Role	
Division	
Organization	The MITRE Corporation
Address	202 Burlington Road, Bedford, MA, 01730, USA
Email	

Abstract

Rigorous approaches to the engineering of decentralized, multi-agent systems are nascent. Providing contrast to these are hierarchical systems, in which central operational entities distribute tasks and aggregate results. We explore self-organized networks of sensors and effectors to understand how systems without centralized command and control can be created to accomplish an enterprise-wide mission when the individual agents do not have a concept of command or hierarchy. The individual agents do, however, have knowledge of how to act given their own embodied situational awareness, with an implicit understanding that these behaviors contribute to collective enterprise mission goals. When such local situation awareness is shared with appropriate other agents, we say the overall system is using *distributed situational awareness (DSA)*. Our practical systems engineering challenges then are to determine what kinds, how much, and through what mechanisms such DSA should be shared, to enable the overall system to be most effective and efficient. The problem used to explore these questions is one of mitigating the damage caused by asteroid impacts within the continental United States with a set of autonomous vehicles that have limited capacities for sensing, communications, and information processing. We use this challenge problem to prototype an enterprise command and control system using an approach rooted in complexity science and computational social science methods. The experimental testbed we describe here allows us to study emergent engineering of decentralized multi-agent systems, by observing their enterprise-level mission performance as we apply stressors and instill agent-level behaviors enabled by DSA information sharing. Interestingly, we find that maximum system-level performance is obtained when 1) the vehicles are able to minimally coordinate mitigation efforts, and 2) are of limited capability with respect to sensor and communications ranges.

Engineering Decentralized Enterprises: Emergent Mission Accomplishment Without Centralized Command and Control

Michael D. Norman(✉), Paul E. Silvey, Matthew T. K. Koehler, and Kirbi C. Joe

The MITRE Corporation, 202 Burlington Road, Bedford, MA 01730, USA
mike.d.norman@gmail.com

Abstract. Rigorous approaches to the engineering of decentralized, multi-agent systems are nascent. Providing contrast to these are hierarchical systems, in which central operational entities distribute tasks and aggregate results. We explore self-organized networks of sensors and effectors to understand how systems without centralized command and control can be created to accomplish an enterprise-wide mission when the individual agents do not have a concept of command or hierarchy. The individual agents do, however, have knowledge of how to act given their own embodied situational awareness, with an implicit understanding that these behaviors contribute to collective enterprise mission goals. When such local situation awareness is shared with appropriate other agents, we say the overall system is using *distributed situational awareness (DSA)*. Our practical systems engineering challenges then are to determine what kinds, how much, and through what mechanisms such DSA should be shared, to enable the overall system to be most effective and efficient. The problem used to explore these questions is one of mitigating the damage caused by asteroid impacts within the continental United States with a set of autonomous vehicles that have limited capacities for sensing, communications, and information processing. We use this challenge problem to prototype an enterprise command and control system using an approach rooted in complexity science and computational social science methods. The experimental testbed we describe here allows us to study emergent engineering of decentralized multi-agent systems, by observing their enterprise-level mission performance as we apply stressors and instill agent-level behaviors enabled by DSA information sharing. Interestingly, we find that maximum system-level performance is obtained when 1) the vehicles are able to minimally coordinate mitigation efforts, and 2) are of limited capability with respect to sensor and communications ranges.

Z. Yang and E. von Briesen (Eds.): CSSSA 2021, SPCOM, pp. 1–28, 2022.
https://doi.org/10.1007/978-3-030-96188-6_10

1 Introduction

1.1 [Centralized] Command and Control

The concept of Command and Control (C2) emerged in the context of military decision making but is generally applicable to multi-agent collective problem-solving in dynamic, uncertain environments (often adversarial or competitive ones). Although many C2 problems are modeled or studied as games with strict constraints such as rules of engagement and bounded numbers and capabilities of agent participants, the real world is much more open-ended. To operate with a singularity of purpose despite a massive distribution of labor, hierarchical organizations emerge, centralizing their strategic decision-making to command levels where the most difficult cognitive processes are concentrated. However, if all relevant information for choosing a course of action were known at the level in which material actions occur, the centralized approach might be replaced by a fully decentralized one. The difficulty of achieving this lies in the fact that neither the necessary information per node, nor the action policies based on that information are easily determined to accomplish enterprise mission objectives under a wide range of situational eventualities.

As noted, command and control of enterprise decision-making has historically been a highly centralized activity. Commanders make decisions on desirable enterprise-scale outcomes, and those desired outcomes are then pushed down through the C2 echelons and successively decomposed into subplans based on the decisions of those at each echelon. Execution of a commander's orders typically follows a hierarchy of C2 by way of echelon. This form of regimented C2 is the current doctrine.

The concept of situational awareness (SA) is used in centralized enterprises to refer to information about the system and its environment that is available to the central command node to inform decision-makers. The SA available to a central authority node is often limited in scope, lossy and potentially stale by the time it reaches decision-makers. In centralized C2, SA information filters up, but it rarely is pushed back down, creating a tension within the tactical components who are needed to collect information about the environment but who rarely benefit from the effort.

1.2 Decentralized Command and Control

Decentralized command and control can only be realized if nodes are provided a significant level of agency and the command hierarchy is flattened. The term agent in this context refers to nodes that have been imbued with the ability to participate in adaptive decision-making, inclusive of leaf nodes at the tactical edge. The scope of their decision-making is concentrated on their local physical actions including (but not limited to) their mobility, but it also includes choices they make with respect to information processing and communication. The information used to inform their decisions is gleaned from local observations of their environment and information shared with them by other agents residing

in the communications network. No single agent is expected to have complete situational awareness of the entire system, nor should we expect any two agents to have synchronized, or even consistent, observations or perceptions of its state. Although more complex than a highly optimized, functionally segregated enterprise, enabling agency may lead to a more resilient and less fragile system as it empowers individual functional units to deal with uncertainty in the proximity of its occurrence.

Information Loss. In a centrally controlled system, data is lost when an edge node must compress its local situational awareness down to a transmittable and digestible chunk for consumption by a central authority [1]. Although this loss of data is often by design, especially in contested environments, the potential information/knowledge/wisdom that may be generated from this data is limited by the decisions of the intervening echelons between the commander and the leaf node. A node may face a manifestation of uncertainty in this process resulting in loss of data that previously held no value under conditions of certainty. An additional challenge is that neither the central authority nor the communications network can feasibly handle the raw experiential data being produced by all the nodes within an area of responsibility. The data lost may well contain vital clues as to what the optimal decision should be in the current situation. This may result in potentially bad action decisions being pushed out to the leaf nodes. The information loss inherent to a hierarchical system architecture created to support centralized decision-making may be presented as an unavoidable technological challenge to be overcome, rather than an inherent effect of system design decisions. This is often seen when an enterprise seeks to collect all data under one roof. However, even if all enterprise data sources are federated, and not aggregated, the meta-strategy of "aggregate all the data first" misses the true challenge: even if one has "all the data", one does not have the path-dependent experience of the embodied leaf nodes that collected that data (nor the time to manually review details at that scale). Information with premeditated value is generated for a disembodied authority which is subsequently challenged by uncertainty when it presents itself in the realm of the data the system was designed to propagate upwards. "All the data" is never all the data; it is the data that was preconceived to have value under conditions of certainty.

Timeliness. Timeliness becomes a challenge for centralized C2 as there is an inherent delay, especially in contested environments, between an edge node encountering an unpredicted event, generating a report of that event (which is subject to the information loss discussed in the previous section), transmitting that report, and awaiting the proceeding command. Like the information loss issue, often the timeliness issue is lensed as a technological obstacle to be overcome, rather than an unavoidable consequence of centralized system design.

In a centralized design, the likelihood of the inability to propagate information from any node to the centralized node is inherently less likely than if there were many possible decision-making nodes. This is of course a consequence of

the topology. All paths that must route to a particular node have a lower probability than if there were either many pathways to that node, or many alternative nodes. The longer the pathway the more likely an error and, likely, increasing delivery time.

Can we break out of this paradigm? Can we design an enterprise that can accomplish its mission without centralized decision-making? The simulation formulated to explore our challenge problem is abstract and coarse-grained to push forward in answering these questions while embracing the holistic complexity of the entire system of agents operating simultaneously.

