In this instance, the designer will redefine the side effects as functions and seek ways to augment them. Functions also may not map cleanly to components as suggested by many prescriptive design methodologies (e.g. [28], [29]).

Reynolds' Boids self-organized flocking simulation [30] is an example of taking a system-level functional requirement to imitate flocking (the paper was written from a computer graphics perspective) and decomposing it into centering, velocity-matching, and collision avoidance at the local level. In other systems, agent-level functions will not be directly derivable from the system-level function. Figure 2 gives an illustration of a system level functional requirement that has been decomposed into sub-functions at the agent level. Some can be derived directly from the top-level requirements, while others must be derived from intermediary group-level requirements.

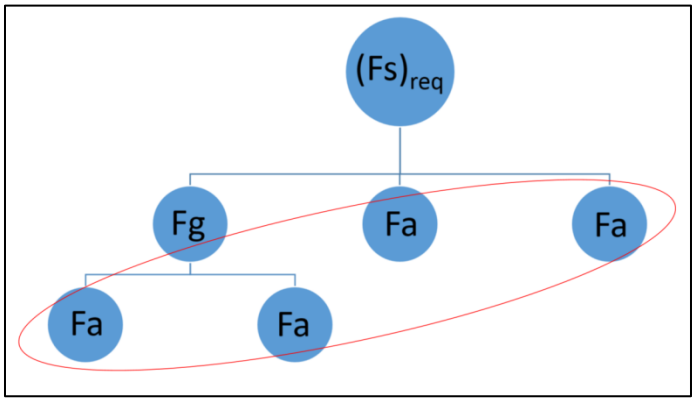

Figure 2: A system-level functional requirement mapped to two agent level functions directly, and to two more through a group level function

In contrast to what a system should do, "behavior" is defined as what a system *actually* does [22]. The concept of behavior is where the design of self-organizing systems differs most significantly from the traditional view of design. Since the agents that form the basis of the system are not inert components, but are in fact active entities, their behavior is more dependent on their internal control logic than on the physical forces surrounding them. In this way, the analysis of their behavior is more similar to the analysis of a computer program than of a cantilever beam. Agent behavior consists of two parts:

- The agent *capacity* is the set of behaviors that an agent is capable of performing.
- The agent *behavioral selection* is the internal decision algorithm determining which behavior from the capacity an agent performs at a given time.

"Structure" describes what the artifact actually is in a tangible sense. The structure of an individual agent is fairly simple to describe with standard engineering blueprints and bills of materials, but higher-order structures must be treated differently in our ontology.

Structure may exist at various levels within a self-organizing system as groups of agents form and disband at runtime. The global structure of SO systems can be very fluid, as in a school of fish, or can be more rigid, as in swarms of cooperative robots [8]. The bridge and raft structures formed by certain species of ant have been called "living architecture" [31] due to their ability to re-form and adapt to a dynamic environment, and are structures found at the group level, partially fulfilling the system-level goals of foraging and escaping predators.

Domains

Knowledge about function, behavior and structure is generated and refined throughout the design process. To accurately capture and classify this knowledge, it is sometimes necessary to categorize its source. For our ontology, we use three domains:

- The *expected domain* [22] covers design states that the designer intends or anticipates to be.
- The *required domain* covers design information that the customer imposes on the designer. The most important is the system-level functional requirement, but there may be others such as constraints on the agent-level structure.
- The *simulated domain* is where the designer evaluates computer simulations of design decisions.

The specific inclusion of a simulated domain is a recognition of SO system engineers' reliance on computer simulations. Because of the highly complex interactions among system components, multi-agent simulations are a near-necessity in their analysis. These domains are sufficient to categorize the systems we have reviewed, and more may be added as future work.

Important Mappings

Throughout the design process, the designer will map the state of the design between different domains and levels. In this section, we identify some of the key mappings and define them from the perspective of SO system design.

Function decomposition is a fundamental process of engineering design that forms the basis of most prescriptive design methodologies [29], [32]. It is the decomposition of a function into sub-functions that will fulfill the top-level function. This decomposition is usually done to transform a large intractable problem into a series of smaller, more manageable problems. In SO systems, system-level functions rarely map cleanly to sub-functions. It may be necessary to design for emergent system-level functions, and use a combination of agent-level and group-level functions in the decomposition.

