

Understanding the Emergence of Self-organized Insurgent Groups

(Stage Report)

Xiaohui Cui, Justin M. Beaver, Jesse L. St. Charles and Thomas E. Potok
Applied Software Engineering Research Group
Oak Ridge National Laboratory

CONTENT

1. Introduction	2
2. Understanding Insurgent as Self-Organized Organizations	2
3. Agent Based Ideology Models for Emergence	6
3.1 Emergence	6
3.2 Agent Based Model	6
3.3 The Ideology Model in ABM	7
4. Developing Ideology Swarm Social Learning Model	9
4.1 Swarm Social Group Model	9
4.2 Adaptive Environment	11
4.3 Computational Experiments	12
4.4 Results	13
4.5 Discussion	15
4.6 Ideology Swarm Model Validation	15
5. Modeling and Validating the Emergence of Self-Organized Groups	17
5.1 Technical Approach	17
5.2 Validation Approach	19
5.3 Validation Process	20
5.4 Determining Accuracy of Fit	20
5.5 Determining Predictive Validity	21
5.6 Validation Results	22
5.7 Next Step	26
6. Community Social Network Extraction and Analysis	27
6.1 OSS Community Network Structures Extraction	27
6.2 Network Structures Analysis	28
7. Next Step Research: Modeling Social Network Emergence of Human Community	30
7.1 Modeling Social Network Emergence of Human Community	30
7.2 Self-motivating teams	31
7.3 Stigmergic Model for Distributed Task Allocation	31
7.4 Setup New Project or Join Existing Project	32
8. Relating the Researches with DoD Need	33
9. References	34

1. Introduction

This is a stage technical report on the work performed under the Office of Naval Research (ONR) contract number N0001408IP20066 during the interval from April 2008 through the end of September 2008. In this research, we are conducting a proof-of-principle research on integrating a particle swarm social model (PASS) with agent based approaches for modeling the social interactions and emergent social behaviors of self-organized insurgent groups. In contrast to the large-scale, time-consuming, and complex agent-based social simulation, this project focuses on developing a PASS model that is composed of ideology agents and is capable of generating complex emergent social behaviors that are realistic enough to understand the social behaviors of insurgent groups. The emergent behavior of the system, in which patterns and trends emerge, is considered more important than the behavior of any single part of the system. The results of this model have been validated in order to increase confidence of this model in simulation training and decision-making analysis.

2. Understanding Insurgencies as Self-Organized Organizations

As Filkins reported in December 2005 [1], Iraqi and American officials in Iraq consider the most significant insurgent issue is the fact that the insurgency consists not only of a few groups, but of dozens, possibly as many as 100, groups. These groups are not, as often depicted, hierarchical organizations whose members carry out orders from the above. They are believed to act on their own and know how to organize themselves with less command, control or charisma. Marion and Uhl-Bien [2] pointed out that these insurgents were self-initiating and self-sustaining. They argued that if bin Laden was not there, someone with another name but with the same character and role as that of bin Laden would be there. This is why it is called a phenomenon, not a person.

Our inability to perceive insurgent groups as self-organized networks puts us at risk in the War on Terror [3]. We use factors that apply to our world (the hierarchical organizations) to understand and assess the results of count insurgent (COIN) actions in Iraq. Considering these factors, we think we are winning. We assume that al-Qaeda is weaker now because its charismatic leader is on the run, hiding in caves. We assume that, if we prevent insurgents from using advanced technology for communication, they will be unable to receive their orders, and therefore, will not launch attacks. We assume that if we remove the top insurgent leaders, or if we decapitate their organization, young terrorists will slink away from the anarchic, leaderless group. However, the real truth is that we are fighting this war in the blind [4].

Unlike most ongoing COIN studies that are typically concerned with enhancing existing military capabilities for countering insurgents, our research focuses on developing a new self-organized human organization model to understand the formation and coordination of insurgent groups, to predict insurgent group behavior, and eventually to develop alternate strategies to defeat them. To prove the feasibility and credibility of this model for simulation training or decision-making analysis, we have developed a method for validating the model developed in this project.

Our research in this project is mainly inspired from the Swarm Intelligence research, which focuses on discovering the essential concepts or self-organized local rules that contribute to the global emergence of social behaviors in insects or animals. In recent years, various social studies, particularly in the political science, have statistically proved the self-organized phenomena of insurgent warfare in Iraq, Columbia, and Afghanistan [5]. Combining their research results with the methods of exploring the essential concepts of Swarm Intelligence in nature, we built a highly ideological agent-based model to represent self-organized insurgent organizations. In Swarm Intelligence research, discovery of local simple rules for the emergence of global behaviors is not used to describe all the details of social insects in nature, but used to represent essential concept of emergence. The significance of this insurgent human organization

ideology model is not to bring up a model that is absolutely right in all its details. This model is not going to claim to be completely right because humans obviously have a far more complex set of attitudes, motives, cultures, behaviors, etc. The importance of this kind of model will offer a powerful and counter-intuitive insight even though it is highly idealized [6].

By using this ideological model, we can understand the complex dynamics of human self-organized insurgent organizations or other human organizations by abstractly representing such an organization as a network of interacting agents. This representation permits of comparison and combined analysis of seemingly divergent domains due to the common patterns of organization displayed in their networks of interacting agents. Such abstract representations across domains are not uncommon: they actually reflect the existence of common principles of organization [7] similar to ones seen in nature [8]. Through investigating both software developers and social insects as agents interaction in a complex network, Valverde and colleagues [9] have proved the existence of common statistical patterns of organization in a wasp colony and in the open source software (OSS) developer communities. In their research, the agents involved—whether they are social insects or humans—have limited knowledge of the global pattern they are developing. Apparently, insects and humans differ significantly in what the individual agent can be aware of the overall designing goals. The analogy will be stronger if the individuals are of equivalent complexity in their ability to make decisions, e.g. a comparative study of human social organizations across the domains of insurgent groups and OSS developer communities. This concept helps us establish the connection between the OSS community and insurgent groups.

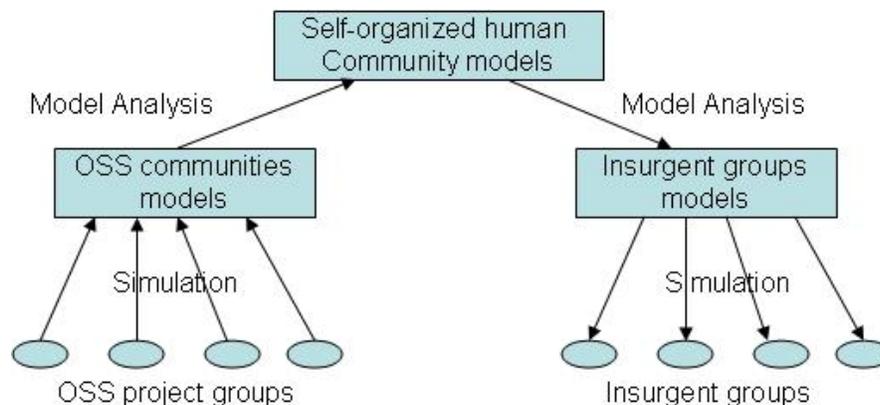


Figure 1: Self-organized social organization model takes place in both the insurgent groups and OSS developer communities

Model extracting and validating depend on the availability of large amounts of real world data. Collecting extremism/insurgent/terrorist data is extremely difficult because of the secretive nature of these organizations. Members of insurgent/terrorist groups are not easily accessible to researchers as are members of other types of social groups. As a result, the small amount of available data itself is suspected and much of the recording on terrorist groups and on terrorism itself is secondary, based on others' data, such as interviews conducted with prisoners. Beyond their special features, the insurgent groups display some overall patterns of organization not far from the ones seen in other types of self-organized human organizations [1, 3, 4]. In this research, to avoid the limited data for actual insurgent groups and leverage the high availability of historical data about some online self-organized social communities, we use the online OSS communities as a metaphor for insurgent groups in our ideological agent based model research. The OSS community on the SourceForge website comprised of more than 200,000 OSS developing teams and nearly a half million registered developers. As shown in Figure 1, we conducted research on ideology self-organized human organization models taking place in both

the insurgent groups and the OSS developer communities. Both systems define interacting agent networks with similar features that reflect limited information shared among agents. By viewing both organizations as a network of interacting agents involving goals and constraints, we can compare insurgent groups to OSS development communities and jointly build up evidence for basic principles of self-organization. Understanding these principles offers a first step toward quantitative reference models to explain emergent social behaviors in insurgent groups. As shown in Figure 2, by illuminating the extent to which self-organization local behavior is responsible for OSS communities' global patterns, such as hierarchical structure, we will be able to gain insight into the origins of organization in insurgent groups. In trying to understand the global behavior displayed by programmers inside the OSS communities, it can be useful to know if some patterns might be a consequence of simple rules shared by both types of systems. If common patterns are found to exist, we will have evidence for basic principles of self-organization that apply to both communities.

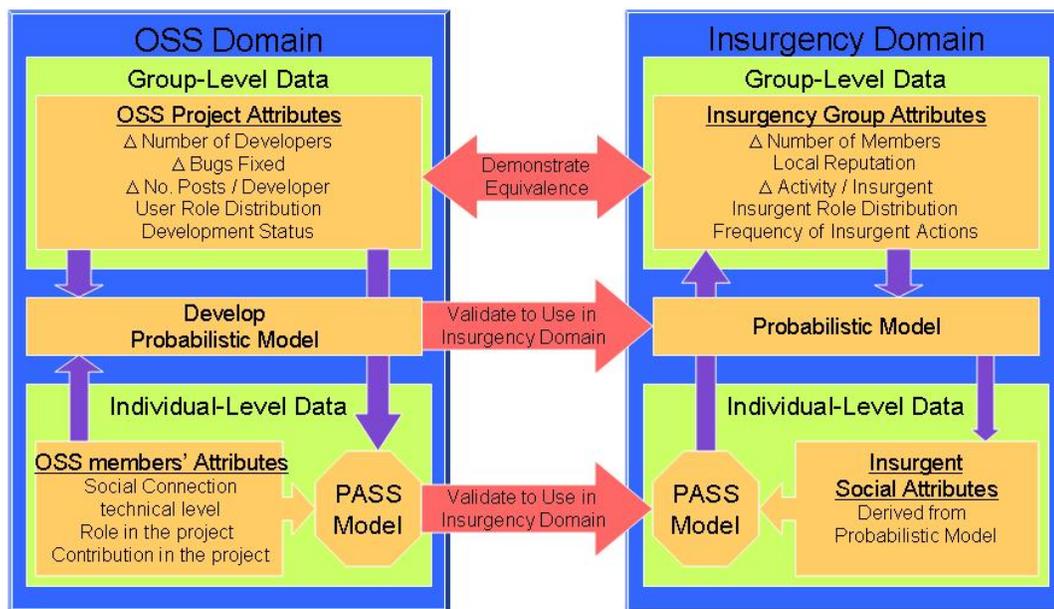


Figure 2: Technical Approach

As a proof-of-principle research, this project developed a predictive simulation framework for understanding the emergent behaviors in the complex social systems by utilizing agent-based modeling. We have generated a computational research methodology that provides a basis for studying self-organized systems through computational experiments. Our research effort has focused on the following three areas:

1. Conducting a proof-of-principle study on building an ideology swarm social learning model for collective problem solvers in an adaptive environment

We studied the integrating swarm algorithm, social knowledge adaptation, and multi-agent approaches for modeling the social learning of self-organized groups and their collective problem solving behaviors in an adaptive environment. The objective of this research is to apply the particle swarm metaphor as a model of problem solvers' group social learning for a dynamic environment. The research provides a platform for understanding and insights into knowledge discovery in social groups. Results from the simulation have shown that effective communication is not a necessary requirement for insurgent groups to attain victory in an adaptive environment.