1.3 Mission as an Emergent System Property

When a decentralized system is operating, its performance is emergent, as the system level dynamics are not reducible to any individual agent. If the performance is in service of achieving a commander's intended outcome, then we may view the mission itself as an emergent property of the system. This is especially true because mission objectives are usually achievable only to a degree, and the outputs of complex adaptive systems are frequently sensitive to small changes in initial conditions or configuration parameters.

Although the general understanding is that there are certain mission-scale decisions that must be made by a centralized authority, the underlying assumption for our current research is that decisions occurring within the execution of many enterprise-supporting functions can be delegated downwards. Future work will include identification and modeling of the decisions of an enterprise that cannot be decentralized. We believe that properly engineered adaptive and self-organized enterprises should be capable of operating in decentralized ways most of the time yet be able to respond to truly unexpected situations by appealing to some forms of centralized command and control. To that end, we begin by exploring architectures where all decisions are localized and distributed, deferring command authorities to future situations where centralization is strongly justified or needed.

1.4 Distributed Situational Awareness

Distributed situational awareness (DSA) is the information that any agent in the system has about the system itself, its environment, and information collected by interacting with its network peers. Any two agents do not likely retain the same perception of state due to observational variations and inherent truth distortions (e.g., two agents may have temperature sensors, but they are not exactly calibrated). We consider the set of data flowing through the system specific to the system-level goals to be "mission information" (MI)—this is typically the only information that a non-command node in a centralized enterprise would have access to. The set of data that exists in a decentralized enterprise's agents which is not in the set of MI, whether it is information an agent has about itself, its peers, or its environment (whether obtained via sensing, processing, inference, or other means), and here is referred to as DSA. The experimental

testbed we describe here allows us to study emergent engineering of decentralized multi-agent systems [8,13,27,33], by observing their enterprise-level mission performance as we apply stressors and instill agent-level behaviors enabled by DSA information sharing, discussed *infra*.

1.5 The Challenge Problem (Asteroid Impact)

To explore how a mission can be accomplished by a decentralized system, a challenge problem was created. The challenge problem is an autonomous asteroid impact response system (AAIRS). The stylized simulation is composed of a population of autonomous vehicles (AVs), each of which has an ability to communicate, and some of which have specialized capabilities for sensing nearby asteroid impacts and/or effectors which can repair the harm caused to people and critical infrastructure by nearby asteroid hits. The AVs are spatially confined to the continental US (CONUS). Asteroid impacts resulting from the Earth's passage through the tail of a comet induce perturbations to the system that affect its performance. The system must locate (sense) these impacts, dispatch repair (effector) units, and execute repair operations as quickly and with as little agent attrition as possible by interacting with one another within a shared environment and through their communication links.

2 Foundational Prior Work

2.1 Cooperative Distributed Problem Solving vs. C2

There is a rich literature in group decision making, organizational structure and learning, and distributed multi-agent problem solving, both from ethnographic studies of humans in teams to distributed artificial intelligence in multi-agent simulations [1,5,9,28]. This research shows that shared situation awareness is an unrealistic objective in practice, being better characterized as compatible situation awareness that both allows for and leverages individual differences in experience and local perspective, where cognition is seen as socially distributed. In Command and Control (C2) contexts this has resulted in evolution of requirements from seeking a Common Operational Picture (COP) to supporting a User-Defined Operational Picture (UDOP) [16,34]. Unfortunately, because there still exists a desire to have common displays for decision makers showing a God's Eye View of the problem environment, even UDOP implementations typically customize only via filtering by data type. If such filtering is done at display time, the overall system still assumes information relevant for any participant will be available at any location, which is both impractical and unachievable at scale.

In response, it may be argued that centralized C2 approaches try to concentrate knowledge within the enterprise to drive uncertainties out of the decision-making process. The irony of this approach is that the further that the strategic or tactical mission decomposition is taken, the more fragile the system becomes to uncertainty and unpredicted events that are virtually guaranteed to occur [6,30].

2.2 Previous Work on the Study of Complex Systems

There is a large body of work around the modeling and simulation of decentralized systems. Much of this work exists under the label of 'complex adaptive systems' [15]. Although much of the research which has leveraged agent-based modeling tools has been focused on capturing the decentralized interaction mechanisms found in the complex adaptive systems that have evolved in nature [25,26], we adopt the techniques pioneered by this field to inform and guide our prototypical decentralized enterprise engineering process. Whether an agent represents a bird or an unmanned aerial vehicle, the abstract and coarse-grained nature of the holistic problem space around emergent functionality is identical. For example, public blockchain functionality is as emergent as that of an ant colony [20,36].

There has been some work to combine the study of decentralized C2 with Agent-Based-Modeling, and we see our work here as continuing this tradition [4,7,14,29,37]. The trade space is enormous: do we want many, inexpensive agents, or fewer, exquisite agents [12]? Do we want homogenous agents [26] or heterogenous agents [23]? Do we want segregated sub-networks or a single global interconnected network? Do we want to imbue agents with heuristic-based local decision-making rules, or do we want to use machine learning to train them to produce opaque but performant reinforcement learning policies [21]? The only way to answer these questions is to form a hypothesis and then create a simulation to test the hypothesis. Since emergent system behavior is definitively irreducible and therefore does not lend itself to closed-form analysis, we must create an agent-based simulation to generate data which can then be analyzed to determine performance [22].

3 Simulation Approach

3.1 Why Use an Agent-Based Model?

How does one begin to approach the challenging task of creating a decentralized system? Since the functioning of a decentralized system is emergent [19], it can only occur if the entire system is executing. This stands in stark contrast to traditional systems engineering approaches which tend to develop unique functional units in isolation, then permanently compose those functional units to assemble the enterprise workflow. In traditional systems engineering, the whole is equal to the sum of the parts. To engineer decentralized enterprises that accomplish their missions by leveraging emergence, where the whole is greater than the sum of the parts, a coarse-grained and holistic starting point is needed. Agent-based modeling tools are purpose-built to address this decentralized system design challenge [11,17,22,25] because they provide the primitives needed to rapidly prototype coarse-grained systems composed of individual decision-making entities as unified, executable wholes.

In this initial step of breaking out of the traditional systems engineering paradigm to one that can explore decentralized systems, some simplifying

assumptions about the enterprise must be made. Firstly, the enterprise in this prototype has but one mission, and that mission does not change. In reality, and in further iterations of this work, it is imperative that the enterprise be capable of adapting itself on a fundamental level, i.e., not just enable agent-scale decisions from moment to moment (which are the subject of this prototypical effort and of most complex adaptive systems research), but also adapting the set of action(s) that an agent is afforded, the heuristics and/or policies that the agent follows when determining whether or not to take said action(s), and how/when they interact with their peers, etc.

Secondly, to enable each agent in our simulation to make appropriate local decisions at run-time, we will need to consider what actions they can take and what information might best inform them in choosing one action over another. Therefore, we see ourselves as system engineers as making design-time decisions about the types of information and action decisions we will give our agents. Then, through extensive simulation experimentation, we collect system level performance metrics to assess how well our design choices work for the enterprise under a wide variety of parameter settings that alter the enterprise configuration. In following this basic methodology, we have found it useful to be guided by the following principles learned from complexity science.

3.2 Four Pillars of Functional Emergence

The study of complex adaptive systems that have evolved in nature provide us with a starting point to begin to assemble and implement some of the concepts that will be necessary for emergent mission accomplishment. We believe these concepts are decentralized decision-making, diversity, functional degeneracy, and stigmergy.

Decentralized Decision-Making. The worst outcome for an agent making a bad local decision is that they may be attritted. In a decentralized system of many agents, this won't have much, if any, effect on the system's continued functioning. On the other hand, in a centralized enterprise, a bad decision by a central authority could cause systemic ruin.

Diversity. Homogeneity across a decentralized enterprise leads to systemic fragility and/or prohibitive cost. Certain agents require certain capabilities, and if every agent has every capability, we will end up with extremely expensive assets. This inflated cost would reduce the number available, and we would end up with a system that is fragile to attrition due to the relatively low number of agents. Additionally, homogenous agents are more susceptible to common-mode failures. This, too, leads to systemic fragility to intelligent perturbation.

According to Ashby's Law of Requisite Variety [2], the complexity of a given environment must be at most equal to the complexity of the interfacing system at the scale of that interface point. If we can design a system with the diversity necessary to generate the complexity required at the scale of the interface point

with the environment (i.e., a single agent), we have a system that may thrive. The potential additional benefits of diversity go far beyond the scope of this section and this paper [24].