Capacity fulfillment is a mapping from agent functions to agent behavioral capabilities. The output of this mapping is an agent design with the necessary behavioral repertoire to fulfill the required and expected agent functions. This is not the specification of the actual behavior of the agent, only its minimum necessary capabilities. For example, in Trianni's work on a cooperative lifting task [33], the basic capacity of attaching to the object to be lifted was included in the design of the agents, while the interactive mechanisms for cooperative lifting were left for later design.

Local-to-global analysis is the mapping from agent behavior to social and system behavior, and is one of the main challenges for understanding self-organizing systems. Agent behaviors cause the formations of groups, and the interactions of groups and individual agents give rise to the system-level behavior. The system and group-level structures and behaviors can in turn effect the agent behavior, resulting in many complex feedback loops that are difficult to comprehend using standard mathematical analysis, so we suggest multi-agent simulation to perform this necessary mapping [34].

The *synthesis* process of mapping an agent's capacity to its structure is a straightforward traditional design process (which is not to say that it is trivial, just that it is a problem that engineers have traditionally excelled in). For example, if an agent's capacity is to move in any direction on a plane, it must be equipped with actuators that are capable of driving and steering the agent. If an agent's capacity is to react to other agents, it must have the hardware to sense the presence of other agents.

We argue that *behavioral design* at the agent level is the most important process in the design of self-organizing systems. This is the mapping of a system-level functional requirement to an agent-level behavioral capacity and selection. The designer must rely on a combination of traditional decomposition, analogy, and intuition to map from global function to local behavior. By choosing to use self-organization, the designer has intentionally given up some control of the group-level and system-level dynamics [35]. So we see that the structure of the agent should follow from the behavior, and the structure and behavior of the higher levels are outside the designer's direct purview. Specifying the behavior at the agent level is where the designer's influence is most evident, and where the design effort should be focused.

Agent Parametric Behavioral Model

In our previous work [13] we introduced a two-field based parametric behavioral model (PBM) to aid in agent-level behavioral design. The agent-level behavior that we defined in this ontology is not physics-based, but is based on the decision logic of the elementary SO units. This definition is a reflection of a unique characteristic of the design of SO systems: the self-

organizing units are rather simple in hardware, and the most significant design work is focused on creating behavioral rules and interaction mechanisms.

This PBM is a reaction to an agent's sense of (natural or artificial) fields in its vicinity. A task field [12] includes all the relevant task objects and goals of an agent's function, and a social field controls agents' interactions with one another. The distinction between task and social fields is made because it is social relations that lead to the formation of groups and structure, whereas task-based rules result in the efficient fulfillment of functions, which may depend on the prior formation of group structure.

We call this a *parametric* behavioral model, because the initial effort should be focused on creating a behavioral design space, rather than a point design. Because emergent behavior is difficult to design without iteration, the design space is necessary to allow for further trial-and-error or optimization processes. For example, a very simple agent could have the capacity to step forward and turn, with parameters governing the magnitude of the step and turning angle. If the step and turning angle significantly affect the global function of the system, then the PBM is a search space, which the designer can explore by varying the two parameters to find proper system behavior.

This can be accomplished for general systems if an agent's decision algorithm is based on parameters whose allowed values will always yield a working (though not necessarily productive) decision algorithm. Properly formulated, a PBM should always contain a valid decision structure as long as the input parameters are within the specified range. This separates the parameters from the decision structure, thus ensuring that the structure is valid throughout the optimization process.

Notation

As in [22], Function, Behavior, and Structure will be represented by capital F, B, and S. The organizational levels will be denoted by a subsequent lowercase s, g, or a, to indicate whether the concept is at the system, group, or agent level. Capacity (agent-level only) is represented by Ca. If it is necessary to indicate the domain of a state in the design process, the state's symbol will be enclosed in parentheses with the necessary information afterwards. The subscripts "sim, req, and exp" will be used in this paper for simulated, required, and expected. For an example of the notation used, the behavior of a group of agents seen in a simulation would be denoted as $(Bg)_{sim}$. The system-level function of a product, where the domain is clear from context or irrelevant, would be denoted as Fs.