2. Applying the simulation framework on the development community, simulating and validating the formation of the OSS developer team

We have been leveraging the OSS development domain to extract concrete measurement of individual and group behavior in order to create a data-based model of the behavior of OSS groups. In addition, the existence of these data sets allow for the validation of the developed models in terms of their ability to predict group behaviors such as membership gains/losses, group efficiency, and the occurrence of group-level actions. Our goal is to evolve our OSS model into a generic model of group-level behavior that may be applied to predict the behaviors of groups such as terrorist cells and insurgencies. To experimentally test and refine our model beyond the parameter values that were fixed in the historical case, we developed an agent-based simulation to test and validate the representative of our model. With the simulation, we experimentally manipulated the parameter values that were fixed in the archival data to investigate whether our model generalizes to a range of real world cases. Proof-of-concept research to develop an emergent behavior predictive toolkit was also conducted.

3. Extracting OSS community social network and developing agent-based ideology models for exploring network emergence in self-organized social systems

We examined how the self-organizing behavior of individual actors affects the emergence of different types of system-level networks. We used detailed historical data on a self-organized human social network to illustrate and test the plausibility of the model's theoretical mechanisms. This social network was derived from the rich real-world data available in the OSS community on the SourceForge website, comprised of more than 200,000 OSS project developing teams and nearly a half million of registered developers. We used the data to examine how this OSS community evolved from year 2003 to 2008. From these historical data, we reconstructed how the local behavior of forming OSS project teams lead to the emergence of the OSS community network. We will use networks of developers who collaborate on OSS project teams as our frame of reference and investigate how the choices developers make in picking and contributing their projects determine the global network topology of the entire field.

3. Agent Based Ideology Models for Emergence

3.1 Emergence

There is a lot of confusion in literature about the definition of emergence [10-12]. We will use the following definition as our working definition:

“Emergence is a dynamic nonlinear process that leads to emergents (properties, behavior, structure, patterns ...) at the macro level of a system from the interaction of the parts at the micro level. Such emergent can not be readily understood by taking the system apart and looking at the parts. They can however be studied by looking at each of the parts in the context of the system as a whole”. [13]

On the basis of the interaction between these elements, properties are likely to emerge that cannot be attributed to the individual elements [12]. The notion of ‘emergent properties’ can be used to describe complex social processes resulting from individual behavior [14]. Holland [10] provides an outline of the characteristics of emergence: 1) Emergence occurs in systems composed of components that obey simple laws. 2) The interactions between the parts are nonlinear so the overall behavior cannot be predicted by summing the behaviors of the isolated components. Gilbert and Conte [15] put the emphasis on emergence as a key concept of multi-agent simulations in the social sciences approach.

With the fast increase of the interaction between humans, the self-organized social structures have been very important to human development. Self-organized human social system exhibits much more emergent social behaviors than any other complex system. The social phenomena of the self-organized human organization are shaped by the individual’s behavioral choices and social interactions. These emergent behaviors or emergent social behaviors (ESB) cannot be predicted by analyzing properties of each individual component in this system. The linear, mechanistic view of the world based on a reductionist paradigm that breaks problems into component parts is no longer suitable for understanding and predicting the emergent social phenomena of self-organized human social system. It is necessary to build a new systematic research framework for understanding the emergent behaviors.

Model building is the most important tool in the study of emergence. The critical steps in constructing a model are the selection of salient features and the laws governing the model behavior. Emergence cannot be understood without models. Modeling the behavior of groups of individuals remains a challenging problem due to the complexity of human beings and their interactions. Interacting groups of people spontaneously organize themselves into groups to create emergent organizations that no individual may intend, comprehend, or even perceive. Rapid advances in computing technique has created a revolutionary modeling tool, the agent-based model (ABM), for scientific modeling and understanding of the complex social systems.

3.2 Agent Based Model

There is a growing realization across the social sciences that one of the best ways to build useful theories of group phenomena is to create working computational models of social units (e.g., individuals, households, firms, or nations) and their interactions, and observe the global structures that these interactions produce. ABM and its computer simulation of human behavioral and social phenomena is a successful and rapidly growing interdisciplinary area. The ABM is a new approach that aims to model the complex social macro dynamic behaviors emerging from the interactions of autonomous and interdependent individual actors. ABM builds social structures from the ‘bottom-up’, by simulating individuals with virtual agents, and creating emergent organizations from the operation of rules that govern interactions among agents.

The ABM was originally developed in computer science and artificial intelligence as a technology to solve complex information processing problems on the basis of autonomous

software units. Each of these units can perform its own computations and have its own local knowledge, but the units exchange information with each other and react to input from other agents. The approach was soon applied to problems involving the complex social dynamics that are of key interest to sociologists: notably emergent social norms, social structure, and social change. ABM provide true bridging explanations that link two distinct levels of analysis: the properties of individual agents (e.g. their attributes and interactions), and the emergent group-level behavior. ABM can be used for examining how very simple rules of local interaction can generate highly complex emergent behaviors that would be extremely difficult (if not impossible) to model by using traditional methods. ABMs can be used as virtual laboratories, to reveal the micro mechanisms responsible for highly complex social phenomena.

3.3 The Ideology Model in ABM

Effectively unlimited computational power removes physical constraints on the elaboration of the models, allowing researchers to write “realistic” models of enormous complexity. Some models even include intricacies such as when agents fall asleep and what is needed to wake them up [16]. In this case, it is far more difficult to analyze results in order to understand the mechanisms that generated them and their relevance when questions are theoretical. In our simulation implementation, the agent in ABM is autonomous. They independently pursue those individual goals based on their own local information. Agents in our ABM follow extremely simple rules. We assume that the overall system’s complexity emerges from the interaction of many very simple components, rather than from great complexity in the behavior of each individual agent.

Adding complexities might be reasonable in a model whose goal is a close match to a specific set of empirical data. However, closer fit to data comes at a cost.; additional processes obscure the fundamental elements of the generative theory, while adding nothing that is conceptually critical. Adding more theoretical components to the model, however well each one could be empirically justified, would only have interfered with that goal. “The more realistic and detailed one’s model, the more the model resembles the modeled organization, including resemblance in the directions of incomprehensibility and indescribability” [17]. For example, Schelling’s model of spatial segregation [18] is a pioneering example of an emerging phenomenon resulting from simple social interaction. In the U. S., the level of residential segregation has remained high, despite the fact that the income inequality between blacks and whites is decreasing. Schelling’s aim was to explain how segregation residential structures could spontaneously occur, even when people are not so very segregate themselves. Local interactions are sufficient for spatial homogeneous patterns to occur; spatial segregation is an emerging property of the system’s dynamics, while not being an attribute of the individual agents. The basic Schelling model is very simple. It did illustrate a surprising phenomenon: the “tolerant” households could generate residential segregation through their local decisions. The Schelling model has been used as an explanation for the persistence of residential segregation despite all the positive, progressive social policies, such as antidiscrimination laws and affirmative action policies. Gilbert [19] has shown that some features that seem to be fundamental to human societies, such as the ability to recognize global features, could be added to the basic Schelling model and only yielded the same type of clusters of similar agents. Although the results may vary slightly in the form of the clusters and the degree of clustering, it is not plausible to conclude that these added features significantly improve the original basic model.

Recently, several physicists have used simulation models to study patterns that can emerge when many humans interact, including human trails [20], traffic jams [21], Mexican waves [22], and panic behavior of pedestrians [23]. In these models humans are represented as particles with variation in speeds and/or position, and without any requirement of cognition for the agents. In some applications of financial markets, agents are explicitly called ‘zero intelligence agents’ to show that full rationality is not required to explain observed patterns in

economic statistics. These simple reactive agent-based simulation models have often provided surprisingly apt accounts of empirically observed behavior. In the next section, we used this particle principle implemented an agent based model and simulation. We followed the KISS (keep it simple, stupid) principle [24] for implementing the self-organized insurgent group ABM. We purposely chose to create highly idealized models that boil down a collective phenomenon to its functional essence. Our ‘meta-goal’ in this research is to discover the principles that explain the phenomena observed empirically. It is possible to illustrate varieties of emergence by using a very simple computational model. To prove the KISS concept can be applied in the ideology agent based model, we first conducted research: developing an ideological model for simulating human organization’s collective problem solving methods, in the next section.

4. Developing Ideology Swarm Social Learning Model for Collective Problem Solvers in Adaptive Environment

The notion of social learning has been used with many different meanings to refer to the processes of learning and the change of individuals and social systems. In this research, social learning refers to the process in which agents learn new knowledge and increase their capability by interacting with other agents directly or indirectly. In this research, a particle swarm ideology social learning model is used to model the group's social learning and collective searching behavior in a dynamically changing environment. Different from the randomly changing environment model used in many research efforts, a new adaptive environment model, which adaptively reacts to the problem solver agent's collective searching behaviors, is proposed. An agent based simulation is implemented for investigating the factors that affect the global performance of the whole problem solver community through social learning. The objective of this research is to apply the swarm metaphor as a model of for social learning in the adaptive environment and to provide insight and understanding of social group knowledge discovery and strategic search in a changing environment.

4.1 Swarm Social Group Model

The swarm social group model describes a population of problem solvers that are affiliated with different groups, seeking to maximize a fitness function V that maps a set of solutions X into real values. The fitness values of all solution X produce the fitness landscape L . Problem solvers use the information provided by other solvers to enhance their capability in finding the highest fitness value solutions in L . The solution landscape L generated by the fitness function V dynamically changes as the problem solvers search for the highest fitness value solution. This demands that the groups not only find the highest fitness value solution v_{max} in a short time, but also track the trajectory of the solution in the dynamic environment. The problem solvers do not have any prior knowledge about the landscape L . The individual solver can share their experience freely with other individuals that belong to the same group. However, the information exchanged between the groups is limited and possibly inaccurate. Social learning behavior occurs when individuals can observe or interact with other individuals. An individual will combine his individual experience and the information provided by other experienced individuals to improve its search capability.

The mathematic model of particle swarm algorithm [25] is used to describe the individual's social learning behavior. The particle swarm algorithm is a swarm algorithm originally developed by Eberhart and Kennedy in 1995 [26], inspired by the social behavior of bird flocks and social interactions of human society. In the particle swarm algorithm, birds in a flock or individuals in human society are symbolically represented as particles. These particles can be considered as simple agents "flying" through a high dimensional solution space searching for a high fitness value solution.

Under the particle swarm metaphor, each individual problem solver is represented as a search particle. The particle moves through the problem solution fitness value landscape to search for a function optimum. Each particle has two associated properties, a current solution fitness value $f(x)$ and a moving velocity v , to represent the particle's experience and searching behavior. Each particle has a memory of its best problem solution location p_{best} in the landscape where the solution can generate the highest fitness value. Each particle also knows the global best location g_{best} , the highest fitness value solution ever found by all other neighbor particles. During each step, in the solution fitness landscape, a particle moves from its current position to a new location based on its velocity vector. The velocity vector is influenced by the particle's previous velocity, its current location, and its p_{best} and g_{best} value. For every generation, the particle's new moving destination is computed by adding the particle's current velocity V to its current location X .

Mathematically, given a multi-dimensional solution space, the i^{th} particle changes its velocity and location according to the following equations:

$$v_{id} = w * (v_{id} + c_1 * rand_1 * (p_{id} - x_{id}) + c_2 * rand_2 * (p_{gd} - x_{id})) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where, p_{id} is the location of the particle where it experiences the best fitness value; p_{gd} is the location at which the particle experienced the highest best fitness value in the whole population; x_{id} is the particle's current location; c_1 and c_2 are two positive acceleration constants; d is the number of dimensions of the problem space; $rand_1$ and $rand_2$ are random values in the range of (0,1). w is called the constriction coefficient [27, 28]. Eq. 1 requires each particle to record its current coordinate x_{id} and velocity V_{id} along the dimension d in a problem space, its personal best fitness value location vector P_{id} and the whole neighborhood population's best fitness value location vector P_{gd} . The best fitness values are updated at each generation based on Eq. 3, where the symbol f denotes the fitness function; $P_i(t)$ denotes the best fitness coordination; and t denotes the generation step.

$$f(P_i(t+1)) = \begin{cases} f(P_i(t)) & f(X_i(t+1)) \leq f(P_i(t)) \\ f(X_i(t+1)) & f(X_i(t+1)) > f(P_i(t)) \end{cases} \quad (3)$$

The P_{id} and the coordinate fitness values $f(P_{id})$ can be considered as each individual particle's experience or knowledge. The P_{gd} and the coordinate fitness values $f(P_{gd})$ can be considered as the best knowledge that an individual can acquire from its neighbors through interaction. The social learning behavior in the swarm algorithm model is mathematically represented as the combination of an individual particle's experience and the best experience it acquired from neighbors for generating the new moving action.