Functional Degeneracy. A decentralized system must maintain functional redundancy despite diversity. Functional redundancy is typically achieved in enterprise architectures via standby nodes in fail-over roles. Taking our cues from nature's laboratory, we notice that one of the most complex systems in the world, the human brain, doesn't have a backup brain on standby. Instead, functional redundancy to ischemic events is achieved by the plasticity of alternative, self-organized neuronal ensembles [10]. Put simply, degeneracy is when a system's function (or a satisficed version of it) can be performed by diverse agents (or neurons, if you like). An agent or network of agents which are specialized for one task may be able to take on a task meant for other agents which are specialized for another task (although the effectiveness may be adversely impacted; it is far superior to system failure.

Stigmergy. If we are to create a decentralized system, then we must be as elegant and minimalist as possible with communications load to ensure agents with heterogenous comms and processing abilities can enable degeneracy within the system. We are trading off heavy amounts of communication and situational awareness aggregation for increased decision-making at the edge. If decisions are being made at the edge, then direct communication of SA that may be observationally apparent to local agents may be unnecessary.

Stigmergy is traditionally known in the sociobiological literature as a class of mechanisms that facilitate interaction between animals [31]. One of the major features of stigmergy is its focus on indirect communication via the perturbation of a shared environment. Another way to look at it is as a set of instinctive inferences that an animal makes about the state and dynamic activity of other animals of relative import. The coordination and subsequent emergent behavior of an ant colony is enabled by the stigmergic effects of ant pheromones [36]. We consider stigmergy to be a type of DSA. It is important to stress that while information supplied to an agent in system via stigmergy can be transformed into a message which can then be directly communicated (i.e., to a centralized decision-making agent), that information is now taking up bandwidth in the communications network, and compute/storage at each node it must then traverse.

Additionally, direct communication doesn't have environmental persistence the way stigmergic communication may. During direct communication, the system, rather than the environment, is responsible for persisting the message. In decentralized environments, this may prove to be onerous at scale if not approached from a minimalist perspective.

3.3 Optimizing to a Scenario

One of the hypothesized factors that will drive system robustness is not optimizing to a specific premeditated scenario. To avoid this, our baseline threat model,

i.e., pattern of asteroid impacts, is stochastic in nature. By building the first iteration of our system to be robust to randomized perturbation, we hypothesize we may develop a design solution that gives us more generalizable results.

4 Simulation Implementation

The simulation we have developed to explore this problem space is called the Autonomous Asteroid Impact Response System (AAIRS) and is implemented in the NetLogo programming environment [32,35]. The environment is a stylized, two-dimensional representation of the continental United States (CONUS). The overall enterprise mission objective, as noted above, is to use agents with sensing capabilities to detect asteroid impacts and agents with effecting capabilities to go to those impact sites and do repair work, and to do all this as quickly and thoroughly as possible while minimizing losses. The essential Mission Information comes from sensing agents when they generate time and location reports of asteroid impacts, as these observations drive overall system response behaviors. Other types of information observations and exchanges constitute our DSA extensions. Before we describe the decentralized approach implemented in our experiments, let's consider how this problem might be solved using traditional C2 hierarchies.

A centralized design might utilize predefined tasking to position the sensor and effector network and its communications support relays into Areas of Responsibility (AORs), assuming that there was enough time to prepare for the anticipated threat. It would then direct the network to funnel all asteroid impact reports to a centralized location, and using such global knowledge, do resource allocation and planning for effector work efforts. Finally, it would send command messages to specific effectors to direct them to their assigned tasking. A central C2 authority would also have to respond to problems introduced because of node attrition, the first being simply to detect losses after they occur, and then to use dynamic planning to appropriately re-assign or backfill for those lost nodes, be they sensors, comms relay nodes, or effectors. And in a worst-case scenario, the enterprise must have contingency plans for what to do if the centralized C2 authority itself is lost.

In the proposed decentralized approach, several aspects of the simulation and how the agents work to solve the challenge problem are driven by random processes, including giving them random starting positions and default random movement behaviors. In addition, the spatial and temporal pattern of asteroid impacts is determined by a stochastic process, by default simply a small but uniform random probability of an impact occurring on the next simulation time tick and if occurring, an equiprobable chance of hitting any CONUS map location.

In addition to these randomized elements, there are several key aspects of the decentralized solution that leverage agent memory and self-directed autonomy. The first is simply that the sensors are created with their sensing capabilities turned on, such that they automatically report the time and location of observed asteroid impacts, as Mission Information (MI) to be shared with whatever nearby

agents can hear their communication broadcast. Because of limited signal range, the agents therefore rely on the dynamic network that emerges from unplanned encounters between agents. Sensors and Effectors have additional specialized behaviors, but all agents participate in the comms network using the default autonomous communication behavior. This manifests itself as each agent immediately broadcasting an MI report (starting with generated sensor observation reports), provided they have not already relayed that particular message on an earlier time step. This constitutes a basic message flooding protocol over the emergent network, which only guarantees message propagation within components of the network that are fortuitously connected to the original sensor within the time it takes for the message to transit the number of hops in the diameter of the subgraph.

The effector agents have memory to keep track of asteroids they have learned about and decentralized decision-making heuristics encoded as algorithms to decide which one they should dispatch themselves to. When they complete repair work at a mission site, they either select another target from their list, or the revert to the default random walking behavior they started with.

Additional simulation details can be found in Appendix A: ODD+D Protocol [17]. A final point regarding validation of the simulation, at this point the simulation is being used as a thought experiment and is of a system that does not currently exist (thus, we have no real-world comparator); therefore, we have opted for simple Empirical Relevance Level 0 [3] and simple face validation. This being the case, we only insist that the simulation not behave implausibly, rather than it behaving in a particular way in relation to another referent system.

The methodology for the simulation's implementation can be found in Appendix C.

4.1 Experimental Simulation Parameters

Figure 1 shows an image of the basic simulation. There are three types of agents: Sensors, which sense asteroid impacts; Effectors, that mitigate asteroid damage; and Communication nodes, which relay messages. During the simulation asteroids impact the continental US. The agents are tasks with 1) finding the impacts, 2) communicating the locations to each other, and 3) mitigating the damage caused by the asteroid impacts. Additional details can be found in Appendix A.

4.2 Experiment Execution

Simulations were executed using NetLogo's BehaviorSpace module, which facilitates the initialization of agents within a specified environment, allowing NetLogo to explore emergent behaviors of these kinds of multi-agent systems, and to log their results for subsequent analysis. NetLogo was also used to compute the necessary measures needed to evaluate model performance. The simulation parameters we used with their combinatoric value-swept settings are given in Appendix B.

