

The Swarm Model in Open Source Software Developer Communities

Xiaohui Cui*, Justin Beaver*, Everett Stiles[†], Laura Pullum*, Brian Klump*, Jim Treadwell*, and Thomas Potok*

*Oak Ridge National Laboratory

Oak Ridge, TN 37831

Email: cuix@ornl.gov

[†]Electrical Engineering and Computer Science Department

University of Tennessee, Knoxville, TN 37996

Email: estiles@utk.edu

Abstract—Most of the current swarm model studies and applications try to mimic the collective behaviors of social animals, such as birds and ants. The studies seek to solve tasks similar to patterns and behaviors exhibited in those animal colonies. In this research, we demonstrated that the swarm model is also the major collaboration and organization model of Open Source Software (OSS) developer communities. OSS developers swarm together and spend their time attempting to achieve their relatively simple goals, while their contributions emerged as a collection of useful and sophisticated functionality that can compete with commercial software. The results discovered in this research will be helpful in demonstrating that the swarm model can not only be considered as a feasible approach to classical optimization problems, but can also be applied to constructing highly sophisticated systems.

I. INTRODUCTION

The swarm model is one of the major results of research on collective behaviors emerging from social animal or insect groups. More than 50 years ago, biologists reported that a different kind of intelligence could emerge from some social insects, fish, birds, and mammals [1][2]. For example, African termites can build mounds that may reach a diameter of 30 m and a height of 6 m [3]. These biological skyscrapers are built by millions of tiny (1-2 mm long) and completely blind individuals. There are many more examples of the impressive capabilities of social insects. However, it is hard to explain the complexity of all the behaviors at the colony scale level by only considering individuals of the colony. In most cases, a single insect is not able to find, by itself, an efficient solution to a colony's problem, though the society to which it belongs finds, "as a whole", a solution very easily [4]. For a long time, these kinds of collective behaviors emerging from social insects or animal society have remained mysterious and something happens as if there was an invisible hand or leader inside the colony that would coordinate the individuals' activities.

In 1950, Grasse [3] introduced the concept of stigmergy in conjunction with his research on termites. His study showed that a particular configuration of a termite colony's environment could trigger a termite to modify its environment, for example, dropping mud in a particular place to build or maintain the nest. The modification in turn stimulates the

original or other termites in the colony to further transform their environment. The concept of stigmergy provides a theory for explaining how distributed, ad hoc contributions from individuals can lead to the emergence of large collaborative enterprises. The stigmergy model is considered as part of a swarm model that is inspired by nature. The typical swarm model in an intelligent system has the following properties: 1) it is composed of many individuals or agents; 2) the individuals are relatively homogeneous; 3) the interactions among the individuals are based on simple behavioral rules that exploit only local information (stigmergy); 4) the overall behavior of the system presents higher level of complexity than does each individual [2].

In recent years, some computer scientists have adopted the swarm model to solve complex problems and have categorized it as Swarm Intelligence. Swarm Intelligence is an artificial intelligence technique involving studies of swarm models in decentralized systems. Currently, most of the swarm intelligence studies and applications try to mimic the collective behaviors of social insects or animals. They seek to solve tasks strictly by relying on simple individual capabilities, local interaction mechanisms and indirect communication (stigmergy). When the swarm model is used to develop a system with a higher level of sophistication for which no similar phenomenon can be found in social animals or insects, the relative simplicity of individual agents sometimes raises questions about their ability to execute such complex cognitive tasks. Recent publications [5] attempt to prove that the stigmergic swarm model is at least as powerful as a Turing machine and the swarm model-based system can be a feasible approach for more sophisticated systems. Considering that most existing swarm algorithms are inspired from phenomenon of animal colonies in nature, it will help inspire new swarm intelligence algorithms by going back to nature to explore the more sophisticated communities that organize and collaborate in a swarm structure. Recent research [6] indicates that some human communities are known to display a large amount of distributed and bottom-up structure. In these communities, the social patterns that people form are often organized without explicit leaders, chains of command, or fixed communication networks. Examples of such spontaneously emerging social groups include fans at a sport

stadium, grassroots political movements, terrorist networks, Wikipedia authors, and open source software (OSS) developer communities.