In the canonical particle swarm algorithm, a particle's knowledge will not be updated until the particle encounters a new vector location with a higher fitness value than the value currently stored in its memory. However, in the dynamic environment, the value of each point in the solution fitness value landscape may change over time. The problem solution with the highest fitness value ever found by a specific particle may not have the highest fitness value after several iterations. It requires the particle to learn new knowledge whenever the environment changes. However, the mathematic model in Eq. 3 lacks a knowledge updating mechanism to monitor the change of the environment and renew the particle's memory when the environment has changed. As a result, the particle continually uses outdated experience/knowledge to direct its search, which inhibits the particle from following the movement of the current optimal solution, and eventually, causes the particle to be easily trapped in the region of the former optimal solution. This swarm social group model needs a new adaptive social learning model to enable each particle to automatically detect change in the environment and use social learning to update its knowledge. In this model, there is no specially designed particle to monitor the change of the environment. Each particle will compare the fitness value of its current location with that of its previous location. If the current fitness value doesn't have any improvement compared to the previous value, the particle will use Eq. 4 for the fitness value update. Eq. 4 is slightly different from the traditional fitness value update function provided in Eq. 3.

$$f(P_i(t+1)) = \begin{cases} f(P_i(t)) * \rho & f(X_i(t+1)) \leq f(P_i(t)) * \rho \\ f(X_i(t+1)) & f(X_i(t+1)) > f(P_i(t)) * \rho \end{cases} \quad (4)$$

In Eq. 4, a new notion, the evaporation constant ρ , is introduced. ρ has a value between 0 and 1. The personal fitness value that is stored in each particle's memory and the global fitness value of the particle swarm will gradually evaporate (decrease) at the rate of the evaporation constant ρ over time. If the particle continually fails to improve its current fitness value by using its previous individual and social experience, the particle's personal best fitness value p_{best} as well as the global best fitness value g_{best} will gradually decrease. Eventually, the p_{best} and g_{best} value will be lower than the fitness value of the particle's current location and the best fitness value will be replaced. Although all particles have the same evaporation constant ρ , each particle's updating frequency may not be the same. The updating frequency depends on the particle's previous p_{best} and g_{best} fitness value and the current fitness value $f(X)$ that the particle acquired. The particle will update its best fitness value more frequently when the previous best fitness value is lower and the $f(X)$ is higher. Usually the new environment (after changing) is closely related to the previous environment from which it evolved. It would be beneficial to use the existing knowledge/experience about the previous landscape space to help a particle searching for the new optimal. In this situation, the particle will keep the best fitness value in its memory until the best fitness value becomes obsolete. The fitness value update equation enables each particle to self-adapt to the changing environment.

In this swarm social group model, the social network(s) need to be implemented to dictate the types and frequencies of problem solver agent interactions. The problem solver agent belonging to the same group can exchange information without any restriction. The information exchange between agents belonging to different groups is not as efficient as that within the same group. Because of the delay and misunderstanding, some group members may not be able to share their newest high fitness problem solution to agents in other groups. In this research, an agent based social learning simulation is implemented to investigate how the different social network architectures affect the performance of the whole social community through social learning.

4.2 Adaptive Environment

In the swarm social group model, the self-organized social groups search for a problem solution with highest fitness function value as well as adapt to the changing environment. At the same time, the change patterns of the environment will be influenced by the collective behaviors of the social groups when these collective behaviors are effective enough to alter the environment. We define this kind of environment as an adaptive environment. To simulate the movement of the solutions, a test function generator, DF1, proposed by Morrison and De Jong [29], is used to construct the solution fitness value landscape L . DF1 can generate test functions over a wide range of complexities and simulate various problem dynamics by changing its parameters. It has been widely used as a generator of dynamic test environments [30-32]. For a two dimensional space, the fitness value evaluation function in DF1 is defined as:

$$f(X, Y) = \text{MAX}[H_i - R_i * \sqrt{(X - x_i)^2 + (Y - y_i)^2}] \quad (i=1, \dots, N) \quad (5)$$

where N denotes the number of peaks in the environment. The (x_i, y_i) represents each cone's location. R_i and H_i represent the cone's height and slope. By using the DF1 generator, a sample landscape can be produced as shown in Figure 3. This sample landscape consists of eight different sized cone-shaped peaks randomly located in the problem space.

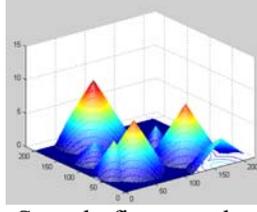


Figure 3: Sample fitness value landscape

The movement of the problem solutions and the dynamic change of the fitness value of different solutions are simulated with the movement of the cones and the change of the height of the cone-shaped peaks. Different movement functions generate different types of dynamic environments. In this research, the environment change rate is controlled through the logic function [29]:

$$Y_i = A * Y_{i-1} * (1 - Y_{i-1}) \quad (6)$$

where A is a constant and Y_i is the value at the time-step i . The Y value produced during each time-step will be used to control the changing step sizes of the dynamic environment. In this research, the dynamic environment is simulated by the movement of the cone's location (x_i, y_i) . The Y value represents the moving velocity of the cone location.

In real-world applications, the evaluated fitness value cannot always be calculated precisely. Most of the time, the fitness value will be polluted by some degree of noise. To simulate this kind of noise pollution in the fitness evaluation, a noise polluted fitness value is generated with the following approach. At each iteration, the fitness value $f(x)$ can only be obtained in the form of $f^n(x)$, where $f^n(x)$ is the approximation of $f(x)$ and contains a small amount of noise n . The function can be represented as [33]:

$$f^n(x) = f(x) * (1 + \eta), \quad \eta \sim N(0, \sigma^2) \quad (7)$$

where η is a Gaussian distributed random variable with zero mean and variance σ^2 . Therefore, at each time, the particle will get a $f^n(x)$ evaluation value instead of $f(x)$.

Another dynamic mechanism of the landscape is the fitness value of the strategic configuration which will gradually decrease with an increasing number of the searching group members that adopt similar strategic configurations.

$$f_i(x, y) = f_{i-1}(x, y) * \left(\frac{1}{e^{(N-1)}}\right) \quad (8)$$

where f is the landscape fitness value of strategic configuration (x, y) at the iteration i . N denotes the number of group members that adopts similar strategic configurations.

4.3 Computational Experiments

The implementations of the swarm social group model and the adaptive environment simulations are carried out in the NetLogo agent modeling environment [34]. Each agent in the NetLogo environment represents one particle in the model. The agents use Eq. 4 to update their best fitness value. Initially, there are 400 agents randomly distributed in an environment that consists of a 100x100 rectangular 2D grid. The grid represents all the possible problem solutions. A dynamic landscape is generated as discussed in Section 3 and mirrored on the 2D grid. The

initial status of the 2D grid is shown in Figure 2. Eight white circuits represent the fitness values of problem solutions, which are generated by the DF1 function discussed in Section 3. The brighter the white circuit, the higher the fitness value is. These white circuits will dynamically move in the grid base on Eq. 6 and the fitness values (brightness of the circuits) are dynamically changed based on Eq. 7 and Eq. 8. The agents are represented as color dots in the grid. Different colors indicate different groups. The social learning of each individual is represented as the highest fitness value broadcast within the group. The searching behavior for finding high-fitness-value solutions is represented as the movement of an agent in the 2D grid. The movement of each agent is controlled by Eq. 1 and Eq. 2, in which c_1 and c_2 are set to 1.49, V_{max} is set to 5, and the w value is set to 0.72 as recommended in the canonical particle swarm algorithm [27]. It is assumed that agents belonging to the same group can exchange information without any restriction. But the information exchanged between different groups will be delayed for a pre-defined number of time-steps and some noise will be added to pollute the value of the information to reduce the information's accuracy. The delayed time-step for information exchange between agent groups is pre-set as 20 time-steps. There is a 20 percent possibility that the information, including the location of the best fitness value and the fitness value itself, is incorrect.

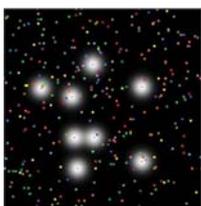
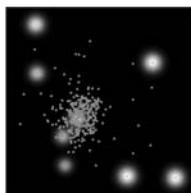
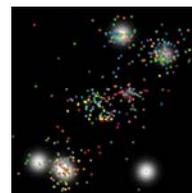


Figure 4: The initial environment and agent groups



(a)



(b)

Figure 5: The collective searching results for scenario (a) one group with 400 agents and (b) Fifty groups, eight agents per group

In this research, we first investigated the change of a particle's social learning performance when the social group structure changed from a single group to multiple groups. The performance evaluation can be generated via computing the average fitness value of all individuals generated in the whole search period. Two different agent group structure scenarios, scenario *a* and scenario *b*, are simulated in this study. In scenario *a*, 400 agents belong to one single group. In scenario *b*, the 400 agents are evenly distributed into 50 different groups with eight agents in each group. Each simulation was run for 500 iterations. The performance of the different group scenarios can then be generated via computing the average fitness value of all individuals generated in the whole searching period. The average fitness value is:

$$av_i = \frac{\sum_{j=1}^k f_j(x)}{k} \quad (9)$$

where k is the number of agents and $f(x)$ is the fitness value of agent j .

4.4 Results

The final agent distribution maps are presented in Figure 5. As shown in Figure 5a, for scenario *a*, all agents belong to the same group. These agents can freely exchange information about their searching performance. Every agent wants to adopt the problem solution that has the highest fitness value. This will cause all agents to swarm around the solution with the highest fitness value. However, because of the dynamic adaptation character of the landscape, the fitness

value of the problem solutions around the highest peak will gradually reduce when the number of problem solver agents around it increases. For scenario *b*, as shown in Figure 5b, limited and noised communication between agent groups make some agents unable to receive the newest information about the best solution that other agents have found. Consequently, agents are distributed relatively even around different solution fitness peaks.

The problem solution searching performance of these two group scenarios is shown in Figure 6, illustrating the average fitness value vs. simulation time step. Initially, scenario *a* has a higher fitness value than scenario *b*, because in scenario *a*, with the help of social learning, all agents can quickly aggregate around the highest peak in the landscape. However, the fitness value in the landscape will adaptively change according to Eq. 8. The congregation of the problem solver agents around the highest fitness value solution will cause a quick decrease in the fitness value of the nearby solution and eventually cause the sum of the fitness value to quickly reduce. As shown in Figure 6, the fitness value of scenario *a* reduces quickly from the peak and remains low. For scenario *b*, because of the delay and inaccuracy of the information between groups, the agents are evenly distributed around all fitness peaks. This distribution makes the fitness value of the nearby landscape not decrease as quickly as scenario *a* and maintains a higher group fitness value than scenario *a* in nearly the whole simulation.

To discover the social network architecture that can generate the highest performance, we tested the performance of different group structures that varied from fully connected social network, in which all individuals belong to a single group, to no connection social network, in which no individuals belong to the same group. The searching performance is recorded as the average fitness value over the whole simulation. The performance chart is shown in Figure 7. According to Figure 7, the performance gradually increases when the agents are divided into large numbers of groups. The performance reveals the highest value when there are 80 agent groups and five agents in each group; then, the performance gradually reduces. The result indicates the problem solver community with a large number of small groups is more efficient than a community with a group that has a large number of members in an adaptive environment.