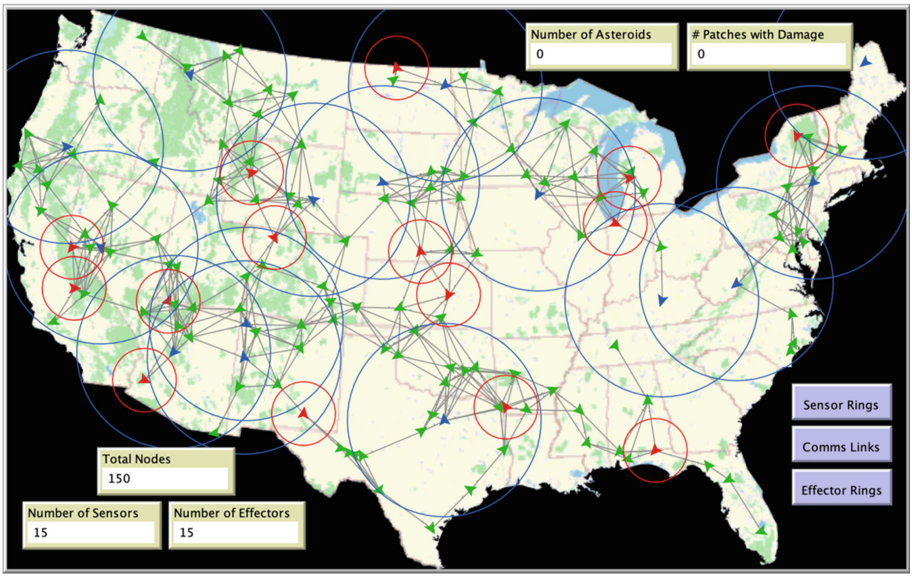

Fig. 1. Example simulation initial conditions

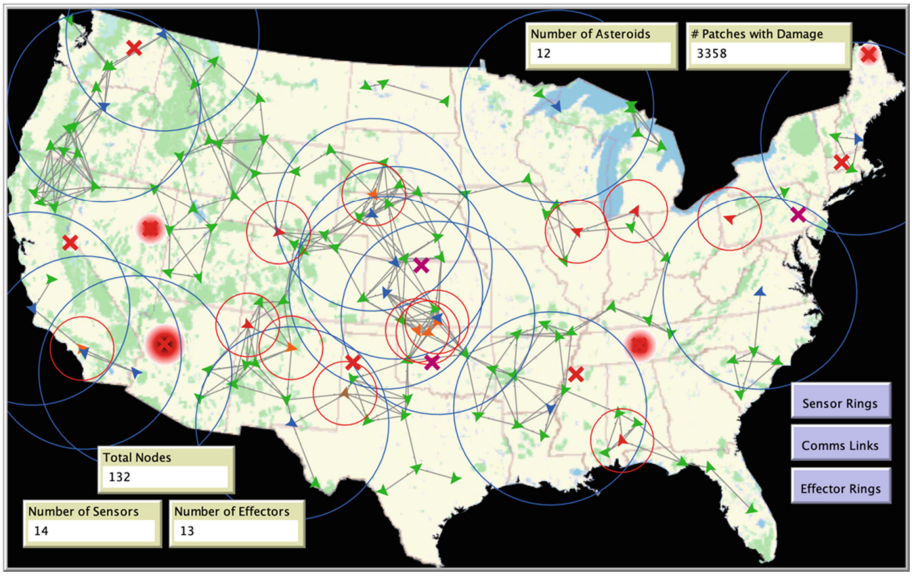

Fig. 2. Example simulation in progress

5 Design of Experiments and Results

5.1 Parametric Changes

Two phases of experimentation were conducted with the simulated system. The first experiment focused on evaluating the performance of the DSA-enabling parameters. The second experiment focused on evaluating the performance of systems composed of fewer exquisite assets against systems composed of numerous cheaper assets via nine different DSA-enabled system specifications.

5.2 Experiment 1: Effects of Toggling DSA Parameters

The goal of Experiment 1 is to establish the validity of the hypothesized DSA parameters by comparing the system's performance when they are and are not enabled.

The DSA parameters being toggled in this phase of experimentation (i.e., "DSA All") are ***selfPreservation***, ***coordinateRepairs***, and ***effectorSensing***. All other parameters are fixed and can be found in Appendix B.

Ten replicates of each configuration were simulated with an identical set of 10 random seeds used to initialize all stochastic processes, including agent starting locations, walking, and asteroid impact times and locations. The generated data is averaged across all 10 runs during postprocessing. Note that the Figure of Merit (outcome variables) are all measured per time step during the simulations, so totals like number of asteroids or number of messages accumulate over the course of each run. As a result, both the rate of change and the final values attained provide meaningful performance metrics.

The results of the first phase of experimentation are shown in Fig. 3. Looking at the Avg. Pct. Asteroids Repaired (Fully Repaired Asteroids divided by Total Asteroids) graph, the performance improvement with the DSA-enabling parameters set to ON is quite salient. At 3000 ticks of the simulation, the average performance over 10 replicates of the DSA-enabled system shows that 90.4% of the damage inflicted by the asteroid impacts has been repaired, while the average performance over 10 replicates of the same set of random seeds of the system operating without those parameters enabled has repaired approximately 65.4% of the damage.

The Avg. Pct. Useful Work graph shows that with All DSA ON, the percentage of Useful Work, defined as:

$$\frac{Cumulative\ Necessary\ Repairs}{Cumulative\ Necessary\ Repairs + Cumulative\ Redundant\ Repairs}$$

produces an average of 85.6% at $t = 3000$. With All DSA OFF, the amount of useful repair work was approximately 35.8%.

Avg. Pct. Remaining Agents (Surviving Agents divided by Total Agents) shows the effect of toggling All DSA on average agent attrition. 79.7% of the agents survive with All DSA ON, and 67.4% survive with all DSA OFF.

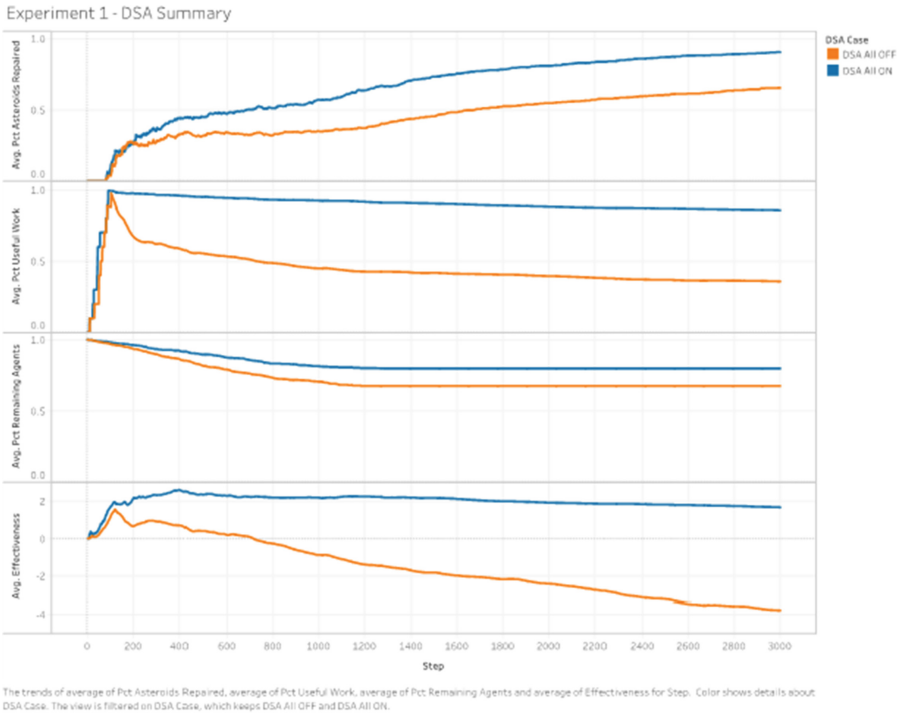

Fig. 3. Experiment 1 distributed situational awareness summary

Effectiveness in this context is defined as:

$$\frac{Cumulative\ Necessary\ Repairs - Cumulative\ Redundant\ Repairs}{Total\ Message\ Count}$$

The Avg. Effectiveness at $t = 3000$ with All DSA ON is 1.7. At $t = 3000$, with All DSA OFF, it is computed to be -3.8.

5.3 Experiment 2: Exquisite Few vs. Crude Many

Given the increase in performance measures seen with the introduction of the DSA parameters in Experiment 1, we fixed those to ON and turned our attention to the system's agent design trade space. For this phase of experimentation, we were interested in comparing performance across systems composed of few exquisite assets vs systems of many crude assets. The parameter combinations used for this experiment are shown in Table 1. Note that Design Point (DP) 5 is the baseline configuration used in Experiment 1.

Table 1. Experiment 2 exquisite few vs. crude many parameter combinations

Design point	Population size	Sensor range	Comms range	Effector range
1	30	162	108	54
2	60	144	96	48
3	90	126	84	42
4	120	108	72	36
5	150	90	60	30
6	210	72	48	24
7	240	54	36	18
8	270	36	24	12
9	300	18	12	6

Just as with Experiment 1, ten replicates of each configuration were simulated with an identical set of 10 random seeds used to initialize and realize all stochastic process outputs, including agent starting locations, walking decisions, and asteroid impact times and locations. The generated data is averaged across all 10 runs during postprocessing. DP 1 has the fewest agents with the largest range of capabilities per agent. DP 9 has the most agents and the lowest range of capabilities per agent. We were interested in which DP produced the most performant system in order to guide our design choices moving forward. It should be noted here that there is ongoing and future work which includes introducing different threat models into the simulation to ensure we are not inadvertently creating a system which is highly optimized to a single type of perturbation.

Figure 4 shows the results of Experiment 2. Looking at Avg. Pct. Asteroids Repaired, we see that DP 6 achieved the highest score at 91.2%, and DP 1 the lowest at 49.6%.

The Avg. Pct. Useful Work metric shows a tight cluster of final values across DPs. The lowest being DP 5 at 90.2%, and the highest being DP1 at 98.5%, with DP9 close behind at 98.5%. Avg. Pct. Remaining Agents graph shows another tight cluster of results. Regardless, DP1 came out on top with 72.3%, followed by DP8 at 70.6%. DP 5 came in last at 66.2% agents surviving.