CASE STUDY

In this case study, we present an expansion on the design of a foraging system from our previous work [13]. We present the design process as a series of mappings between different states identified in our ontology, and describe the strategies used and difficulties encountered while performing these mappings. An overview of the design process is presented in Figure 3.

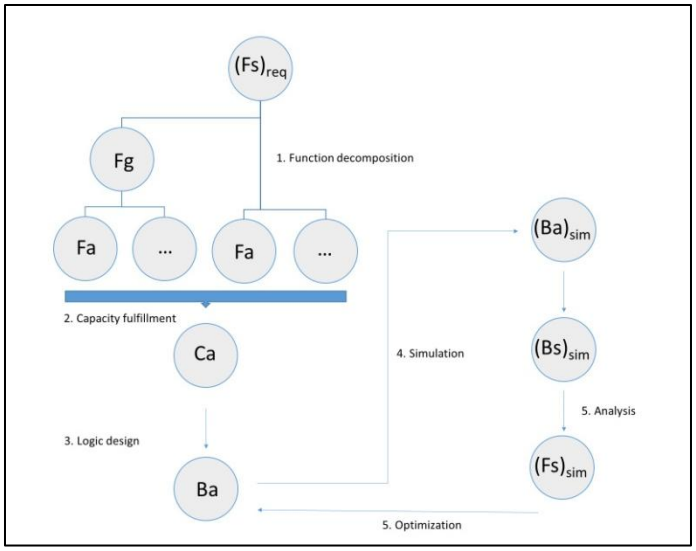

Figure 3: Foraging system design process

Foraging

Foraging is the act of finding an important resource and returning it to a specified location for storage. Social insects present examples of advanced foraging behavior from the interactions of simple agents. The study of ants [36], [37] in particular has given biological inspiration to SO search methods. Social insects remember and indicate the location of food by initially searching randomly until they find food, and then laying down pheromones as they return the food back to their nest. These are chemical markers that other insects from their colony can detect, so insects seeking food can sense and follow the trails left by those who have already successfully found it, and then reinforce the pheromone trail on their return trip. Using a similar approach, we attempt to design a system of agents which can find and retrieve food from a location far away from the home base.

Assumptions

For practical purposes, self-organizing systems will not be economically competitive with most other systems unless they are based on very simple hardware. This simplicity allows for mass-scale manufacturing, and little economic loss from the failure of any single unit. This also causes a distribution of functionality and redundancy throughout the system, which may be necessary in the design of self-organizing systems which are to be deployed at very small scales [8], in huge numbers [38], or in hostile environments [34]. We use the assumptions of $(Ba)_{req}$ limited sensory radius and $(Sa)_{req}$ simple hardware found in our previous work on Cellular Self-Organizing Systems [11], [39]. The result of these assumptions is that we restrict ourselves to the design of agents who can only sense each other, environmental objects, and food within a local radius, and have very little onboard memory. Agents can sense the direction toward their home base at all times, simulating the condition of a central location sending out a strong homing beacon.

Function Decomposition

The functional requirement $(Fg)_{req}$ of the foraging system is to "fetch food." The Fg of fetching food could be decomposed into four Fa of "find food, pick up food, carry food to home, and drop food." This is how many animals that stockpile food, or in fact humans, operate, and all of these functions are within the capability of a single agent. We know, however, from the study of social insects, that the function of "find food" can be made much easier if there is corresponding function of "indicate food location". The food may be a relatively large distance away from the home base. This distance is many times the radius of an agent's sensory capabilities, so directing agents between the food and home is outside of the capabilities of a single agent. This cannot be Fa, and must be expected at a higher level of Fg. The use of pheromones is not realistic in a robotic system, but the social insect can be used as analogy for implementing state change in the Ba. To achieve the Fg of indicating location, it becomes necessary for individual agents to change their behavior based on whether or not they have found food. From the Fg, we then derive the Fa of indicate state (and corresponding sense state).