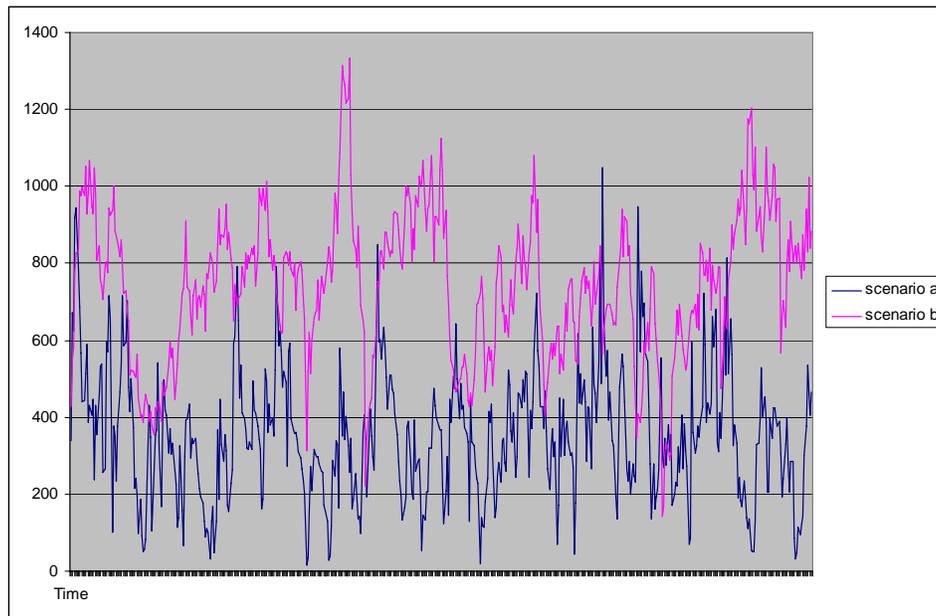


Figure 6: Comparison of the average fitness values for scenario a and b

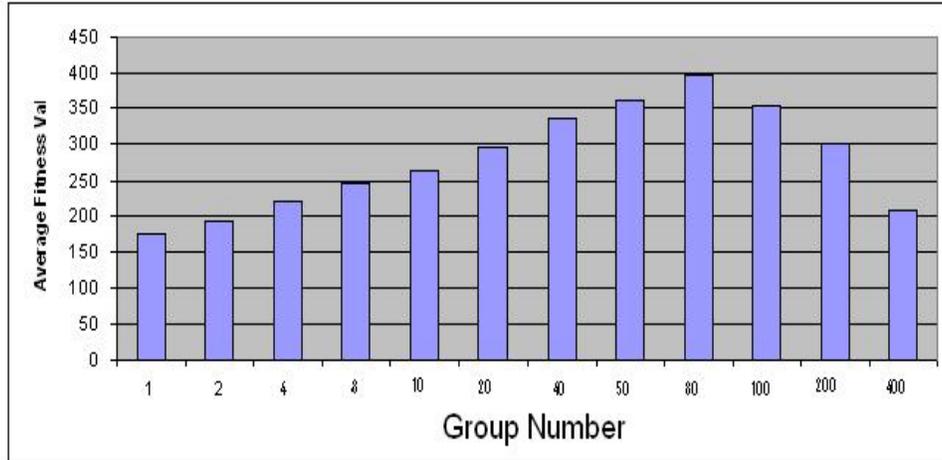


Figure 7: Comparison of the average fitness values of different agent group structures

4.5 Discussion

Most reported searching behavior models only discuss the scenarios in a static environment or a randomly changing environment. The performance evaluation of various approaches is mainly based on how fast an approach can find the optimal point in the benchmark problems. However, the real social world is rarely static and its changes are not random. Most of time, the changes are influenced by the collective actions of the social groups in the world. At the same time, these influenced changes will impact the social groups' actions and structure. In this paper, a modified particle swarm social learning model is developed to simulate the complex interactions and the collective searching of the self-organized groups in an adaptive environment. We constructed a novel agent based simulation to examine the social learning and collective searching behavior of different social group scenarios. Results from the simulation have shown that effective communication is not a necessary requirement for self organized groups to attain higher profit in an adaptive environment.

4.6 Ideology Swarm Model Validation

Part of the hesitation to accept multi-agent and swarm-based modeling and simulation results rests in their perceived lack of robustness. A model needs to be validated before it can be accepted and used to support decision making. The next step in this research will be focused on the self-organized social group model simulation and validation. There are several traditional ways to validate agent-based systems and the choice depends on model complexity and the actual phenomenon investigated. These validation methods include [35-38]:

(1) Systematically comparing simulation model results to data coming from a real world system;

(2) Comparing simulation model results with mathematical model results. This approach has the disadvantage of requiring construction of the mathematical models which may be difficult to formulate for a complex system; and

(3) Docking with other simulations of the same phenomenon. Docking is the process of aligning two dissimilar models to address the same question or problem, to investigate their similarities and their differences, and to gain new understanding of the issue being investigated [39].

Because of the heterogeneity of the agents and the possibility of new patterns of macro behavior emerging as a result of agent interactions at the micro level, model validation in PASS project is different from the traditional validation [40-43]. In the next section, we provided a

validation method with two stages of model validation, corresponding to the two levels at which agent-based models exhibit behavior: the micro level and the macro level. The first stage is the micro-validation of the behavior of the individual agents in the model. In the simulation, agents are not replications of specific human individuals. They are simplified, general representations. The simplicity and generality reduces the ambiguity of any analysis of their behavior and social interaction at the cost of losing expressiveness relative to qualitative studies of observed actors.

The second stage is macro validation of the model's aggregate or emergent behavior when individual agents interact, which can be done by utilizing global network properties and data mining. If the social network created by the simulation has the same network properties; (e.g. statistically similar clustering coefficients and scale-free parameters to that of the empirical social network), then we claim that our simulation social network structure is similar to the real social network. Data mining offers a novel validation technique, because we can perform the same data mining and clustering algorithms on the simulation data, and the algorithm tells us what are considered the most significant factors. Therefore, if data mining produces the same factors and the same clusters for the simulation data as for the empirical data, then we claim that the trends and patterns within the simulation correlate to the same trends and patterns in the real system.

5. Modeling and Validating the Emergence of Self-Organized Groups

We have been leveraging the domain of the Open Source Software development to extract concrete measures of individual and group behaviors in order to create an emergence model of self-organized human groups. The eventual task in this project is building a highly ideological agent-based model to represent self-organized insurgent organizations. Beyond their special features, the insurgent groups display some overall patterns of organization not far from the ones seen in other types of self-organized human organizations [1, 3, 4]. To avoid the limited data for actual insurgent groups and leverage the high availability of historical data about some online self-organized social communities, we use the online OSS communities as a metaphor for insurgent groups in our ideological agent based model research. Understanding these principles offers a first step toward quantitative reference models to explain emergent social behaviors in insurgent groups.

We used detailed OSS real world historical data on SourceForge to illustrate and validate the model's theoretical mechanisms. SourceForge, an online center for OSS development communities, provides collaborative resources for approximately 200,000 projects. This data consisted of all the activity information of OSS software developers and users that registered on SourceForge from 2003 to 2008. We developed scripts that query the SourceForge Research Data Archive for project data that meet our criteria. Because this project seeks to understand both the social and technical impacts on projects, establishing a minimum threshold for team size was necessary to insure that the social element was properly represented. We were interested in projects that reached a minimum team size of 20 developers at some point in the project lifetime in order to insure that the social and technical factors were well represented. In addition, we eliminated projects that did not appear to use the SourceForge collaboration tools as a significant means for communication and coordination. In all, we identified 67 projects as viable for use as training data for our model. From these data, we can reconstruct how the local behavior of setting up the project teams that created individuals led to the emergence of the systemic network. We created an agent-based model simulation to test the representation of our model. With the simulation we experimentally manipulated the parameter values that were fixed in the archival data to investigate whether our model generalizes to a range of real world cases. In addition, the existence of these data sets allows for the validation of the developed models in terms of their ability to predict group behaviors such as membership gains/losses, group efficiency, and the occurrence of group-level actions.

5.1 Technical Approach

We have developed a model that uses agent technology to simulate both an OSS project group and each individual within that group, in order to characterize its complex social and technical aspects. OSS communities are environments that offer unique insight into the mechanics of communication between individuals, actions of individual contributors, and the popularity and performance of project groups. These elements are tracked and measured to a sufficient level of detail such that they can serve as direct inputs into a model of OSS group behavior. This research leverages that availability of data in order to create a data-based model of the complex system of OSS software development.

Our approach to simulating the complex environment of group behavior is to use an agent-based simulation framework and model each individual as a unique agent capable of making independent decisions. Figure 8 depicts the agent architecture used in this simulation. Software agents are used to represent the behavior of both individuals and also the group. Connecting the agents are interfaces, which are the mechanisms for communicating group state to each individual agent, and for communicating individual contributions to the group agent. The Group Agent was designed to be a data-based model, using Bayesian Belief Networks, which

would exhibit behavior consistent with the collected project metrics. Each Individual Agent is a rule-based model that adjusts its behavior based on the current state of the project. The rules are meant to reproduce each individual developer's degree of satisfaction with the group and motivation to continue to work towards group goals.

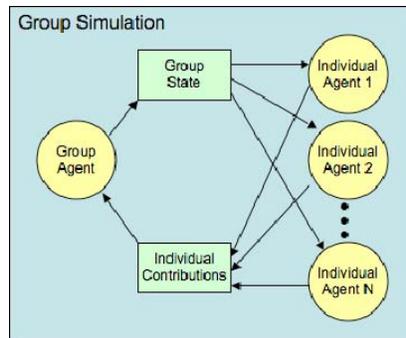


Figure 8: Agent-based Simulation Architecture

In the simulation, information flows cyclically from the group to the individual agents and fed back to the group until stopping criteria are met. The group agent uses the individual contributions to make decisions about group-level events, and individual agents use the group-level events to make decisions about their individual level of contribution. The decisions made by each agent are based on the data extracted from the OSS domain and thus are grounded by the actions of actual OSS projects.

Initially, the project agent and each of the initial developer agents are initialized to either random values or known values, depending on the presence of any known initial conditions. Information flows cyclically from the project to the individual agents and fed back to the project again (as depicted with arrows in Figure 8) until stopping criteria are met. The project agent uses the developer contributions to make decisions about project-level events, such as the occurrence of a software release, or the addition/subtraction of project developers. The developer agents use the project-level events to make decisions about their individual level of contribution in terms of both source code and communication via project message boards. The simulation was implemented using the Multi-agent Simulator of Networks (MASON), a set of libraries provided jointly by George Mason University's Evolutionary Computation Laboratory and Center for Social Complexity.

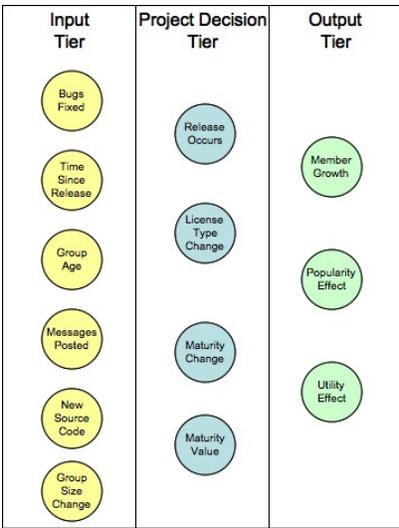


Figure 9: Project Agent Bayesian Belief Net Design

The Project Agent was designed to be a data-based model that would exhibit behavior consistent with the collected project metrics. We used a three-tier Bayesian Belief Network as the probabilistic mechanism for implementing the model, as shown in Figure 9. Software Engineering and project metrics from the Developer Agents are accumulated and entered as known values into the nodes at the Input Tier of the network. Discretization was accomplished by mapping state levels to standard deviations in the underlying metric’s distribution. The six metrics identified at the Input Tier were found to be the most significant factors in a descriptive analysis. The Project Decision Tier infers changes in the state of the project based on the inputs. These changes include whether or not the project is ready to make a software release, or whether the maturity of the project has improved. The Project Agent makes decisions on the state of each of these variables by applying a random number generator to the spectrum of beliefs associated with the appropriate node. The Output Tier uses the inferred states of the Project Decision Tier to deduce the effects of those states on variables such as the utility (number of downloads) of the software product, or the popularity (group ranking) of the project. The states and associated beliefs of the Output Tier are the data that drive the decisions made in the Developer Agents.

Each Developer Agent is a rule-based model that adjusts its behavior based on the current state of the project. The rules are meant to reproduce each individual developer’s degree of satisfaction with the project and motivation to continue to work on it. A Developer Agent’s likelihood to contribute to the project in terms of forum messages, source code contributions, or bug fixes is dependent on that developer’s perception of the project’s progress, prestige, and utility. The Project Agent provides probability values for each of these measures to each Developer Agent, in addition to information regarding changes in project membership or whether a release has occurred. The Developer Agent determines a level of participation in the project by applying a random number generator to these probabilities.