The Effectiveness graph of Fig. 4 shows us a diverse set of results across system designs. DP9 stands out far above the rest with a result of 139.7. DP1 is second at 49.4 and DP8 third with 48.4. Interestingly, the rest of the DPs are clustered between DP4 at 2.7 and DP7 at 16.0.

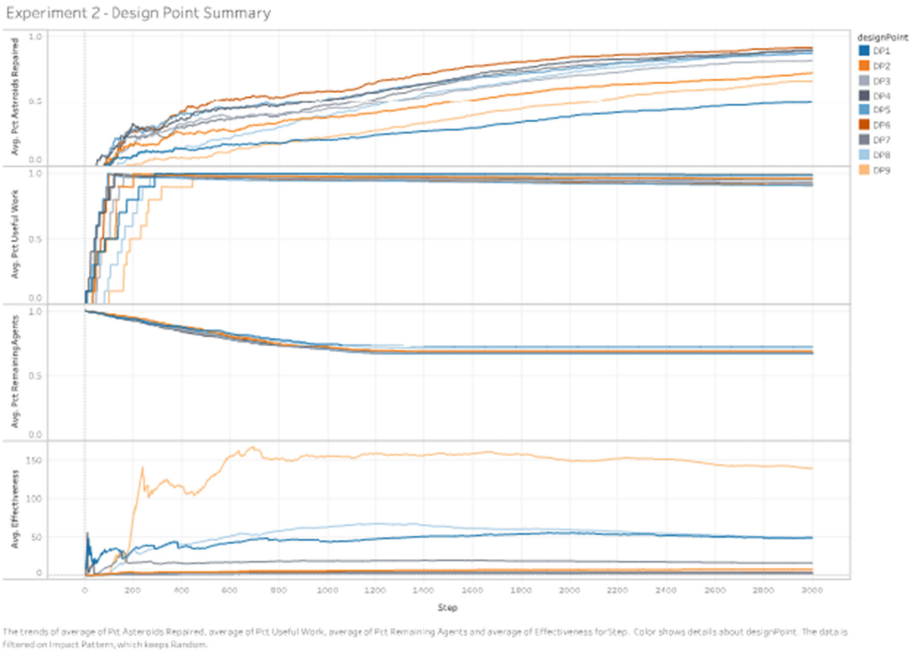

Fig. 4. Experiment 2 exquisite few vs. crude many summary

6 Discussion

We posit that functional emergence depends on DSA as well as decentralized decision-making. If situational awareness is not made available to an agent in the system, then the decisions being made by one agent will not be influenced by the previous decisions made by another agent. Although this seems to be obvious, there is sometimes a tendency to understate the importance of the embodiment and coordination-over-distance that more sophisticated, engineered complex adaptive systems may require.

It's important to keep in mind when evaluating a system that is functionally emergent that failures at the individual agent level do not necessarily indicate bad performance, as the dynamic whole is greater than the sum of the static parts. For example, expecting zero attrition is unreasonable in an uncertain and dangerous environment. Avoiding the potential for targeted attrition to lead to systemic ruin requires the development of systems that do not contain single points of failure and is one of the major motivators of taking on the challenge of mastering functional emergence despite the relative ease of creating centralized and optimized point solutions.

6.1 Experiment 1

The difference in the percentage of asteroid damage repaired when all DSA parameters were enabled is 25.0%. This indicates that imbuing our system with the foundational capabilities of DSA and being guided by the four pillars of functional emergence drove a strong improvement in the performance of the system.

The difference in the ratio of useful work to total work with the stigmergic parameter ***coordinateRepairs*** enabled is significant. Only 14.4% of all the repair work executed by the effector agents was unnecessary in the All DSA ON case. With a difference of 49.8% between All DSA ON and OFF, enabling indirect communication between leaf agents via embodied, real-time environmental sensing and decision-making has an incredible impact on system efficiency and performance.

In terms of survivability, we may conclude that 12.3% of agent attrition is caused by a lack of distributed situational awareness an agent has about its local environment and lack of decentralized decision-making capability that provides the optionality to temporarily override a self-destructive command.

The DSA-enabling parameters have a notable effect on the resulting Effectiveness metric. With a final difference of 5.4, and the All DSA OFF case value at −3.8, clearly the amount of redundant repairs was more than the number of necessary repairs, hence the value going negative, and then the magnitude of the negative value with All DSA OFF shows that the system's message passing was quite efficient compared to the amount of work performed (with the caveat that the large majority of the work was not useful to achieving the system-scale goal).

On the whole, performance across all three categories of measurement was improved with the activation of the three DSA parameters. To give us a way to summarize overall performance concisely, we averaged the values of Pct Asteroids Repaired, Pct Useful Work, and Pct Remaining Agents into a single overall metric. On this combined metric, ALL DSA ON ended up with a final score of 85.3% whereas ALL DSA OFF scored only 56.8%, yielding a 28.5% performance improvement using DSA.

6.2 Experiment 2

Using this same overall performance measure averaging the three individual percentage metrics, we found that DP 6 in Experiment 2 was the best performing design early and during most of the simulation run periods, with a final score of 82.9%. However, DP 8 ended up in the final best position, scoring slightly higher at 84.9%, for reasons discussed below. The most salient takeaway from this overall metric is that DP 1 is a terrible design, with the lowest of the final scores at 73.5%, and only barely able to repair half the asteroid-induced damage. Large, exquisite assets which are relatively few in number produce, in this case, a system which is the least capable of handling uncertainty. More work is needed to formulate conclusive statements, but the lesson taking shape here

is that the enterprises designed with large, monolithic functional components are not the most performant in a stochastic and uncertain adversarial environment. Additionally, these DP 1-type systems leave little possibility for functional degeneracy.

The Avg. Pct. Useful Work metric reflects a tight cluster across DPs, which is most likely due to the effects of ***coordinateRepairs***, which was ON for the entire experiment, along with the other DSA parameters (see Appendix B for complete list of parameter settings). Interestingly, DP 1 had the highest score, barely ahead of DP 9. The difference between them amounts to 0.1%. Given the extreme ends of the spectrum these two DPs represent, the value of this metric in determining efficacy within this trade space is questionable. This measure is much more appropriate for determining the value of the DSA parameters as demonstrated in Experiment 1, but was nonetheless included here for consistency across the phases of experimentation.

The Avg. Pct. Remaining Agents results were not informative in determining robustness of the systems. In fact, it is quite surprising to see that relative survivability was not impacted by these highly differentiated design points. Further research is needed here, but some initial observations are that systems with larger populations present more targets for potential attrition and the simplifying assumptions that a more exquisite asset present the same targetable area and same susceptibility to damage as a crude one may need to be challenged.

Naïvely, the Effectiveness measure shows that DP8 and DP 9 were the most effective designs in terms of operational efficiency, with a huge amount of the work getting done with relatively little overhead communication costs. This seems counterintuitive considering that this DP has the largest number of individual agents, thus it would be safe to assume that the largest population would generate the most messages. Upon visual inspection of the simulation at runtime, it became clear to the research team that the reason for the apparent efficiency was that the comms range of the crudest population was so small, that very few links were getting established. It turned out that the functional degeneracy which enables the Effector agents to sense asteroid damage (parameter ***effectorSensing***) appeared to be wholly responsible for the system's functionality, despite the extremely limited range (an Effector's sensing range is equal to their effector range when ***effectorSensing*** is enabled). This means that the population of Effectors was large enough that they managed to stumble directly upon asteroid impacts during their random walking, which explains the relatively poor performance compared to most of the other DPs (65.6% asteroids repaired). It is intriguing that such a segmented and crude system functioned at all.

7 Next Steps

This research effort is just scratching the surface of decentralized system design. It is believed that the work presented here will feed further work defining decentralized system design principles. The results of this work have inspired a number of interesting next steps for this research.

7.1 Threat Modeling

The implicit belief that motivates the design of the system against a random scenario is that a system that can respond effectively under complete uncertainty will also be resilient in the face of less entropic threats. Currently, we are proceeding with implementing a number of threat models which go beyond randomly placed asteroid impacts to include a number of lower entropy patterns. Adding pseudo-intelligence to the threat model will allow us to evaluate if our system is overly-optimized for our random scenario. In essence, we want to try to avoid the creation of a "highly optimized tolerant", or HOT, system as described by [6]. Rather we would prefer to create a system more along the lines of a "COLD," constrained optimization with limited deviations [18], system. COLD systems have risk aversion in their design that dramatically reduce the likelihood of catastrophic failure. In order to move in that direction, we will define a number of different threat models to ensure we do not design for one type of threat at the expense of others.