In this paper, we report our research results on the swarm model we discovered in OSS communities. The result demonstrates that the OSS community and their achievements can be viewed as a collective intelligence system. The OSS product is the result of collective workings of individual developers with different motivations and personal goals. OSS developers swarm together and spend their time attempting to achieve their relatively simple goals. It is the swarm model adopted by the OSS community that helps the community merge the collection of useful functionality that can compete with commercial software. In the next section, we first briefly describe the open source software phenomenon along with its surprising impacts on business and society. In section III, we provide some related research about OSS communities. In section IV, the experiment design and the data source are discussed. The experiment results are discussed in section V and the final conclusion is given in section VI.

II. OSS DEVELOPER COMMUNITIES

The OSS developer community is a new online software development group where participants can read, modify, and redistribute software source code without cost. Global distributed online communities collaborate to create useful and sophisticated computer software. The online software development teams are composed of unaffiliated individuals and organizations who work in a seemingly chaotic fashion and who voluntarily participate in the development team without direct financial incentive. In recent years, these communities have had a surprisingly powerful impact. For example, 78 million web server sites now utilize the software products which were created and freely distributed by the Apache OSS developer community. Most organizations and individuals can now benefit directly from the computer programs being produced by these OSS communities. Yet, all of this has been accomplished by non-paid volunteers and/or by the employees of corporations who do not directly profit from their employees' activities. These open source developers operate from remote locations around the globe, they choose their own tasks, and they work at their own pace. The result has been described as a kind of "bazaar" of activity [7].

In the software development field, a long-held belief is that large scale software development should be regarded as a collaborative activity to be conducted in a hierarchically structured organization such as a company. However, many OSS development systems do not evolve the same as these commercial systems. The distributed and unplanned OSS development model has been shown to be very effective as a software development paradigm and outperforms commercial software development schemes. The best known examples include the Linux operating system, which is starting to compete with Microsoft Windows and gradually replace the Windows operating system which has dominated the computer operating system market for decades.

III. RELATED WORK

The collaboration and organization models of the OSS community have been extensively researched. Some efforts to explain the OSS phenomenon refer to OSS as a new form of organization, a new model for software production, and a new kind of innovation. Crowston, et al. [8] proposed a model for effective work practices in OSS development. The model was based largely on an existing model of group effectiveness initially proposed by Hackman [13] in 1986. Smith, et al. [10] presented an agent-based OSS simulation model that includes the software modules' complexity, the software's fitness for purpose, the motivation of developers, and the role of users in designing requirements. In research on how OSS developers collaborate, research considers the OSS movement as a self-organizing system and a collaborative social network [11]. Social Network Analysis was used for analyzing OSS community structure. However, in the OSS community, direct connections between individuals are unusual. They exchange information through the forum or email-list indirectly. Most of the time, the actor does not even know who the recipient of his/her message is prior to sending the message. Hinds [12] presented that the social network structure of an open source software project community has no important effect on community success. He addressed the question of "how open source software project communities can successfully develop complex artifacts such as software without being impacted by the social network structures of closure, bridging or leader centrality". His hypothesis is - technical artifacts may be substituting for the social network as a knowledge transfer medium, and that the overall need for knowledge transfer within an OSS project may be lower than in a traditional team-based project. The OSS project community is actually neither a traditional "team" nor a "community", but is a new kind of social entity which is built upon a socio-technical development process involving extensive interactions between humans and technical artifacts. This kind of interaction has characteristics similar to those of the stigmergy model in an insect colony.

Elliott [6] argued that collaboration in small groups (roughly 2-25) relies upon social negotiation to evolve and guide its process and creative output. Collaboration in large groups (roughly more than 25) is enabled by stigmergy. Heylighen [13] proposed to distinguish stigmergy in the OSS community as direct and indirect. In OSS development, the unfinished jobs serve as the direct stigmergy, which stimulates other actors to participate in finishing the jobs. Indirect stigmergy can be recognized in forums where bugs or function requests are posted. These forums are regularly consulted by the developers, thus attracting their attention to tasks that seem worth performing. Their results partially supported the hypothesis that the swarm model is a feasible approach for understanding how OSS communities' collective behaviors emerge on the system level. However, the numerical research and the mathematical stigmergy model are not discussed in Elliott's and Heylighen's publications. In our earlier research [14], we proposed a stigmergy collaboration OSS model to produce a simulation

that accurately represents the collaboration phenomenon in an OSS community. The simulation outputs are compared with the empirical data retrieved from actual OSS project log information. The simulation is able to partially reproduce the forum evolution trend in many OSS projects. The closeness of the simulated results to the empirical data indicates that the stigmergy model reflects the collaboration processes that occur in OSS evolution. It also numerically proved the conclusion made by Elliott [6] and Heylighen [13].