5.2 Validation Approach

Validating the group behavior model involves using the OSS data to confirm the ability of the model to characterize the data set, and the ability of the model to make predictions for unknown data. The Accuracy of Fit is a quantitative determination of how well a model represents the underlying data, and is measured through an analysis of the Equality of Means and the Equality of Variances between the modeled values of group behavior and the associated actual values. The Predictive Validity is a quantitative way of determining how well a given

model characterizes an unknown, and is measured using the Average Absolute Error (AAE). This section describes the approach to these activities including an outline of the validation process, and a description of the statistical tests and formulas used to quantify accuracy of fit and predictive validity.

5.3 Validation Process

The general approach to validation of the agent-based simulation of OSS projects is to train the model with the software engineering data and then measure the ability of the model to both accurately represent the distribution of selected measures for a given project and accurately predict the distribution of those measures.

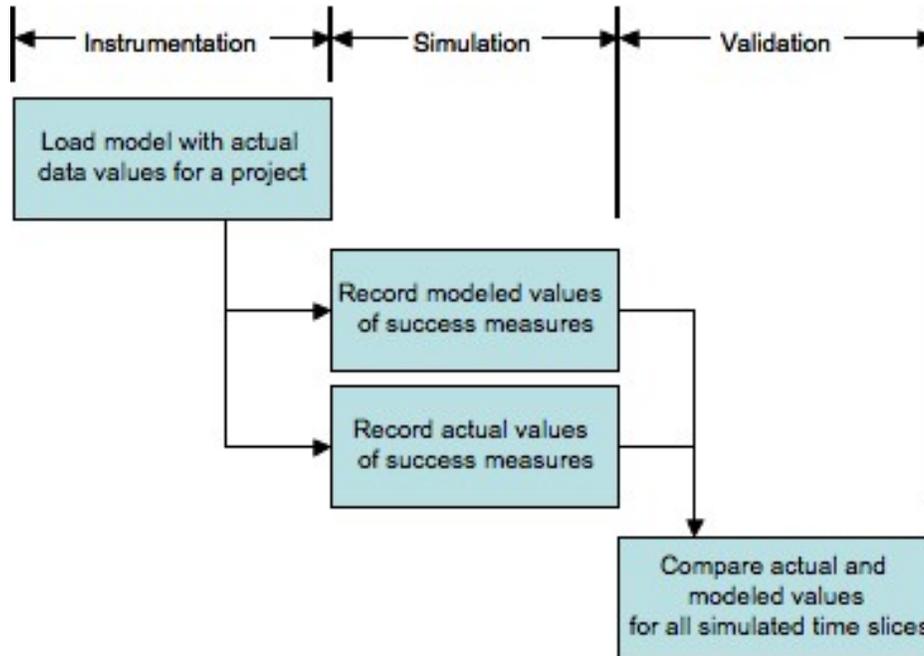


Figure 10: Model Validation Process

The validation process for the agent-based simulation involves three phases, and is depicted in Figure 10. Instrumentation is the phase where the model steps through a predetermined number of time-slices, but loads each of the random variables in the model from the actual data. In effect, instrumentation is the initialization of the model to a specific project. In the simulation phase, the model is tested with both modeled and actual data values for each of the random variables at each of the time slices are recorded. The Validation phase is the statistical comparison of the modeled and actual data. The specific tests that are used to validate the model are described in following Sections.

5.4 Determining Accuracy of Fit

The Accuracy of Fit measure is a quantitative determination of how well a model represents the underlying data. It is a measure that indicates the correctness of the model with respect to the data that was used to construct the model. Accuracy of Fit is determined through an analysis of the Equality of Means and the Equality of Variances between the modeled values of the open source software measures and the associated actual values.

The Equality of Means Hypothesis Test as shown in Figure 11 quantifies the confidence that the mean value of two given populations is equivalent. By comparing the Equality of Means

between actual open source software values and modeled open source software values, the ability of the model to accurately characterize the underlying data set is revealed. If the Test Statistic (t) for the given quality measure is less than the critical value ($t_{n-v, \alpha/2}$) for that measure, then the null hypothesis must be accepted, the means are determined to be equivalent, and the model is said to provide an accurate fit for the underlying data. For this study, a confidence of $\alpha = 0.9$, or 90 percent was used for all Equality of Means calculations. Thus, there is 90 percent confidence that all Equality of Means determinations are correct.

$H_0: \mu_{\text{modeled}} - \mu_{\text{actual}} = 0$	Reject H_0 if: $ t > t_{n-v, \alpha/2}$
$H_a: \mu_{\text{modeled}} - \mu_{\text{actual}} \neq 0$	
where,	$t = \text{Test Statistic} = \frac{\mu_{\text{actual}} - \mu_{\text{modeled}}}{\sqrt{\text{msE} * (2 / n)}}$
and,	
μ_{modeled} = the mean value of the modeled variable	
μ_{actual} = the mean value of the actual variable	
msE = the mean square error	
n = the total number of samples	
v = the number of degrees of freedom	
$t_{n-v, \alpha/2}$ = Student's t-distribution for confidence $(1 - \alpha) * 100\%$	

Figure 11: Hypothesis Test for Determining Equality of Means

This approach to calculating the Equality of Variances is to use a textbook rule of thumb test. The purpose of this test is to compare the modeled data to the actual data if the ratio of the maximum variance value to the minimum variance value is less than three to consider the variances equivalent. The application of this rule of thumb is appropriate in that it bounds the relationship of the variances. The goal for the Equality of Variances is not to get a quantifiable confidence on the accuracy of the modeled data (which is already accomplished through the Equality of Means test), but to get a discrete indication that the variance of the modeled data is on the order of the variance of the actual data.

5.5 Determining Predictive Validity

The Predictive Validity is a measure of how accurately the model predicts a variable using an unknown data set as input. It is a quantitative way of determining how well a given model characterizes an unknown. Predictive Validity is measured using the Average Absolute Error (AAE) and Average Relative Error (ARE). AAE and ARE describe the deviations of the actual data from the modeled data, and are defined in Figure 12 and Figure 13.

$$ARE = \frac{1}{N} * \sum_{i=1,N} \left| \frac{(y_{i,modelled} - y_{i,actual})}{y_{i,actual}} \right|$$

where,

$y_{i,modelled}$ = the modeled value of the variable
 $y_{i,actual}$ = the actual value of the variable
 N = the total number of samples

Figure 12: Calculation of Average Relative Error

$$AAE = \frac{1}{N} * \sum_{i=1,N} | y_{i,modelled} - y_{i,actual} |$$

where,

$y_{i,modelled}$ = the modeled value of the variable
 $y_{i,actual}$ = the actual value of the variable
 N = the total number of samples

Figure 13: Calculation of Average Absolute Error

By definition, the lower the values of AAE and ARE, the more closely the model approximates the actual data. The AAE and ARE in determining Predictive Validity provide assurance that the model developed is reliable in making predictions about unknown data. In this study, ARE was used to validate lower variance variables and AAE is used to validate higher variance variables, where values can confound ARE results. These measures already been confirmed as a significant measure for the Predictive Validity of software engineering models [53, 54], and in the case of ARE, a value of 0.25 or less (within 25% of actual value on average) is considered acceptable to provide a useful prediction.

5.6 Validation Results

This section captures the results of applying the tests for Accuracy of Fit and Predictive Validity to an agent-based simulation that models group behaviors in the domain of OSS development. The validation of the developed model is in terms of five different success measures, shown in Table 1.

Table 1 OSS Success Measures

Success Measure	OSS Domain Metric	Description
Group Maturity	Development Status	Measures the efficiency and effectiveness of the group.
Group Membership	No. of Developers	A count of the group's core membership.
No. of Events	No. of Software Releases	A count of the number of orchestrated actions that the group has performed.
Group Utility	No. of Downloads	Measures the degree to which the group's actions are found to be useful in the community.
Group Popularity	Sourceforge Group Ranking	The popularity of the group in the community.

These five measures are indicators of effectiveness and organization in a group. The goal of this simulation is to accurately represent the underlying OSS data in terms of the measures as shown in Table 1 and to be a reliable predictor of future trends in these measures for each studied group. In this validation exercise, the model was instrumented with actual project data for 9 time slices, and then simulated for additional 1, 3, 6, 9, and 12 time slices. At each interval, an assessment of the accuracy of fit and predictive validity were recorded. Each of the charts that present validation results use the projected (simulated) time slice values to show the trends for the accuracy of fit and predictive validity of each variable as the simulation progressed.

The results of applying the Accuracy of Fit statistical tests, Equality of Means, and Equality of Variances are shown in Figures 14 and 15. The Equality of Means analysis, shown in Figure 14, highlights the statistical threshold for this test as a black line. The interpretation of this graph is that those data points below the threshold line indicate that the simulation was able to maintain a mean value consistent with the underlying data for those metrics. Similarly, data points above the threshold indicate that the simulation diverged from the distribution upon which the simulation is based. Figure 14 shows that the mean values of four of the five OSS success measures were consistent with their actual mean values for up to three time slices. In the case of Group Membership, the mean value was consistent for nine simulated time slices. Thus the model performed well in remaining consistent with the underlying mean values of the data for short-term simulation, but became less consistent as the simulation progressed. The Group Popularity was an exception in this test, and is discussed further below.

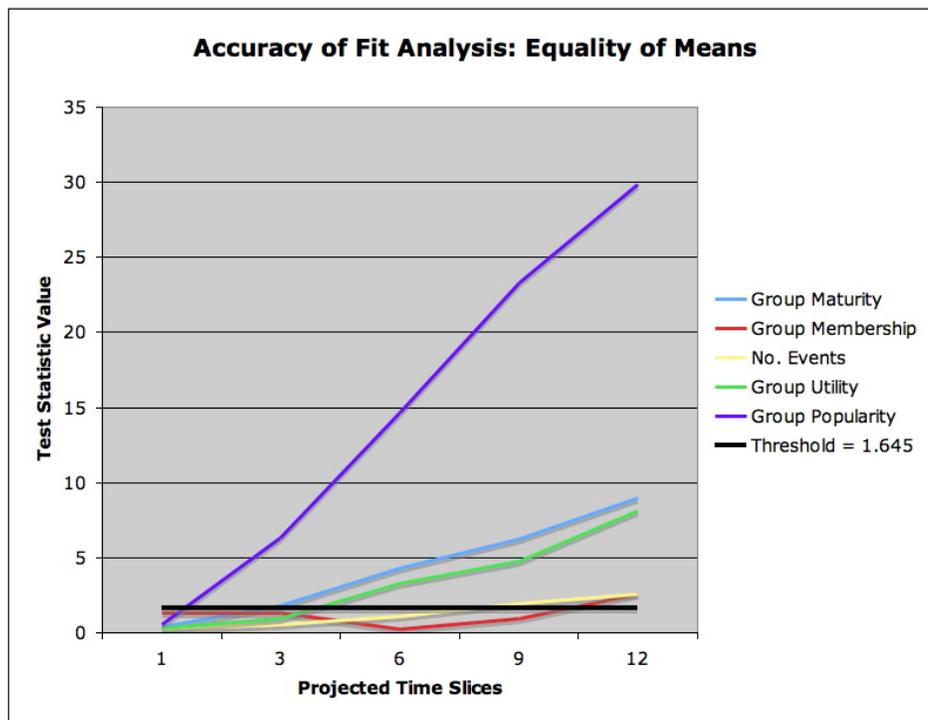


Figure 14: Equality of Means Results

The results of the Equality of Variances test, shown in Figure 15, are similar to the Equality of Means in their interpretation. Values above the threshold line indicate points of unequal variance between actual and simulated data, and values below the threshold indicate points of consistency across the variances. The model did very well at simulating the variances

for modeled values consistent with the underlying OSS data. The most striking exception, as with the Equality of Means test, is the Group Popularity measure.