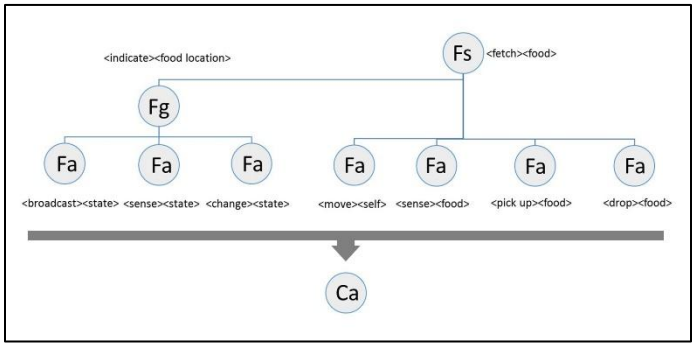

Figure 4: Function decomposition in a foraging system

Behavioral Design

The decomposed Fa related to picking up, moving, and dropping food can be directly mapped to Ca as they would be in a traditional design process. The Fa derived from Fg "indicate food location" require social interactions among agents and, once identified, can also be mapped to Ca. This capacity fulfillment is shown in Figure 4.

There was an expectation that flocking would be a useful and efficient way for the agents to explore the field, so social flocking behaviors such as cohesion (attraction), nonlinear avoidance, and alignment were also added to the Ca to fulfill this $(Bg)_{exp}$. There were also parameters governing the intensity of an agent's attraction toward food (when sensed) and home. These capacities for action combine to become the total capacity of an agent, and the manner of selection among these possibilities is the Ba. The Ba is a field-based parametric behavioral model with 18 parameters to govern the movement of agents.

The equation for generating the social field is given in (1) and (2):

$$FLDs(r,\theta,\phi) = C \times \frac{-1}{N}\sum_{i\in\eta}^{N} r_i + O \times \frac{-1}{N}\sum_{i\in\eta}^{N}\frac{1}{r_i} + A \times \frac{1}{N}\sum_{i\in\eta}^{N}\|v_i\|\cos(\theta-\phi) \quad \textbf{(1)}$$

$$FLDt(r,\theta,\phi) = step \times [\mathrm{F}\times\cos(f-\phi) + H\times\cos(h-\phi)] \quad \textbf{(2)}$$

where r_i is the distance from an agent to its neighbor, ϕ is the agent's current angular heading, θ is the neighbor's current heading, $step$ is the maximum step size, f is the angle toward food (if nearby), and h is the angle toward home. C, O, A, F, and H are the behavioral model's parameters.

C, O, and A each need 4 parameters for four possible scenarios depending on whether or not the agent has food, and whether or not its neighbor has food. F and H each required two parameters for the states of an agent being with or without food. Finally, there were 'R' parameters for random stepping behavior with our without food. An agent would calculate the sum of the two field values for all points within its maximum stepping radius, and move to the point with the highest field value at each timestep. An agent's behavior to pick up and drop food was not parameterized. It was automatically triggered as an agent moved to a patch of land containing food or its home base. Every agent applied this behavior in parallel at each timestep.

Simulation and Optimization

Any particular set of 18 parameters fixes a Ba, which can be simulated in a NetLogo [40] multi-agent simulation to produce $(Bg)_{sim}$ and $(Bs)_{sim}$. For the simulation, a system of 30 agents was placed on a field between their home base and a food source. At each timestep, every agent would sense its local neighborhood, and apply the decision algorithm of its Ba. If an agent found the patch containing food, it would pick up and carry five units of food. This was repeated for 1000 timesteps, and a fitness score was used to judge the $(Fs)_{sim}$ at the end of a simulation run. This fitness score can be calculated from (3).

$$\text{fitness} = \text{food}_{\text{r}} + \frac{1}{\text{N}}\sum_{\text{i}=1}^{\text{N}}\text{food}_{\text{c}_\text{i}} \quad \textbf{(3)}$$

In this equation, the subscript 'r' stands for the food returned home by the final timestep, and the subscript 'c' stands for food that is being carried at the final timestep, but was not returned. The summation term is carried out over every agent in the system.