Additionally, the size of impacts is currently homogenous via a fixed parameter. Implementing a ranged parameter for varying impact size within a given run will provide a more realistic threat across newly-developed impact patterns.

7.2 Deeper Analysis

Applying additional data analysis will allow us to understand how small changes to parameters may have nonlinear effects on outcomes.

We have inspected our results by averaging across replicates to characterize a given configuration. The mean may be misleading if the generated data has a non-gaussian distribution. Future efforts will apply more rigorous data analytics to ensure appropriate conclusions are being reached.

In the spirit of Experiment 1, further work is needed to tease apart which DSA parameter combinations produce maximum and minimum performance for which metrics. Certain combinations are suspected to be the most efficient yet least performant.

Extending the results of Experiment 2, a deeper dive into the capabilities endowed by varying the system Design Points is needed.

The long term vision we are working towards would be the identification of design principles that enable the creation of an enterprise whose tactical decisions are decentralized and whose functions are emergent, but is also capable of adapting itself on a fundamental level, i.e., not just enable agent-scale decisions from moment to moment (which are the subject of this prototypical effort and of most complex adaptive systems research), but also adapting the set of action(s) that an agent is afforded, the heuristics and/or policies that the agent follows when AQ3 determining whether or not to take said action(s), and how/when they interact with their peers, etc. This may involve the introduction of some semblance of a hierarchical structure within which the decentralized components would be embedded or perhaps a decentralized governance mechanism for proposing rule AQ4 changes and reaching consensus.

Acknowledgements. The authors would like to thank Alan Stone, Eric Bellows, Elizabeth Jones, Samuel Gomez, and Sarah Mulutzie for their inspiration and tireless support of this effort. Approved for Public Release; Distribution Unlimited. Public Release Case Number 21-2699.

NOTICE: Portions of this technical data were produced for the U. S. Government under Contract No. FA8702-19-C-0001 and W56KGU-18-D-0004, and is subject to the Rights in Technical Data-Noncommercial Items Clause DFARS 252.227-7013 (FEB 2014).

A ODD+D

A.1 Overview

Purpose. The purpose of this simulation is to explore design principles for an asteroid monitoring and response system, as a motivational problem for an autonomous system designed to achieve a goal against a largely unknown threat.

Entities, State Variables, and Scales. The model is made up of a set of simulated autonomous aerial vehicles, they are broken down into three types: sensors (these entities can sense specific events in the environment), communications nodes (these entities relay messages they hear), and effectors (these entities can impact their environment); the environment is made up of the Continental United States, time is relatively abstract in the simulation but a time step is roughly one hour of wall clock time and the simulation takes place over roughly 4 months.

Process Overview and Scheduling. The simulation progresses via time steps in the following general way: 1) if it is time for another asteroid impact, the effected patches' state are updated, 2) agents with the ability to sense the environment for asteroid impacts, 3) agents then process all messages received during the previous time step, 4) agent next push messages out to each other, 5) agents then move, 6) all agents engaged in damage mitigation next continue that work, 7) as agents have now moved, we next calculate the communications network, 8) finally, patches damaged by an asteroid "wane" (the thought here is that blast and fire damage will eventually dissipate on its own). Within each group, agent activation is randomized.

A.2 Design Concepts

Theoretical and Empirical Background. The theoretical and empirical background consists of the basic idea of emergence, in that the individual behaviors of a group of agents may produce a macroscopic pattern that is difficult to infer from a study of the components in isolation, or beyond the capabilities of the individual component. See the main body text for a complete discussion of the background on distributed situational awareness and on command and control.

Individual Decision-Making. Agents make a number of decisions in this simulation: a) they randomly adjust their heading when not engaged in other activities, b) they decide whether or not to respond to a message stating the location of an asteroid impact, c) when reacting to an asteroid impact they decide on a new speed, d) they decide whether or not to relay information to other agents, e) they decide if it is too dangerous to respond to an asteroid impact.

Learning. The simulated agents do not learn.

Individual Sensing. Agents in this model sense a number of features within their environment: a) their location (as they will not leave the airspace of the Continental US), b) their proximity to others so they will not crash into each other and can communicate with each other (communication ranges are finite), and c) damage caused by asteroids.

Individual Prediction. The simulated agents do not predict.

Interaction. The agents interact with each other via messages.

Collectives. There are a number of collectives within the simulation: Agents are of one of three types (effectors, sensors, and communications nodes), agents interact via communication networks which are dynamic and recalculated each time step.

Heterogeneity. Heterogeneity is part of the simulation in a number of ways, agent randomly adjust their heading when not responding to an asteroid impact, agents react to their environment and so they each have a unique trajectory over the course of the simulation, and agents are of one of three types.

Stochasticity. Stochasticity is used in a number of ways: we set the random seed and then build algorithmically a script for the asteroid impacts, we then reinitialize the random number generator and proceed to lay down the agents with random locations and headings.

More specifically: To vary the random behaviors described above, we include a seed parameter for the simulation's random number generator, and guarantee that if it is fixed, the same initial configuration of agents and patterns of asteroid impact will recur on repeated runs. The total number of initial agents present in the simulation is set by the ***numAgents*** parameter. The percent of those agents that are sensors and effectors (in addition to being comms agents) are set by the ***percentSensors*** and ***percentEffectors*** parameters respectively. Each agent type has parameters that define their reach radius for the behaviors they implement. The ***sensorRange*** parameter determines how far away

from an asteroid impact sensors can be while still observing it for reporting purposes. The ***commsRange*** parameter represents the maximum signal propagation distance for being able to communicate with neighboring agents, and the ***effectorRange*** parameter determines how close an effector must be to an impact site to enable it to do repair work there. When agents are randomly walking, they vary their movement by making nominal changes to their current heading and speed, choosing a random normal increment (or decrement) from their current values. They are however, limited to a max velocity, as specified by the ***maxNodeSpeed*** parameter, and they will turn 180 degrees from their current heading if they are about to leave the boundary of the CONUS map. An implied minimum velocity of zero allows for agents to occasionally stand still.

Observation. Gathered data includes: messages sent, total damage caused by asteroid impacts, damage mitigated by agent activity, redundant agent mitigation activities, number of types of agents remaining in the simulation (agent are occasionally destroyed by an asteroid).

A.3 Details - Implementation

Initialization. Initialization is done in three distinct steps: 1) the random number generator is fixed and the asteroid script is created, 2) the raster image of the continental US is loaded to create the environment for the agents, 3) finally the sensors, communication nodes, and effector agents are created and randomly placed about the continental US. Specifically: the US map with an initial configuration can be found in Fig. 4, with 150 total agents, 10% of which are sensors shown as blue triangles with blue ***sensorRange*** circles, and 10% of which are effectors shown as red triangles with red ***effectorRange*** circles. The remaining comms-only agents are shown as green triangles, and the ***commsRange*** results in light grey links between agents who are close enough in space to communicate directly. Figure 2 shows the map display after several hundred steps during a simulation run. Asteroid impacts appear as X's, with A damage filled red circles that are in various states of dissipation, leaving them eventually as just red X's, turning to a magenta color when their B damage has been fully repaired. Effector agents who are on their way to an asteroid site, or who are stopped there to do repair work are shown in orange. The display uses these changes in size and color to visualize dynamic activities.

Input Data. The simulation uses raster data to define the continental US all other simulation data is created at initialization algorithmically.

Submodels. There are four basic submodels to this simulation. First, there is a submodel used for the asteroids. It is made up of three basic components: a) an algorithmic generator for the time and location of asteroid impacts, this was done to allow us to vary the script at initialization but also make it repeatable; b)

a component that reads the script at runtime and creates the impact damage at the determined time and place; c) and finally a component that "degrades" the damage caused by the asteroid impact over time irrespective of agent activities.