For both of the Accuracy of Fit statistical tests, the model's lack of ability to represent the Group Popularity measure is unexpected. The developed agent-based simulation performs well in modeling this measure for one time slice, and then its distribution quickly diverges for subsequent time slices. We postulate that Group Popularity is modeled poorly because the assumptions regarding a group's influence on its popularity were flawed. In the agent-based simulation of OSS environments, each measure is calculated with the assumption that the actions and interactions of individuals in the group are causal factors in the emergence of group behavior. In the case of Group Popularity, this assumption does not hold because popularity is relative to the other groups in the environment. For example, Group Ranking, which is the Sourceforge OSS measure for popularity, is relative to the popularity of the other projects on the web site. That is, it is the efficiency and organization of not only one group that affects its popularity, but also all groups that relates to each other. Because the current model is focused on the simulation of a single OSS group and does not account for the presence and efforts of other groups, it is unlikely that it will be able to represent Group Popularity accurately. This is confirmed through the results of the Accuracy of Fit for that measure.

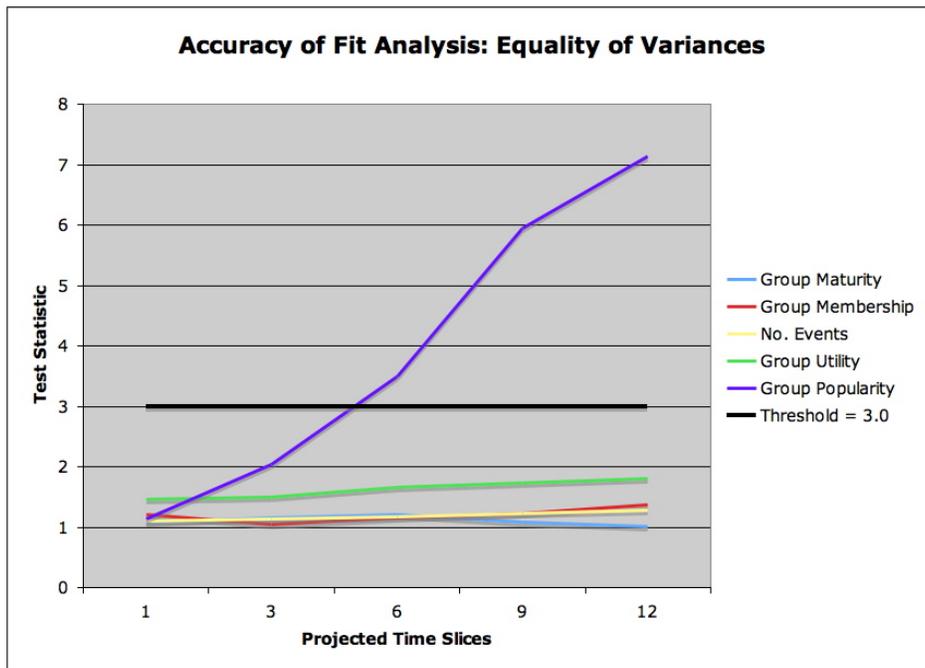


Figure 15: Equality of Variances Results

The results from applying the predictive validity tests, ARE and AAE, are shown in Figure 16 and Figure 17. The lower variance OSS success measures are shown in the ARE results in Figure 16. Recalling that an acceptable threshold for ARE is 0.25 or less, any values below the threshold line indicate that the model is a good predictor for that variable. The chart shows that for early predictions, two of the three variables are acceptable predictors. For the variables the Number of Events and the Group Maturity, this model is a good predictor for up to three projected time slices. Group Membership is not as easily predicted as in the earlier time slices, but converge toward the accuracies of the other variables at approximately the sixth time slice of simulation.

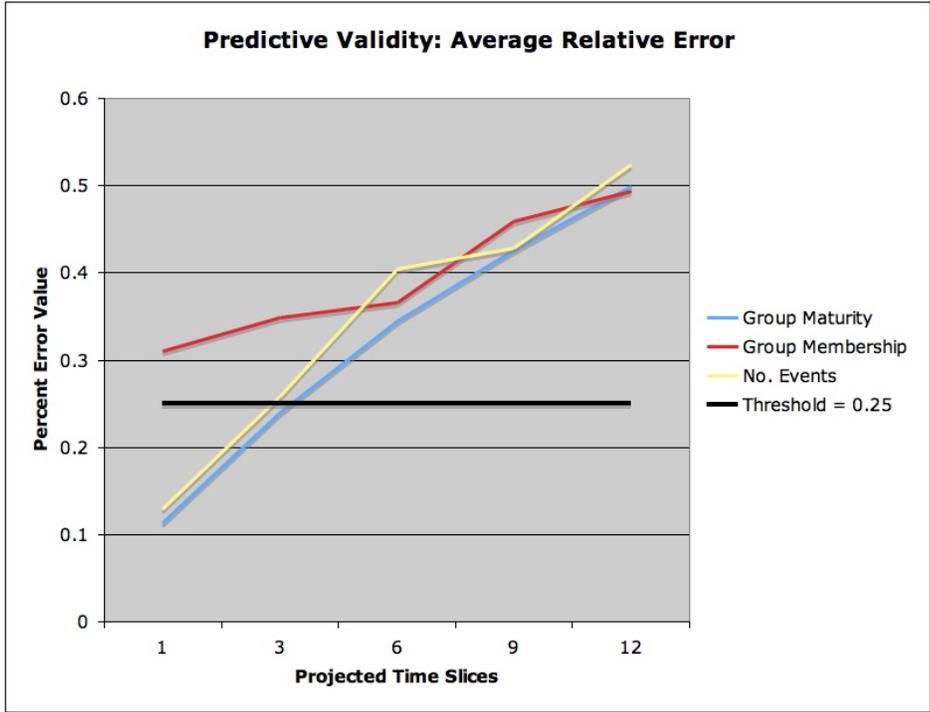


Figure 16: Average Relative Error (ARE) Results

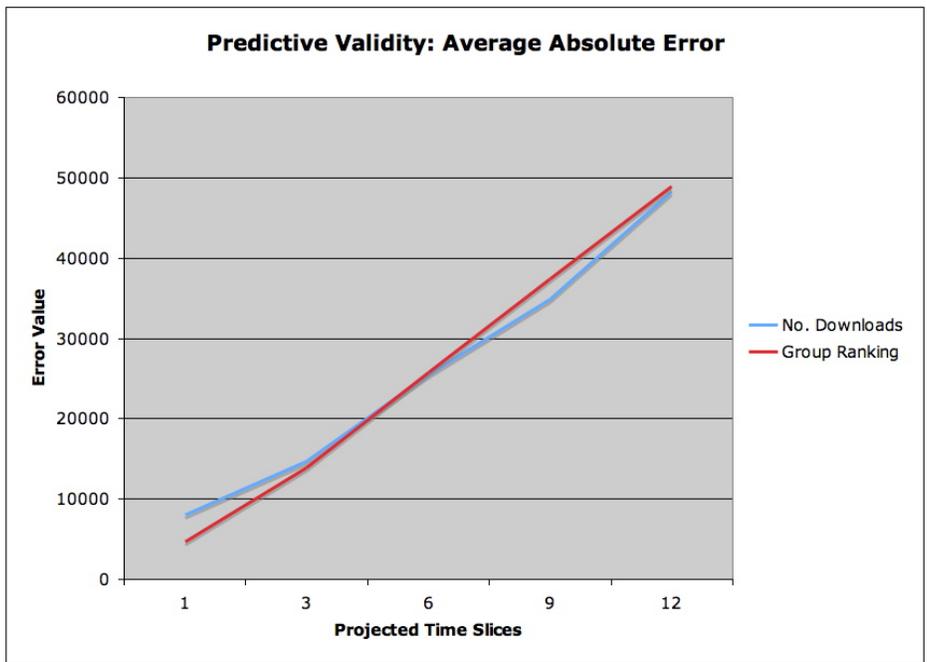


Figure 17: Average Absolute Error (AAE) Results

Figure 17 shows the AAE values for the high variance measures that were used to characterize the OSS project success. These values are not normalized and cannot be compared to a threshold. However, they can be analyzed in the context of their ranges. Group Popularity, for example, is instantiated in the model as the Sourceforge Group Ranking measure, which has a range equivalent to the number of Sourceforge projects (~200,000). So, an average error of approximately 5,000 for Group Popularity in the first time slice is surprisingly accurate given the range of the metric. Similarly, the Group Utility measure, represented in the model as the Number of Sourceforge downloads, is relative to a range of tens of thousands. Similar to the success variables analyzed through ARE, the AAE values for Group Utility and Group Popularity indicate the model is a reasonable predictor of near-term future values; however, its performance degrades significantly as the simulation progresses. More analysis is needed to refine the model such that its predictive performance can be improved.

The results presented in this section are the first steps in simulating the behavior of social groups using OSS data as a basis for modeling. These results are promising as an initial attempt and we expect the Accuracy of Fit and the Predictive Validity to improve further as the model structure is refined and the scope of variables modeled is increased. The intent is to mature this model to be more robust and to represent a full spectrum of group and individual attributes that can be represented through real data sets when available.

5.7 Next Step Research Plan

We have developed a data-based approach for agent based group behavior model that leverages OSS data and agent technology to produce a simulation that accurately represents the source data set. Using this model, we have been able to predict group-level behaviors, such as group membership changes, group efficiency and popularity, and the occurrence of group-level events or actions. We are going to extend this work to analyze the following:

- Model the self-organization of groups in order to identify those factors that contribute to a group attracting members, and those factors that repel members,
- Model an environment with multiple groups that have low-coupling interactions, and
- Apply the developed model to the domain of insurgent groups.

The intersection between a decentralized insurgency and a centralized coalition response is the target zone of the next step research. How does a centralized response to a decentralized actor impact on insurgent activity in an urban environment? Our goal is to provide a model for group behavior that is applicable to a wide variety of group behavior domains.

6. Community Social Network Extraction and Analysis

6.1 OSS Community Network Structures Extraction

This is a preliminary research to explain how micro level actions can account for the formation of different classes of networks in macro level. Micro level actions refer to the choices that actors make in forming their local, direct ties, much like the choices OSS developers make when joining a developing project or civilians make when joining an insurgent group. In these organizations, teams are assembled because of the need to incorporate individuals with different ideas, skills, and resources. Self organized collectives of people create emergent group-level patterns that are rarely understood or intended by any individual. A considerable amount of early work on group behavior from social psychology focused on interpersonal relations and the attributes that characterize good leaders or work teams. However, the social patterns that people form are often organized without explicit leaders, chains of command, or fixed communication networks.

To understand how the self-organizing behavior of individual actors in the OSS community affects the emergence of different types of systemic level networks, we extracted social networks of the OSS community from the SourceForge OSS database, SourceForge Research Data Archive (SRDA), managed by the University of Notre Dame [44]. We used networks of developers who collaborate on OSS project teams as our frame of reference; we investigated how the choices that developers make in deciding their projects and contributions determine the global network topology of the entire field.

Currently, accessing the SRDA programmatically was somewhat difficult. Web service access to the database was approved by Notre Dame. These services were preliminary and relied, on making queries in Perl using the SOAP::Lite library while data retrieval was limited to basic ‘wget’ operations. The amount of submitted messages in the OSS project discussion forums from one developer to other members is a good indicator of his/her social position in the OSS community. Nodes and links (i, j) of the OSS social network represent members and forum message communication from i to j , respectively. For example, at any time, a new software bug is discovered by the member i , a notification message is send in the project bug discussion forum. Then, other expert members investigate the origin of the bug and eventually reply with the solution.

The SRDA had some structural quirks that we noticed. First, each monthly snapshot of SourceForge’s database contained a snapshot of all the discussion forums that existed that month on the SourceForge website. What this means is that for forums that exist in a given month, the database contains all threads ever posted in the forum. However, if a group deleted a discussion forum in a month prior to the database snapshot, none of the threads contained within the deleted forum would be present in the database. Simply put, we could not trust any database snapshot to contain all threads related to a particular group. Given this, we took the approach of querying each monthly snapshot only for messages that were posted that same month. We also were forced to obtain new forum name lists for each month given that they could be created or deleted in any month.