A genetic algorithm (GA) was used to optimize the parameters of the PBM. Genetic algorithms are partially stochastic search algorithms that work by mutating and recombining partial solutions to optimization problems [41]. They work on a population of candidate solutions, and the probability of selection for recombination or mutation is proportional to a candidate solution's fitness. A reformulation Ba to Ba was iteratively performed by the GA by generating different candidate solutions (each candidate consisting of a set of 18 parameters from the PBM), generating $(Ba, Bg, Bs)_{sim}$ and selecting candidate solutions for recombination based on the fitness function. This GA was run for 50 generations with a population of 50 solution candidates. The optimized Ba at the

final generation resulted in a system that would make 38 round trips during 1000 timesteps (returning 190 units of food).

The $(Bs)_{sim}$ that was found did in fact fulfill the function to fetch food. The strategy employed by the system was to form lines of agents along the boundary of the arena, which would funnel the individual gatherers toward the food, as seen in Figure 5. These lines were necessary, since an agent would simply get stuck on the boundary and stop foraging if it were not for the social influences of other agents prodding them to keep moving. This $(Bg)_{sim}$ was actually a surprising repurposing of the original flocking behaviors included in the Ca. The alignment behavior was originally included with the $(Bg)_{exp}$ of flocking, for efficient group movement. In fact, the GA evolved a large negative value for the alignment parameter, causing agents without food to set their heading opposite the heading of their neighbors. This ensured an unstable condition when agents would get stuck near one another at the wall. One would have to turn around and face back away from the wall. With enough agents forming a line at the wall, this behavior ensures that the entire system cannot get stuck.

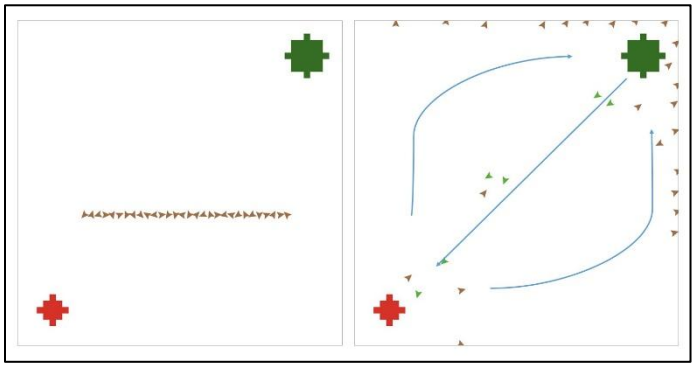

Figure 5: Initial conditions (left) and emergent behavior (right) of foraging simulation. The food is at the top-right, and the home base is at the bottom left. Note: size of agents is enlarged for presentation.

Reformulation of Ca

The lines at the edge were the optimized foraging strategy found within the PBM given to the GA, but the use of so many static agents as barriers is inefficient, because at any given time there may be only a fraction of the agents actually moving between food and home. This required a reformulation of a more fundamental nature, a reformulation of Fg and Fa, and thus new elements added to Ca.

The primary new element of the agents' Ca was the addition of boundary detection capabilities, and the Ba included a parameterized attraction or repulsion from the boundary. With this addition, the optimized $(Bs)_{sim}$ did not drastically change. Lines of agents still formed near the edges, but instead of forming directly on the edge, they maintained a slight equilibrium distance, allowing much smoother flow of agents toward the food. This small addition to Ca resulted in a much more productive search space for the GA, as the optimized strategies with boundary detection returned twice as much food as the strategies without boundary detection, as seen in Figure 6, which tracks the progress of the GA's optimization across 50 generations.

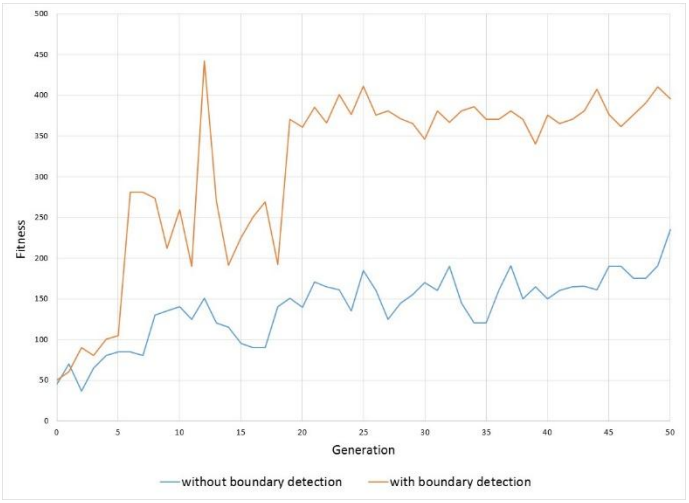

Figure 6: Performance of top candidate found by GA at each generation of optimization, for systems with and without boundary detection