More specifically: The random threat model is controlled by a few additional parameters, including some that determine how agents are affected by asteroid impacts. The ***maxAsteroids*** parameter determines how many asteroids will strike over the course of a simulation run, and they will fall at a uniform random rate determined by half the total simulation run time and the this max target. In the experimental runs described here, we set the ***maxTicks*** parameter to 3000 and ***maxAsteroids*** to 50, yielding an approximate 3.33% probability of an asteroid striking on any of the first 1500 steps in each run and then zero percent after that. The ***impactScale*** parameter is used to determine the radius of damage that an asteroid causes, which is currently fixed for all asteroids in a simulation run. There are two types of damage caused by an asteroid, one we call A damage which wanes over time via a constantly decreasing radius per time step until it dissipates completely. This type of damage can interfere with communications links that traverse the impacted area, such that there is a chance they will not be received normally. Furthermore, if any agent finds itself within an asteroid's A damage radius, it will die and be removed from the simulation. As a result, agents that happen to be near impact points will die, and those who try to enter a damaged area to quickly will also die. Two parameters affect these susceptibilities to A damage, the ***nodeTolerance*** parameter and the ***linkTolerance*** parameter. Asteroid A damage values greater than these settings will kill the agent or block the transmission link, respectively.

The other type of asteroid damage is called B damage, and this persists within the initial ***impactScale*** radius until effector agents arrive on the scene and begin to do their repairs. Each repairing effector can accomplish a linear amount of repair work per time step, clearing cells within the B damage radius. By default, the effector agents remain at an impact site until their sole efforts in repairing damage would be sufficient to clear the entire impact area. More intelligent divide and conquer collaborative repair efforts are described below as part of a DSA experimental parameter.

The second submodel is used to move messages among the agents. Messages are instantiated as agents for bookkeeping convenience. Message are created and passed around to provide information about the location of asteroids and who is responding to them. The third submodel is for agent movement, agents move in one of two ways: if they are responding to an asteroid impact they set their heading toward that asteroid impact location and move forward, if they are not responding to an asteroid impact they move randomly about the airspace above the continental US by adding a bit of noise to their current heading and speed (subject to a maximum and minimum speed). The final submodel is used by effector agents to repair damage caused by asteroids, here effector agents decrease the amount of damage associated with an asteroid impact once they have arrived on scene and do book keeping to measure the among of redundant work done on that particular asteroid impact.

B DOE Parameters

In order to accommodate the number of desired simulation runs as well as the high volume of generated data, we utilized a high performance computing cluster to perform the simulations. The cluster used is a collection of 32 compute nodes, each with 28 CPUs, making it useful for compute-intensive tasks. The large number of CPUs allows for tasks to be broken up into sub-tasks and run in parallel, thus making large compute jobs much more time efficient. We use the cluster in conjunction with NetLogo in order to run simulations using a wide variety of parameter combinations in a quick, efficient manner.

As described in Sect. 5, two experiments were run which varied different sets of parameters. Table 2 contains all of the parameters which were fixed in both Experiments 1 and 2 as well as their initial values.

Table 2. Parameters fixed across all experiments

Parameter name	Type	Value
Cluster delay	Int	25
Message protocol	Str	One time flooding
Variable dispatch speed	Bool	True
Max node speed	Float	1.5
Reverse	Bool	False
Percent effectors	Int	10
Percent sensors	Int	10
Fix seed	Bool	True
Cluster range	Int	50
Max ticks	Int	3000
Impact scale	Int	20
Link tolerance	Int	1000
Max asteroids	Int	50
Chance of asteroid	Int	5
Node tolerance	Int	5
Logging	Bool	None
Max dispatch distance	Int	100

Table 3 details the parameters varied in Experiment 1 and the set of possible values they could take on. Experiment 1 consisted of all combinations of the listed parameter values.

Table 3. Parameters for experiment 1

Parameter name	Type	Value(s)
Population size	Int	150
Sensor range	Int	90
Comms range	Int	60
Effector range	Int	30
Effector sensing	Bool	True, False
Self preservation	Bool	True, False
Coordinate repairs	Bool	True, False
Seed	Int	123, 456, 789, 987, 654, 321, 111, 222, 333, 444

The parameters contained in each of the 9 design points used in Experiment 2 are enumerated in Table 4. Experiment 2 consisted of all combinations of the 9 design points with the parameter values listed in Table 5.

Table 4. Design points used in experiment 2

Design point	Population size	Sensor range	Comms range	Effector range
1	30	162	108	54
2	60	144	96	48
3	90	126	84	42
4	120	108	72	36
5	150	90	60	30
6	210	72	48	24
7	240	54	36	18
8	270	36	24	12
9	300	18	12	6

Table 5. Parameters for experiment 2

Parameter name	Type	Value(s)
Effector sensing	Bool	True
Self preservation	Bool	True
Coordinate repairs	Bool	True
Seed	Int	123, 456, 789, 987, 654, 321, 111, 222, 333, 444

C Experimental Methodology

Our initial experiments with this ABM design demonstrated that the agents could, without any central C2, successfully detect asteroid impacts and repair them. However, the initial behaviors as described above did result in some pathological behaviors that we were able to discern by watching the simulation visualizations during manual runs. This section describes the iterative approach we used to propose ways to use various kinds of DSA, and the eventual experimentation we did to show the effects of using them or not.

The first pathology we observed was that the effectors were rushing to the scene of asteroids, often getting there before the killer type A damage had dissipated. Other agents simply doing their random walks were also being killed off in this manner, so we decided to give all the agents the ability to sense such danger in front of them, and to slow down or stall their movement if the path ahead would result in them being harmed. We parameterized this DSA information and associated behavior using a ***selfPreservation*** Boolean parameter.

The next pathology we observed was that many effectors were traveling great distances to reach an asteroid impact site and crowding around it, while later asteroids that fell near to where some of those effectors came from were going unattended for long periods of time. If the network connectivity was good (as it often was initially), the first asteroid could become known to most of the effector agents, and the unintended consequence of this was a swarming or bunching of effectors around these early asteroids. The first change was to enable effectors to preempt their targets when they were en route and knowledge of a closer target became available. While this helped some, it still made the enterprise too reactive, so we added additional behavioral logic for effectors that was controlled by a new parameter ***maxDispatchDistance***. The behavioral change from this was that when an effector was further than ***maxDispatchDistance*** from its closest known asteroid, it would continue to stay in its unassigned mission (or random walking) mode, and only dispatch to asteroids that were within ***maxDispatchDistance*** from its current position. All of the experiments we report on here had a fixed value for this parameter at a value of 100 distance units on the map.

The swarming or effector clumping behavior also enabled us to realize the benefit of letting a group of effectors collaborate on the repair work of a single asteroid (following the adage "many hands make light work"). In the real world, this would enable total repair time to be reduced significantly, so we modified the simulation to optionally allow agents to cooperate or not, via a DSA parameter called ***coordinateRepairs***, which was inspired by the Stigmergy principle since this would amount to agents being able to locally observe each other's work in the environment itself, and could be done without additional message passing overhead. When agents are coordinating, they can leave the scene of an asteroid impact as soon as all the B damage has been repaired by the collective effort, whereas they will operate as if they are alone if not cooperating, doing so-called "redundant" work and staying longer than they would otherwise need to. This redundant work effort is measured as a cost that offsets the necessary (or useful)

work. Even when cooperating, however, agents who arrive at an asteroid that has been fully repaired are charged with a single time-step's worth of repair work as being redundant work, which might be saved if they had known not to keep coming when they were no longer needed. We anticipate that further kinds of DSA messaging regarding the state of asteroid repair can be used in later experiments to further reduce this kind of redundant work.

Another behavior we noticed was an obvious missed opportunity, when effector agents were on their way to an asteroid impact area and moved right past another one close to their current path that they hadn't been told about. Using the principle of functional degeneracy, we decided to optionally allow effector agents to also act as sensors, using their ***effectorRange*** reach radius parameter as a sensing range. This was controlled with a DSA parameter called ***effectorSensing***, and the behavior associated with it was to preempt the current mission target they were heading toward (if any) as well as to generate a Mission Information asteroid report and broadcast it to the network in the same manner that Sensor agents do.

In order to determine whether and how well our distributed agent design achieve enterprise mission objectives, we defined several performance metrics which alone or in combination could be used to evaluate any given parameterized scenario. All of these Figures Of Merit (FOMs) are measured dynamically at each simulation time step, so we can plot their incremental progress over time. All of the simulation experiments we describe below were performed with 3000 simulated time ticks, for two separate experiments that were designed to test particular hypotheses by varying certain of the parameters previously described. These FOMs represent our outcome variables in the experiments we ran, and they include agent attrition by type, percent of current total asteroids completely repaired, cumulative amount of repair work done (more on this later), and total messages transmitted by original sender agent type. We used these raw values to compute some aggregate measures as well, to combine the positive benefits with the negative costs. For example, we report on a metric called Effectiveness, which is computed as the necessary work minus the redundant work, all divided by the number of messages transmitted.