In our work we made some assumptions about the nature of social interaction on these forums. The social network that we generated has edge weights based on the frequency of communication. Larger edge weights indicate a stronger social connection between two individuals for that month. Also, we simplified the representation of thread-based communication (forum threads). When we extracted social links between users participating in the same thread discussion, we assumed that regardless of whom specifically the author was replying to, communication was made with all participants in the thread. Graphically, this results in a clique for every thread. We felt this simplification was reasonable; it allowed us to forgo textural analysis to determine the intended audience of a message post.

Using the PERL scripting language data we extracted from the SourceForge database, we converted into a social network format Dynamic Network Markup Language (DyNetML) [45]. Once all the data was changed to this format, it was analyzed and visualized using the Dynamic Network Analysis tool Organization Risk Analyzer (ORA) [46]. The DyNetML is an XML derivative for representing meta-networks. This language was developed in 2003 by Tsvetovat, Reminga, and Carley [45]. They saw a growing need for a more robust social network data representation solution. Aiming to maximize expressivity, compatibility, and interoperability, they created DyNetML as the new standard in meta-network data representation. In our work, we use DyNetML to represent extracted OSS social net data and to measure and understand Oss social network. Using DyNetML also allows the analysis tool (ORA) to easily import the network data. ORA is primarily purposed as a risk assessment tool for analyzing organizations given knowledge, social, and task network information.

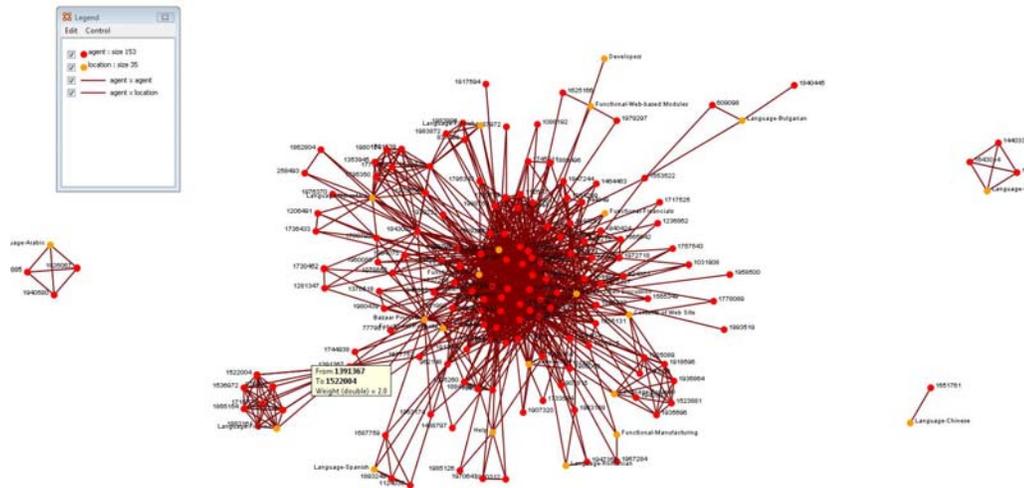


Figure 18: Social Network Visualization for OSS group (175962) at month 15

6.2 Network Structures Analysis

Preliminary analysis indicates some common relationships between OSS projects and fundamental dynamic network analysis metrics. There is a characteristic pattern of asymmetric interaction, where a few core members dominate the activity of the developing team. Many successful open source projects also display a hierarchical or onion-like organization. In many of these communities there is a core team of members who contribute most of the code and oversee the design and evolution of the project. We can identify these core developers by assuming that members with many social ties have leadership roles in the community. The network analysis of open source communities revealed the existence of common statistical patterns of social organization. However, sharing collective properties does not imply the underlying organizations comprise the same micro interaction mechanisms.

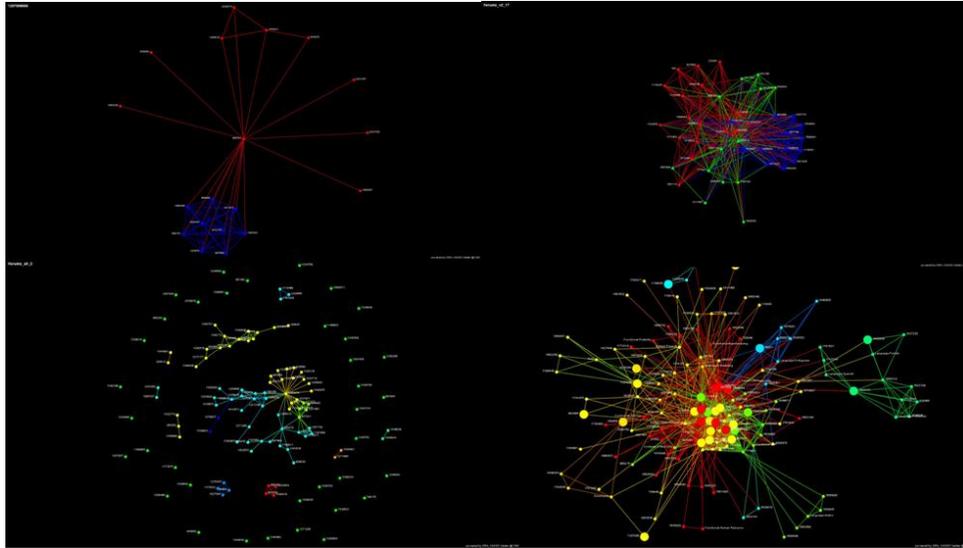


Figure 19: A Variety of OSS Developer Social Networks

As shown in Figure 19, there exists a variety of OSS developer social networks. Different models have been published to explain how the self-organizing behavior of individual actors affects the emergence of different types of systemic level networks. For example, the preferential attachment model argues that new entrants to a network prefer to bond to already highly connected actors, a process that eventuates into a network where many actors have a few ties and a few actors have many ties. Our experiments also indicate that not all OSS groups match the preferential attachment model.

Just as neurons interconnect in networks that create structured thoughts beyond the ken of any individual neuron, so people spontaneously organize themselves into groups to create emergent organizations that no individual may intend, comprehend, or even perceive. Teams are assembled because of the need to incorporate individuals with different ideas, skills, and resources. Self-organized collectives of people create emergent group-level patterns that are rarely understood or intended by any individual. While existing researchers have learned much about how networks govern resource allocation, we know relatively little about how networks emerge. The next step in our research presents our plan for answering these questions.

7. Next Step Research

Modeling Social Network Emergence of Human Community

Previous researches have proved that human social networks (at macro-level) can be represented as complex self-organizing systems that emerge from the interaction of individual social behaviors (at micro-level). In the OSS community, there exists a variety of OSS developer social network structures. Recently, several studies have emphasized the importance of the social structure of human self-organized systems and the impact that structure has on organizational performance. However, several questions about the emergence of network structures still exist. How do systemic-level network structures emerge from local interactions? What local actions suddenly shift or stabilize the systemic-level network? Why are certain networks remarkably resilient? While existing researches have learned much about how networks govern resource allocation, we know relatively little about how networks emerge. The next step in our research will focus on answering these questions.

In Section 5, we developed and validated an agent based model for predicting OSS group behaviors such as membership gains/losses, group efficiency, and the occurrence of group-level actions. In our next step research, we will first examine how the change of individual agent's behavior affects the emergence of different types of network structures. We will explain why some topologies (e.g., small world networks), have high incidence rates relative to other types of networks as described in Section 6. An agent based social model motivated by previous works on social network formation will be implemented. We will use social networks of OSS developers extracted in Section 6 as references and investigate how the choices developers that make in deciding and contributing their projects determine the global network topology of the entire field. Our goal is to evolve the OSS social network model into a generic agent based model that may be able to explain how the self-organizing behavior of individual agents affects the emergence of different types of systemic level networks. It will also help us understand the self organized groups' social network evolution and global network behaviors. The model will be validated at both the individual level and the macro level to illustrate and test the plausibility of the model's theoretical mechanisms.

7.1 Modeling Social Network Emergence of Human Community

Individual actions can lead to emergent features, visible at the societal or macro level. At the same time, such features can also influence or constrain individual action. This approach makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interaction of many individuals. By using agent based simulation, we can effectively describe these behaviors as the actions of agents in an environment, where the agents are the individuals and the environment is the complex self-organizing system. It is possible to reproduce social societies into a synthetic environment by creating "artificial societies" in an agent based simulation.

To experimentally test and refine our team formation model beyond the parameter values that were fixed in the historical case, we will develop an agent-based simulation to test and validate the representativeness of our model. With the simulation, we will experimentally manipulate the parameter values that were fixed in the archival data to investigate whether our model generalizes to a range of real world cases. The model provides a dynamic team formation environment where agent teams form spontaneously in a completely decentralized manner and the agents' decision making is based solely on local information. This model of collaboration networks will illustrate how the behavior of individuals in assembling teams for OSS projects can give rise to a variety of large-scale network structures over time. It is an extension of the team assembly model presented by Guimera et al., [47]. Guimera et al., argued that many of the

general features found in the networks of creative enterprises can be captured by the team assembly model with two simple parameters: the proportion of newcomers participating in a team and the propensity for past collaborators to work again with one another. A key analytical feature of the team formation model [48] is that it suggests that differences in the types of links rather than differences in actors' intrinsic characteristics significantly govern emergence, which enables us to treat emergence in a novel way. The rules of the model draw upon observations of collaboration networks from our understanding of the OSS community.

7.2 Self-motivating teams

Considering that the goal of this research is modeling the common social model of self-organized organizations (OSS community and insurgent groups), we proposed a self-motivating team model. A self-motivating team can be defined as a group of agents being responsible for the performance of a task, whereas each worker possesses a variety of skills relevant to that task. In theory, a self-motivating team has the freedom to allocate and perform the task any way the team likes. Therefore, within self-motivating teams, there must be self-organizing processes. In the model, tasks are generated periodically and globally advertised to the organization. Agents attempt to form teams to accomplish these tasks. Since we are only concerned with the formation process, tasks are generic in that they only require that a team of agents with the necessary skills form to accomplish the specific task. In this model of team formation, the organization consists of N agents, $A = [a_1, a_2, \dots, a_N]$, where each agent can be considered as a unique node in the social network. The network is modeled as an adjacency matrix E , where an element of the adjacency matrix $e_{ij} = 1$ if there is an edge between agent a_i and a_j and $e_{ij} = 0$ otherwise. The social relationships among the agents are undirected, so $e_{ij} = e_{ji}$, and for all agents, $e_{ii} = 0$. In the agent organization, each agent is also assigned a skill matrix $K = [k_1, k_2, \dots, k_n]$ to represent the initial skills of each agent. During the team formation process, each agent can be in different states: Team member or non-committed developer.

7.3 Stigmergic Model for Distributed Task Allocation

A stigmergic model will be implemented for representing the task coordination within the self-organizing groups. A process is stigmergic if the work done by one agent provides a stimulus that entices other agents to continue the job [12]. This concept was initially proposed to explain how a group of dumb, uncoordinated termites manage to build their complex, cathedral-like termite hills. The basic idea is that a termite initially drops a little bit of mud in a random place, and because of the heaps that are formed in this way stimulates other termites to add to them (rather than start a heap of their own), thus making them grow higher until they touch other similarly constructed columns. The termites do not communicate about who is to do what, how, or when. Their only communication is indirect: the partially executed work of the ones providing information to the others about where to make their own contribution. In this way, there is no need for a centrally controlled plan, workflow, or division of labor.

While people, are of course, much more intelligent than social insects and do communicate, the OSS development community uses essentially the same stigmergic mechanism: any new or revised document or software component uploaded to the site of a community is immediately scrutinized by the members of the community that are interested in using it. When one member discovers a shortcoming (e.g., a bug, error, or lacking functionality), that member will be inclined to either solve the problem him/herself, or, at least, point it out to the rest of the community, where it may again entice someone else to take another look at the problem. The more high quality material that is already available on the community site, the more people will be drawn to check it out; thus, the more people are available to improve it further. Open access can profit from a positive feedback cycle that boosts successful projects. This explains the explosive growth of systems such as Wikipedia or Linux in the OSS community.