To recap the case study, the original $(Fs)_{req}$ was decomposed into an Fg of "indicate food location" and Fa corresponding to flocking and state-signaling and necessary functionality such as picking up and dropping food. All of these functions were assigned corresponding behaviors in the Ca, with a behavioral logic Ba formed to choose among the behaviors in the Ca. The Ba was formed as a parametric behavioral model, where the action is the result of stimuli generated by fields within the agent's sensory radius. The parameters were the relative weights of the contributions of various stimuli in the agent's field of detection.

These relative weights were tuned by a genetic algorithm which would repeatedly simulate the Bs of a system of agents running Ba's based on different parameter sets. This optimization was used to repeatedly update the Ba until the system was able to retrieve 190 units of food within 1000 timesteps. The Fg of "indicate food location" was fulfilled by very negative value of the parameter governing angular alignment with neighbors, so that if agent's got stuck on a wall close to one another, one would have to turn and move toward the middle of the field, and thus the food.

The GA efficiently searched within the PBM given to it, but after analysis of the $(Bs)_{sim}$, the designer felt that it was necessary to change the Ca and run a new optimization process with the resulting PBM. This new PBM included boundary detection in the Ca. The Fg of indicating food location was still fulfilled by the Sg of lines of agents at the edges of the arena, but these lines now maintained a slight distance away from the edge, allowing faster flow of agents toward the food, and resulting is $(Fs)_{sim}$ of returning 400 units of food within 1000 timesteps.

CONCLUDING REMARKS AND FUTURE WORK

By analyzing example designs from the literature, we have prescribed a notation for the fundamental elements of design in terms of *function*, *behavior*, and *structure* at three levels of system architecture: *agent*, *group*, and *system*. This paper presents an initial step in our efforts to categorize the states and processes involved in the design of self-organizing systems. In future work we will attempt to improve the ontology in two ways: by expanding the scope to cover more examples of self-organizing systems and to refine the components into more detailed levels.

The ontological entity that we explored in most detail was the agent behavior, Ba. We distinguished between two elements of agent behavior, the capacity of possible behaviors, and a mechanism for selecting among these behaviors at any given time. We have described a parametric behavioral model for characterizing agent behavior based on fields of influence that the agent senses in its immediate neighborhood.

Other ontological entities can be the focus of future work as in-depth models of function and structure in self-organizing systems could yield important insights if compared to our model of Ba. The concept of chronology in SO system deployment should also be discussed in more detail. In our foraging system, the Ss formed at run-time to accomplish the Fs, but in other systems, artificial embryogeny [42] for example the Ss is the self-organized output of the design process, with structure fixed before deployment. We will explore where designers choose to allow self-organization with respect to the system life cycle, and the significance of this chronology on system performance.

The mapping processes during design, where most of the design decisions are made, should receive careful scrutiny. When and how should designers update the Ca if optimization of Ba is not sufficient? How can function decomposition be aided if designers have only a vague idea by analogy of how Fg or Fs can be accomplished? Can a standard set of self-organizing "recipes" be created that can aid or automate some of these design steps? All of these questions will guide our future research.

In our case study on the design of a foraging system, we provided an example of using a GA to perform the reformulation of Ba according to its simulated Bs. This particular optimization algorithm was chosen because of successful similar applications reported in the literature [39], [43] and its black-box approach which makes it easy to integrate with a diverse array of simulations, but there are other optimization algorithms that should be evaluated to see if they can more successfully optimize the detail designs of self-organizing systems.

A fully defined ontology for the self-organizing system design process will allow us to push forward on other research goals, such as modeling systems of heterogeneous agents, growing systems, and connected systems. As the behavior of these systems becomes advanced enough to solve real-world problems, it can be programmed onto physical robots for study of the full design process from functional requirements to physical agents.

REFERENCES

[1] M. Prokopenko, "Design Versus Self-Organization," in *Advances in Applied Self-Organizing Systems*, Springer London, 2013, p. Ch. 1.