IV. EXPERIMENT DESIGN, DATA AND METHODS

As we discussed in the introduction, the typical swarm model in an intelligent system has four properties: 1) it is composed of many individuals or agents; 2) the individuals are relatively homogeneous; 3) there is stigmergic collaboration; and 4) there is higher level complexity in overall behavior than for each individual. To prove the swarm model exists in the OSS community, we need to discover enough evidence from the OSS community to support the hypothesis that the OSS community has similar properties. Earlier research [7][14][15][16] demonstrated that OSS developers' continuing contribution of their effort to an OSS project has an important effect on its success. And many successful OSS projects, such as Linux, Apache, Eclipse, etc., are the collective contributed results of a very large number of OSS developers. Our earlier research [14] also demonstrated that the collaboration in the OSS community is enabled by stigmergy. In OSS communities, artifacts, such as source code, bug reports, etc., substitute for the social network as a knowledge transfer medium, and the overall need for knowledge transfer through traditional social networks within an open source software project are much lower than in a traditional team-based project. In an OSS developer group, while people are much more intelligent than social insects, open software development uses essentially the same stigmergic mechanism for collaboration.

Our aim in this research is to prove two additional properties in OSS communities: the individuals are relatively homogeneous and the overall behavior exhibits a higher level of complexity than do the individuals. Our hypotheses are: 1) most participants within the OSS community only contribute to a single project; 2) most developers are driven to achieve their own objectives while contributing to the OSS community; and 3) virtually all OSS developers do not have global knowledge of the entire project. We will demonstrate that OSS developers' contributions are limited to a narrow scope within the OSS community. We highlight details that reveal most developers contribute to a single project, and loss or addition of any individual developer's contributions has no significant effect on the OSS project's future success. We will address this in two stages. In the first stage we will use an external database, containing information about the OSS communities, to analyze developer participation at a community level. The second stage will encompass the use of the OSS project source code version control system. Using this record we will examine developers' contributions at a project level.

In most online hosting environments, project-related actions are logged and the log information can later be mined to understand the community structure and interaction patterns. This log data provides enormously detailed information for analysis [17][18]. We used detailed OSS community log data on SourceForge to illustrate the model's theoretical mechanisms. SourceForge, an online center for OSS development communities, provides collaborative resources for approximately 200,000 projects and millions of OSS users and developers. This data consists of all the activity information of OSS developers and users registered on SourceForge. The University of Notre Dame [19] hosts a database which maintains a vast amount of data collected from SourceForge. The data collection started with a dump of SourceForge's log content in January of 2003. We developed scripts that query the database for OSS project empirical data that meet our criteria. For example, we eliminated data on projects that appeared to use SourceForge only as a means to download the software, not as a development environment. However, the data extracted from University of Notre Dame-hosted database is not detail enough to determine a single developer's contributions to a given project. We use Concurrent Versions System (CVS) data retrieved from SourceForge server to bridge this gap and calculate developers' contributions to a project. CVS maintains the history of all changes within a project. This system, an OSS project itself, is used widely throughout the SourceForge community. Its purpose is to track changes within source code and record appropriate information that can be used to revert back to previous versions. This information includes details on the change along with the author who committed it. By using the CVS historical record, we are able to get each individual developer's contribution to a project at any given time by counting the lines of code he/she had modified or developed at that time. For sophisticated software, we assume that the percentage of knowledge an OSS developer acquired about the OSS project depends on how many lines of code he/she contributed and how many source code files he/she generated. Different from the traditional software development project, where the software functions are pre-designed by the software developers before they start the actual programming, the OSS project normally has no pre-design stage. The functionality of the OSS project is not fixed and it continually evolves with the OSS developers' contributions. The more contributions a developer provides, the more effect the project receives from the developer's personal knowledge. In our research, the CVS data was collected directly from SourceForge through the servers that they provide. Using their system and a standard CVS client, the entire source code repository for a number of selected projects was downloaded. Through further use of the CVS client, the project's history was extracted using a standard command named 'log'. This is the data used in the research described in this paper.