7.4 Project Setup Model

At each tick, a new task with a skill list requirement is posted on a discussion forum. If an agent has a portion of the skill list the task need, and the agent itself not involve it in any existing project, the developer agent will have ***Ps*** probability to setup a new project. If the agent is committing to a project, the agent will have ***Pm*** probability to add a new function in the existing project to suit the task need. If there are no new tasks or the agent's skill is not suitable for the new task, the agent will consider join in the existing project. Each developer will join a project based on several elements.

8. Relating the Research to DoD Needs

In 2005, Dr. Robert Popp, Deputy Director of DARPA Information Exploitation Office, described the new DARPA initiative for dealing with the 21st-century strategic threat as:

“exploring the innovative quantitative and computational social science methods and approaches that could enable commanders and analysts to understand and anticipate the preconditions Understanding and countering today’s strategic threat requires a wide range of nonlinear mathematical and nondeterministic computational theories and models for investigating human social phenomena”.

Events that have happened in Somalia, Iraq, and Afghanistan indicate that insurgency remains a significant challenge for the U.S.. Conventional methods are not well-suited for analyzing the formation and evolution of these insurgent groups. The U.S. military needs new research into understanding the formation and coordination of enemy insurgent groups, predicting insurgent group behavior, and developing alternate strategies to defeat them. Currently, most ongoing work in insurgency warfare simulation is typically concerned with enhancing existing military capabilities for countering insurgents rather than building a scientific understanding of the insurgency. In terms of modeling, there does not appear to be any mature or widely used methodology addressing insurgency warfare. At the same time, the lack of trustworthy insurgency data makes building and validating such a model a challenge.

In Iraq, the insurgents are a diverse collection of groups including: the Bassthists, the former regime loyalists associated with Saddam Hussein; the Nationalists, mostly Sunni Muslims fighting for Iraqi independence; Sunni Islamists, the indigenous armed followers of the Salafi movement; foreign fighters, largely driven by anti-U.S. feelings and religious doctrine, symbolized by the Jordanian Al-Qaeda operative, Abu Musab al-Zarqawi; Militant followers of Shisa Islamist cleric Moqtada al-Sadr; and non-violent groups that resist the foreign occupation through peaceful means such as the National Foundation Congress. Each group has different reasons for opposing the U.S. occupation, but all have the same goal of forcing the U.S. to remove its troops from Iraq. The violent means to this goal are pursued by most of the groups in a largely decentralized, uncoordinated effort to destabilize the country. Military victory is not a goal of any of the groups. American military supremacy is close to absolute. The commitment of each group varies. The primary military goal of groups like Al Qaeda and Ansar al Sunna is not to win but simply not to lose; to hang on until the U.S. runs out of will and departs.

As we discussed in Section 1, beyond their special features, these insurgent groups may display some overall patterns of organization similar to the ones seen in other types of self-organized human groups, such as the OSS community. The OSS community is considered a complex, self-organizing system. It is comprised of large numbers of locally interacting software developers. Creation of OSS in this community is considered as a collective action by part-time software developers creating and maintaining OSS software for different reasons. Our literature researches have shown that insurgent groups and open source software communities share statistical organization patterns. Our belief is that models of the OSS community can help us understand insurgent behavior and make better use of the limited data available. Supporting validation techniques have been applied in our research to increase confidence in using the OSS community as an analogous data source for insurgent groups. In our next step research, a quantitative comparison between insurgent groups and OSS societies will be conducted to understand the common principles of organization behind them. The resulting particle swarm-based insurgent group model and simulation will be validated for use in future decision making.

The understanding of the self-organized human organizations can help us develop a dynamic insurgent model for simulating the social behavior and interactions in insurgent groups and for understanding the emergence and evolution of these groups. Any model is a

simplification of reality—this one is not unique. Certain complications like heterogeneity among insurgent groups, the effect of third party audiences, and other factors known to associate with the relative rise or decline in numbers of insurgents can be ignored in this ideology model to maintain the simplicity of the model. We believe that a better understanding of the relationships between individuals and the responses of complex systems to external stimuli will provide a slight convex shape to our current COIN method. This proof-of-principle demonstration and research will enable us to better understand the emergence and evolution of self-organized insurgency groups. The results from this study will not only shed light on the evolution of insurgent groups, but will also provide a possible explanation for the formation and evolution of scale-free networks, which are prevalent in insurgent relationships. Furthermore, this research can provide tools for more in-depth analysis of how a community of people congregates, interacts, and learns from each other to form an insurgent group over time.

REFERENCES

1. FILKINS, D., *Profusion of Rebel Groups Helps Them Survive in Iraq*, in *New York Times*. 2005: New York.
2. Marion, R. and M. Uhl-Bien, *Complexity Theory and Al-Qaeda: Examining Complex Leadership*. *Emergence*, 2003. **5**(1): p. 22.
3. Wheatley, M.J., *Leadership of Self-Organized Networks: Lessons from the War on Terror*. *Performance Improvement Quarterly*, 2007. **20**(2): p. 7.
4. Findley, M. and J. Young, *Swatting Flies with Piledrivers?: Modeling Insurgency*, in *Meeting of the International Studies Association*. 2006: San Diego, California, USA.
5. Huddleston, S.H., G.P.L. Sr., and J. Fox, *Changing Knives into Spoons*, in *2008 IEEE Systems and Information Engineering Design Symposium*. 2008: Charlottesville, VA, USA.
6. Epstein, J.M., *Modeling civil violence: An agent-based computational approach*. *PNAS*, 2002. **99**(90003): p. 7243-7250.
7. Solé, R.V., et al., *Selection, Tinkering, and Emergence in Complex Networks*. *Complexity*, 2002. **8**(1): p. 13.
8. Solé, R.V. and B. Goodwin, *Signs of Life: How Complexity Pervades Biology*. 2001: Basic Books.
9. Valverde, S., et al., *Self-organization patterns in wasp and open source communities*. *Intelligent Systems*, 2006. **21**(2): p. 14.
10. Holland, J., *Emergence: from Chaos to Order*. 1998: Addison-Wesley.
11. Goldstein, J., *Emergence as a construct: History and issues*. *Emergence*, 1999. **1**(1).
12. Heyligen, F., *Self-organization, emergence and the architecture of complexity*, in *Proceedings of the 1st European Conference on System Science*. 1989: Paris.
13. Wolf, T.D. and T. Holvoet, *Emergence as a General Architecture for Distributed Autonomic Computing*, in *International Workshop on Engineering Self-Organising Applications 2004*.
14. Gilbert, N. and K.G. Troitzsch, *Simulation for the Social Scientist*. 1999: Open University Press.
15. GILBERT, N. and R. Conte, *Computer simulation for social theory*, in *Artificial societies: The computer simulation of social life*, N. GILBERT and R. Conte, Editors. 1995, UCL Press: London.

16. Younger, S., *Reciprocity, Normative Reputation, and the Development of Mutual Obligation in Gift-giving Societies*. Journal of Artificial Societies and Social Simulation, 2004. **7**(1).
17. Starbuck, W.H., *Organizations and Their Environments*, in *Handbook of Industrial and Organizational Psychology*, M.D. Dunnette, Editor. 1976, Rand: Chicago. p. 1101.
18. Schelling, T., *Micromotives and Macrobehavior*. 1978, New York.
19. Gilbert, N., *Social agents: ecology, exchange, and evolution*, in *Agent 2002*. 2002: Chicago.
20. Helbing, D., J. Keltsch, and P. Molnár, *Modelling the evolution of human trail systems*. Nature, 1997. **388**: p. 3.
21. Resnick, M., *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds* 1997, Cambridge: The MIT Press.
22. Farkas, I., D. Helbing, and T. Vicsek, *Social behaviour: Mexican waves in an excitable medium*. Nature, 2002. **419**.
23. Santos, G. and B. Aguirre, *A critical review of emergency evacuation simulation models*, in *Building Occupational Movement during Fire Emergencies Workshop*. 2004: Gaithersburg, MD.
24. Axelrod, R., *Advancing the Art of Simulation in the Social Sciences*, in *Simulating Social Phenomena*, R. Conte, R. Hegselmann, and P. Terna, Editors. 1997, Springer: Berlin. p. 19.
25. Chio, C.D., R. Poli, and P.D. Chio. *Modelling Group-Foraging Behaviour with Particle Swarms*. in *9th International Conference Parallel Problem Solving from Nature*. 2006. Reykjavik, Iceland: Springer.
26. Eberhart, R. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. 1995. Nagoya, Japan: IEEE.
27. Clerc, M. *The swarm and the queen: towards a deterministic and adaptive particle swarm optimization*. in *Proceedings of the 1999 Congress on Evolutionary Computation*. 1999. Washington, DC, USA: IEEE.
28. Clerc, M. and J. Kennedy, *The particle swarm-explosion, stability, and convergence in a multidimensional complex space*. IEEE Transactions on Evolutionary Computation, 2002. **6**(1): p. 58-73.
29. Morrison, R.W. and K.A. De Jong. *A test problem generator for non-stationary environments*. 1999. Washington, DC, USA: IEEE.
30. Angeline, P.J. *Tracking extrema in dynamic environments*. 1997. Indianapolis, IN, USA: Springer-Verlag.
31. Blackwell, T. and J. Branke. *Multi-swarm optimization in dynamic environments*. 2004. Coimbra, Portugal: Springer-Verlag.
32. Blackwell, T.M., *Particle swarms and population diversity*. Soft Computing, 2005. **9**(11): p. 793-802.
33. Parsopoulos, K.E. and M.N. Vrahatis, *Recent approaches to global optimization problems through particle swarm optimization*. Natural Computing, 2002. **1**(2-3): p. 235-306.
34. Tisue, S. *NetLogo: A Simple Environment for Modeling Complexity*. in *International Conference on Complex Systems*. 2004. Boston, MA.

35. Xu, J., Y. Gao, and G. Madey. *A Docking Experiment: Swarm and Repast for Social Network Modeling*. in *Seventh Annual Swarm Researchers Meeting (Swarm2003)*. 2003. Notre Dame, IN.
36. Banks, S., *Exploratory Modeling for Policy Analysis*. Operations Research, 1993. **41**(3): p. 435-449.
37. Sargent, R.G. *Verification and Validation of Simulation Models*. in *Proc. 2003 of Winter Simulation Conference*. 2003. New Orleans, Louisiana.
38. Macal, C.M. and M.J. North, *Validation of an Agent-Based Model of Deregulated Electric Power Markets*, in *North American Association for Computational and Social Organization (NAACSOS) Conference*. 2005: Notre Dame, Indiana.
39. Axtell, R., et al., *Aligning simulation models: a case study and results*. Computational and Mathematical Organization Theory, 1995. **1**(2): p. 18.
40. Moss, S. and B. Edmonds, *Sociology and Simulation: Statistical and Qualitative Cross-Validation*. American Journal of Sociology, 2005. **110**: p. 37.
41. moss, S., *Alternative Approaches to the Empirical Validation of Agent-Based Models*. Journal of Artificial Societies and Social Simulation, 2007. **11**.
42. Fagiolo, G., A. Moneta, and P. Windrum, *A critical guide to empirical validation of agent-based economics models: Methodologies, procedures, and open problems*. Computational Economics, 2007. **30**(3): p. 31.
43. Midgley, D.F., R.E. Marks, and D. Kunchamwar, *Building and assurance of agent-based models: an example and challenge to the field*. Journal of Business Research, 2007. **60**(8): p. 9.
44. Christley, S. and G. Madey, *Collection of Activity Data for SourceForge Projects*. 2005, Dept. of Computer Science and Engineering, University of Notre Dame: Notre Dame, IN.
45. Tsvetovat, M., J. Reminga, and K. Carley, *DyNetML: Interchange Format for Rich Social Network Data*, in *NAACSOS Conference 2003*. 2003: Pittsburgh, PA.
46. Carley, K. and J. Reminga, *ORA: Organization Risk Analyzer*. 2004, Carnegie Mellon University, School of Computer Science, Institute for Software Research International.
47. Guimera, R., et al., *Team Assembly Mechanisms Determine Collaboration Network Structure and Team Performance*. Nature, 2005. **308**(5722): p. 5.
48. Uzzi, B., et al., *Emergence: The Dynamics of Network Formation*, in *Annual meeting of the American Sociological Association*. 2006: Montreal Convention Center, Montreal, Quebec, Canada.