[2] M. Hadzic, P. Wongthongtham, T. Dillon, and E. Chang, "Introduction to Ontology," in *Ontology-Based Multi-Agent Systems*, Springer Berlin Heidelberg, 2009, pp. 37–60.

[3] N. Washington and S. Lewis, "Ontologies: Scientific data sharing made easy," *Nature Education*, vol. 1, no. 3, 2008.

[4] Y. Labrou, T. Finin, and Y. Peng, "Agent communication languages: the current landscape," *IEEE Intelligent Systems and their Applications*, vol. 14, no. 2, pp. 45–52, 1999.

[5] R. Beckers, O. E. Holl, J. L. Deneubourg, Z. Bielefeld, and D.- Bielefeld, "From local actions to global tasks: Stigmergy and collective robotics," in *Artificial Life IV*, 1994, pp. 181–189.

[6] J. Werfel and R. Nagpal, "Extended stigmergy in collective construction," *Intelligent Systems, IEEE*, vol. 21, no. 2, pp. 20–28, 2006.

[7] W.-M. Shen, B. Salemi, and P. Will, "Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 700 – 712, Oct. 2002.

[8] A. Requicha and D. Arbuckle, "Issues in Self-Repairing Robotic Self-Assembly," in *Morphogenetic Engineering: Toward Programmable Complex Systems*, 2012, pp. 141–156.

[9] L. Bai and D. Bree, "Chemotaxis-Inspired Cellular Primitives for Self-Organizing Shape Formation," in *Morphogenetic Engineering: Toward Programmable Complex Systems*, 2012, pp. 141–156.

[10] R. Nagpal, "A catalog of biologically-inspired primitives for engineering self-organization," in *Engineering Self-Organising Systems*, Springer, 2004, pp. 53–62.

[11] G Zouein, C. Chen, and Y. Jin, "Create Adaptive Systems through 'DNA' Guided Cellular Formation," in *Design Creativity 2010*, 2010, p. 149.

[12] C. Chen and Y. Jin, "A Behavior Based Approach to Cellular Self-Organizing Systems Design," in *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Washington, DC, 2011.

[13] J. Humann, Y. Jin, and N. Khani, "Evolutionary Computational Synthesis of Self-Organizing Systems," *AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 2014.

[14] W. Chiang and Y. Jin, "Toward a Meta-Model of Behavioral Interaction for Designing Complex Adaptive Systems," in *ASME IDETC/CIE 2011*, 2011, pp. 1077–1088.

[15] R. Doursat, H. Sayama, and O. Michel, "Morphogenetic Engineering: Reconciling Self-Organization and Architecture," in *Morphogenetic Engineering*, R. Doursat, H. Sayama, and O. Michel, Eds. Springer Berlin Heidelberg, 2012, pp. 1–24.

[16] B. Edmonds, "Using the experimental method to produce reliable self-organised systems," in *Engineering Self-Organising Systems*, Springer, 2005, pp. 84–99.

[17] B. Fontaine, K. van Achterberg, M. A. Alonso-Zarazaga, R. Araujo, M. Asche, H. Aspöck, U. Aspöck, P. Audisio, B. Aukema, N. Bailly, M. Balsamo, R. A. Bank, C. Belfiore, W. Bogdanowicz, G. Boxshall, D. Burckhardt, P. Chylarecki, L. Deharveng, A. Dubois, H. Enghoff, R. Fochetti, C. Fontaine, O. Gargominy, M. S. G. Lopez, D. Goujet, M. S. Harvey, K.-G. Heller, P. van Helsdingen, H. Hoch, Y. De Jong, O. Karsholt, W. Los, W. Magowski, J. A. Massard, S. J. McInnes, L. F. Mendes, E. Mey, V. Michelsen, A. Minelli, J. M. N. Nafrıa, E. J. van Nieukerken, T. Pape, W. De Prins, M. Ramos, C. Ricci, C. Roselaar, E. Rota, H. Segers, T. Timm, J. van Tol, and P. Bouchet, "New Species in the Old World: Europe as a Frontier in Biodiversity Exploration, a Test Bed for 21st Century Taxonomy," *PLoS ONE*, vol. 7, no. 5, p. e36881, May 2012.