V. RESULTS

A. Participant's Community Contributions

To demonstrate our first hypothesis, namely, that most participants within the OSS community contribute to a single project, we collected data from the SourceForge database that reflected participation within OSS projects. The original data include projects that have been inactive for some time. To receive results that more accurately describe the month being analyzed, the data was filtered each month to include only active projects. Active projects were defined as projects that had issued a software release within the two years prior of the month being evaluated. For each month that data was available, we calculated the percentage of developers who participated in one project, two projects, and so on. The data for each month was an accumulation from all previous months. We have presented these findings in Fig. 1. An almost constant trend from 2003 to present can be observed. Data points that reflect participation during the months of July to September 2007 are corrupted due to a privacy issue SourceForge had with the database that was used. The data demonstrates approximately 85% of developers on active projects contribute to one project, with less than 4% of developers contributing to more than three. A linear trend can be extracted to show that the percentage of developers working on a single project is rising. Likewise, linear trends show that participation in multiple projects is declining.

B. Participant's Project Contributions

The hypothesis that virtually all developers do not have knowledge of the entire project can be proved by examining each developer's participation level within the project. Our assumption is if the developer does not have any involvement with a particular source code file, he will not have knowledge of this source code file. By measuring the percentage of files that each developer had made contributions to within a single project, we can tell each individual developer's knowledge level about the projects. SourceForge's CVS system contained a repository of information that was used to perform this measurement. We analyzed projects with the highest amounts of participation during the month of July 2009. Out of the top 20 projects identified, 10 of these projects, including the top 4, were available to be analyzed using the CVS repository. We measured the percentage of files that developers had made contributions to within a single project. A developer had to edit at least one line of a file to be counted as a contributor. Measurements are listed in Table 1. The number of participants for each project and the percentage of files developers contributed to are indicated. Table 1 indicates that developers have a narrow scope within their projects. In some of these projects, the majority of developers contributed very little to the projects. Since there are relatively no project members participating in all coding efforts within a project, we can conclude that virtually all developers do not have knowledge of the entire project.

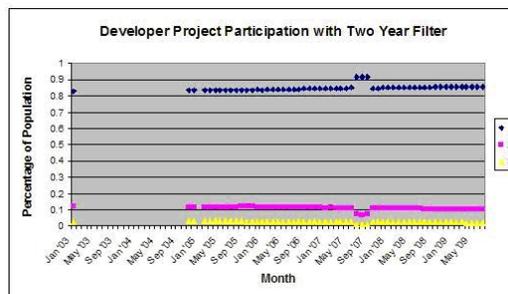


Fig. 1. Developer Participation within Active Projects on SourceForge (Jan. 2003 - July 2009).

VI. DISCUSSION

Past research [20] revealed virtually all participants in the open source community started contributing due to a work-related need. This is combined with the observation that nearly all participants discontinue their contributions within one year [20]. We can assume that the majority of participants are stimulated to contribute based on those needs. Our research results demonstrated that contributions from each developer were very narrow. Contributions from a single developer were mainly funneled into a single project and only impacted a small percentage of that project. This reinforces the notion that developers are driven to achieve their own objectives while contributing to the open source community. A very small number of long-term contributors, who likely started contributing for their own needs, continues to exercise authority over OSS projects. Through code revisions and software releases, long-term contributors guide the progress and future of open source projects. In our findings, we discovered some developers within the open source community that worked on a great number of projects and others that had contributed to a large portion of the project in which they were participating. With most participants working to achieve their own goals and long term contributors maintaining the projects, functioning software is produced. This conclusion supports our hypothesis that the open source community software development phenomenon can be explained by modeling the community as a swarm.

VII. CONCLUSION

Currently, most of the swarm intelligence studies and applications try to mimic the collective behaviors of social insects or animals. This mimicry limited the swarm model algorithms to solving problems that require patterns similar to those in the social insects or animals. Our research about swarm modeling in the OSS community expands the swarm intelligence research from the simple animal colony to the more sophisticated human community. In this paper, we developed a swarm model in order to explain how useful open source software can be created by developers who only contribute to satisfy personal needs. We presented our research results to demonstrate that this swarm model is a reasonably accurate portrayal of the OSS community, thereby correcting the (relatively) long-held

TABLE I
DEVELOPER'S SCOPE WITHIN PROJECTS

Number of Participants	0%	0% -1%	1% -10%	10% -20%	20% -50%	50% -100%
84	61	16	4	2	0	1
93	0	70	20	0	3	0
94	56	22	10	3	1	0
103	34	40	20	3	6	0
112	28	57	20	7	0	0
129	12	83	28	4	2	0
151	77	53	20	1	0	0
162	73	64	24	1	0	0
249	68	136	44	1	0	0
428	204	184	32	4	3	1

assumption that it is the OSS community's intention to work together to achieve a common ideological goal. While this view does not hold for every contributor, it represents the views of a large portion of the OSS community.