References

1. Alterman, R., Feinman, A., Introne, J., Landsman, S.: Coordinating representations in computer-mediated joint activities. In: Proceedings of the 23rd Annual Conference of the Cognitive Science Society, pp. 43–48. Lawrence Erlbaum Associates (2001)
2. Ashby, W.R.: Requisite variety and its implications for the control of complex systems. Cybernetica **1**(2), 83–99 (1958)
3. Axtell, R.: Three distinct kinds of empirically-relevant agent-based models. Brookings Institution Center on Social and Economic Dynamics Working Paper (2005)
4. Bonabeau, E., Hunt, C.W., Gaudiano, P.: Agent-based modeling for testing and designing novel decentralized command and control system paradigms. Technical report, Icosystem Corp, Cambridge, MA (2003)

5. Carley, K.M.C.: Coordinating for success: trading information redundancy for task simplicity, p. 10 (1990)
6. Carlson, J.M., Doyle, J.: Highly optimized tolerance: robustness and design in complex systems. Phys. Rev. Lett. **84**(11), 2529 (2000)
7. Ceruti, M.G.: Mobile agents in network-centric warfare. In: Proceedings 5th International Symposium on Autonomous Decentralized Systems, pp. 243–246. IEEE (2001)
8. Doursat, R., Ulieru, M.: Emergent engineering for the management of complex situations. In: Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems, pp. 1–10. Citeseer (2008)
9. Durfee, E.H., Lesser, V.R., Corkill, D.D.: Trends in cooperative distributed problem solving. IEEE Trans. Knowl. Data Eng. (1989)
10. Fuster, J.M., Bressler, S.L.: Past makes future: role of pFC in prediction. J. Cogn. Neurosci. **27**(4), 639–654 (2015). https://doi.org/10.1162/jocn_a_00746
11. Grimm, V., et al.: The odd protocol for describing agent-based and other simulation models: a second update to improve clarity, replication, and structural realism. J. Artif. Soc. Soc. Simul. **23**(2) (2020)
12. Hunt, E.R., Hauert, S.: A checklist for safe robot swarms. Nat. Mach. Intell. **2**(8), 420–422 (2020)
13. Kicinger, R.P.: Emergent engineering design: design creativity and optimality inspired by nature. Ph.D. thesis (2004)
14. Lee, T., Ghosh, S.: Simulating asynchronous, decentralized military command and control. IEEE Comput. Sci. Eng. **3**(4), 69–79 (1996)
15. Miller, J.H., Page, S.E.: Complex Adaptive Systems: An Introduction to Computational Models of Social Life. Princeton University Press, Princeton (2009)
16. Mulgund, S., Landsman, S.: User defined operational pictures for tailored situation awareness. In: Proceedings of the 12th International Command and Control Research and Technology Symposium-Adapting C2 to the 21st Century, Newport, RI, pp. 19–21. Citeseer (2007)
17. Müller, B., et al.: Describing human decisions in agent-based models-ODD+ D, an extension of the odd protocol. Environ. Modell. Softw. **48**, 37–48 (2013)
18. Newman, M.E., Girvan, M., Farmer, J.D.: Optimal design, robustness, and risk aversion. Phys. Rev. Lett. **89**(2), 028,301 (2002)
19. Norman, M.D.: Complex systems engineering in a federal IT environment: lessons learned from traditional enterprise-scale system design and change. In: 2015 Annual IEEE Systems Conference (SysCon) Proceedings, pp. 33–36 (2015). https://doi.org/10.1109/SYSCON.2015.7116725
20. Norman, M.D., Karavas, Y.G., Reed, H.: The emergence of trust and value in public blockchain networks. In: IX International Conference on Complex Systems, p. 22 (2018)
21. Norman, M.D., Koehler, M.T.K., Kutarnia, J.F., Silvey, P.E., Tolk, A., Tracy, B.A.: Applying complexity science with machine learning, agent-based models, and game engines: towards embodied complex systems engineering. In: Morales, A.J., Gershenson, C., Braha, D., Minai, A.A., Bar-Yam, Y. (eds.) ICCS 2018. SPC, pp. 173–183. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96661-8_18
22. Norman, M.D., Koehler, M.T., Pitsko, R.: Applied complexity science: enabling emergence through heuristics and simulations. In: Emergent Behavior in Complex Systems Engineering: A Modeling and Simulation Approach, pp. 201–226. Wiley (2018)
23. Berner, C., et al.: Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680 [cs, stat] (2019)

24. Page, S.: The Diversity Bonus. Princeton University Press, Princeton (2019)
25. Railsback, S.F., Grimm, V.: Agent-Based and Individual-Based Modeling: A Practical Introduction. Princeton University Press, Princeton (2019)
26. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 25–34 (1987)
27. Ronald, E.M., Sipper, M.: Engineering, emergent engineering, and artificial life: unsurprise, unsurprising surprise, and surprising surprise. Artif. Life **7**, 523–528 (2000)
28. Salmon, P.M., Stanton, N.A., Walker, G.H., Jenkins, D.P., Rafferty, L.: Is it really better to share? Distributed situation awareness and its implications for collaborative system design. Theor. Issues Ergon. Sci. **11**(1–2), 58–83 (2010)
29. Scacchi, W., Brown, C., Nies, K.: Exploring the potential of virtual worlds for decentralized command and control. In: 17th International Command and Control Research and Technology Symposium (ICCRTS) (2012)
30. Taleb, N.N.: Antifragile: Things That Gain from Disorder, vol. 3. Random House Incorporated (2012)
31. Theraulaz, G., Bonabeau, E.: A brief history of stigmergy. Artif. Life **5**(2), 97–116 (1999)
32. Tisue, S., Wilensky, U.: NetLogo: a simple environment for modeling complexity. In: International conference on complex systems, vol. 21, pp. 16–21. Boston, MA (2004)
33. Ulieru, M., Doursat, R.: Emergent engineering: a radical paradigm shift. Int. J. Auton. Adapt. Commun. Syst. **4**(1), 39–60 (2011)
34. Van Dijk, H.: Situation awareness in crisis situations: development of a user defined operational picture. In: ISCRAM (2015)
35. Wilensky, U.: NetLogo: center for connected learning comp.-based modeling, vol. 158. Northwestern Univ, Evanston (1999)
36. Wilson, E.O.: Sociobiology: The New Synthesis. Belknap Press (1975)
37. Yang, A., Abbass, H.A., Sarker, R.: Evolving agents for network centric warfare. In: Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation, pp. 193–195 (2005)

Author Queries

Chapter 10

Query Refs.	Details Required	Author's response
AQ1	Please confirm if the corresponding author and his email address is correctly identified. Amend if necessary.	
AQ2	This is to inform you that as the Institutional email address of the corresponding author is not available in the manuscript, we are displaying the private email address in the PDF and SpringerLink. Do you agree with the inclusion of your private e-mail address in the final publication?	
AQ3	Kindly provide the volume number and page range for Ref. [9], if applicable.	
AQ4	Kindly provide the page range for Ref. [11], if applicable.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	··· under matter to remain	New matter followed by
Insert in text the matter indicated in the margin		or (x2)
Delete	through single character, rule or underline or through all characters to be deleted	or (x2)
Substitute character or substitute part of one or more word(s)	through letter or through characters	new character or new characters
Change to italics	under matter to be changed	
Change to capitals	under matter to be changed	
Change to small capitals	under matter to be changed	
Change to bold type	under matter to be changed	
Change to bold italic	under matter to be changed	
Change to lower case	Encircle matter to be changed	
Change italic to upright type	(As above)	
Change bold to non-bold type	(As above)	
Insert 'superior' character	through character or where required	or under character e.g. or
Insert 'inferior' character	(As above)	over character e.g.
Insert full stop	(As above)	
Insert comma	(As above)	
Insert single quotation marks	(As above)	or and/or or
Insert double quotation marks	(As above)	or and/or or
Insert hyphen	(As above)	
Start new paragraph		
No new paragraph		
Transpose		
Close up	linking characters	
Insert or substitute space between characters or words	through character or where required	
Reduce space between characters or words	between characters or words affected	