[18] M. J. Yoder, I. Miko, K. C. Seltmann, M. A. Bertone, and A. R. Deans, "A gross anatomy ontology for Hymenoptera," *PLoS One*, vol. 5, no. 12, p. e15991, 2010.

[19] The Gene Ontology Consortium, "The Gene Ontology: enhancements for 2011," *Nucleic Acids Research*, vol. 40, no. D1, pp. D559–D564, Nov. 2011.

[20] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, and A. Kasarskis, "Gene Ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–9, May 2000.

[21] J. S. Gero and M. A. Rosenman, "A conceptual framework for knowledge-based design research at Sydney University's Design Computing Unit," *Artificial Intelligence in Engineering*, vol. 5, no. 2, pp. 65–77, 1990.

[22] J. S. Gero and U. Kannengiesser, "The situated function–behaviour–structure framework," *Design Studies*, vol. 25, no. 4, pp. 373–391, Jul. 2004.

[23] J. Hirtz, R. B. Stone, D. A. McAdams, S. Szykman, and K. L. Wood, "A functional basis for engineering design: reconciling and evolving previous efforts," *Research in engineering Design*, vol. 13, no. 2, pp. 65–82, 2002.

[24] A. Chakrabarti, K. Shea, R. Stone, J. Cagan, M. Campbell, N. V. Hernandez, and K. L. Wood, "Computer-Based Design Synthesis Research: An Overview," *Journal of Computing and Information Science in Engineering*, vol. 11, no. 2, p. 021003, 2011.

[25] C. Bryant, R. Stone, D. McAdams, T. Kurtoglu, and M. Campbell, "Concept generation from the functional basis of design," in *International Conference on Engineering Design, ICED*, 2005, vol. 5, pp. 15–18.

[26] J. K. S. Nagel and R. B. Stone, "A computational approach to biologically inspired design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 26, no. 02, pp. 161–176, Apr. 2012.

[27] A. Chakrabarti, V. Srinivasan, B. S. C. Ranjan, and U. Lindemann, "A case for multiple views of function in design based on a common definition," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 27, no. 03, pp. 271–279, Jul. 2013.

[28] N. P. Suh, *The principles of design*. New York: Oxford University Press, 1990.

[29] G. Pahl, K. Wallace, and L. Blessing, *Engineering design a systematic approach*. London: Springer, 2007.

[30] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *ACM SIGGRAPH '87 Conference Proceedings*, 1987, vol. 25–34.

[31] S. Garnier, T. Murphy, M. Lutz, E. Hurme, S. Leblanc, and I. D. Couzin, "Stability and Responsiveness in a Self-Organized Living Architecture," *PLoS Comput Biol*, vol. 9, no. 3, p. e1002984, Mar. 2013.

[32] S. Pugh, *Total design*. Addison-Wesley, 1990.

[33] V. Trianni, *Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots*. Berlin: Springer, 2008.

[34] J. Humann and A. M. Madni, "Integrated agent-based modeling and optimization in complex systems analysis," presented at the Conference on Systems Engineering Research (CSER 2014), Redondo Beach, CA, 2014.

[35] K. Kelly, *Out of control : the new biology of machines, social systems and the economic world*. Reading, Mass.: Addison-Wesley, 1994.

[36] E. Bonabeau, *Swarm intelligence: from natural to artificial isystems*. New York: Oxford University Press, 1999.

[37] U. Wilensky, *NetLogo Ant Model*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University, 1998.

[38] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3293 –3298.

[39] J. Humann and Y. Jin, "Evolutionary Design of Cellular Self-Organizing Systems," presented at the ASME IDETC/CIE, Portland, OR, 2013.

[40] U. Wilensky, *NetLogo*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University, 1998.

[41] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Wesley Professional, 1989.

[42] O. Yogev, A. A. Shapiro, and E. K. Antonsson, "Computational Evolutionary Embryogeny," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 301–325, Apr. 2010.
[43] F. Stonedahl and U. Wilensky, "Finding Forms of Flocking: Evolutionary Search in ABM Parameter-Spaces," in *Proceedings of the MABS workshop at the Ninth International Conference on Autonomous Agents and Multi-Agent Systems*, 2010.