Based on our research presented in this paper, we have shown that most participants within the OSS community contribute to a single project and virtually all developers do not have global knowledge of the entire project. The developers make contributions based on their personal needs, while their contributions collect and emerge as a related collection of useful functionality that can compete with commercial software. The concept of the swarm model in OSS provides a theory for explaining how disparate, distributed, ad hoc contributions from individuals could lead to the emergence of the largest collaborative enterprises the world has seen. The results discovered in this research demonstrated that the swarm model can not only be considered a feasible approach to classical optimization problems, but also can inspire the computer scientist constructing highly sophisticated intelligent systems. The result will also help scientists explain how loosely formed open source software developer communities can successfully develop complex software artifacts.

Acknowledgments.: This work was supported in part by the Lockheed Martin Corporation Shared Vision program. The views and conclusions contained in this document are those of the authors. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

REFERENCES

- [1] Beni, G. and J. Wang (1991). Theoretical problems for the realization of distributed robotic systems, Sacramento, CA, USA, Publ by IEEE, Piscataway, NJ, USA.
- [2] Bonabeau, E., M. Dorigo, et al. (1999). Swarm intelligence from natural to artificial systems. New York, Oxford University Press.
- [3] Bonabeau, E., F. Henaux, et al. (1998). Routing in telecommunications networks with ant-like agents. 1437: 60.
- [4] Camazine, S., J.-L. Deneubourg, et al. (2001). Self-Organization in Biological Systems. New York, Princeton University Press

- [5] Christley, S. and G. Madey (2005). Collection of Activity Data for SourceForge Projects. Notre Dame, IN, Dept. of Computer Science and Engineering, University of Notre Dame.
- [6] Crowston, K., H. Annabi, et al. (2004). Towards a Portfolio of FLOSS Project Success Measures. Open Source Workshop of the International Conference on Software Engineering (ICSE 2004).
- [7] Cui, X., L. Pullum, et al. (2009). A Stigmergy Approach for Open Source Software Developer Community Simulation. Symposium on Social Computing Applications (SCA09). Vancouver, Canada.
- [8] Elliott, M. (2006). "Stigmergic Collaboration: The Evolution of Group Work." M/C Journal 9(2).
- [9] Franke, N. and E. v. Hippel (2003). "Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software." Research Policy 32(7): 16.
- [10] Gelernter, D. (1998). Machine Beauty: Elegance And The Heart Of Technology Basic Books.
- [11] Ghosh, R., R. Glott, et al. (2002). Free/libre and open source software: survey and study. FLOSS Report, International Institute of Infonomics, University of Maastricht.
- [12] Grasse, P.-P., Ed. (1984). Termitologia. Tome II. Fondation des Socits. Paris, Masson.
- [13] Hackman, J. R. (1986). The Handbook of Organizational Behavior. The design of work teams. J. W. Lorsch. Englewood Cliffs, NJ, Prentice-Hall.
- [14] Hann, I.-H. (2002). Delayed returns to open source participation: An empirical analysis of the Apache HTTP Server Project. Open Source Software : Economics, Law and Policy. Toulouse, France, Carnegie Mellon University.
- [15] Hertel, G., S. Niedner, et al. (2003). "otivation of software developers in open source projects: An Internet based survey of contributors to the linux kernel." Research Policy 32(7): 18.
- [16] Heylighen, F., Ed. (2007). Why is Open Access Development so Successful? Open Source Jahrbuch, Lehmanns Media.
- [17] Hinds, D. and R. M. Lee (2008). Social Network Structure as a Critical Success Condition for Virtual Communities. Proceedings of the 41st Hawaii International Conference on System Sciences, Hawaii, IEEE.
- [18] Jin Xu, Y. G., Scott Christley, Gregory R. Madey (2005). A Topological Analysis of the Open Souce Software Development Community HICSS 2005.
- [19] Kuan, J. W. (2001). Open source software as consumer integration into production, Stanford Institute for Economic Policy Research
- [20] Lakhani, K. R. and E. v. Hippel (2003). "How open source software works: "free" user-to-user assistance " Research Policy 32(6): 20.