PUBLIC GOODS THEORY OF THE OPEN SOURCE DEVELOPMENT COMMUNITY USING AGENT-BASED SIMULATION

Scott Christley, Jin Xu, Yongqin Gao, Greg Madey
Computer Science and Engineering, University of Notre Dame

ABSTRACT

Creation of open source software (OSS) can be considered a collective action by individuals with the mutual self-interest of creating and maintaining a public good. We explain the OSS development process in terms of public goods theory including the notions of connectivity and communality for interactive communication systems; we add an additional feature which is the characteristics of the environment as it interacts with the OSS development community. OSS is more than a simple, classical, physical good; it is a complex, socially-constructed, informational good that has heterogeneous resource requirements and provides heterogeneous benefits to its users. Critical mass occurs not at a singular point in time but is ongoing adaptive process, and the notion of success for a project is dependent upon both the group's cognitive purpose of the software and each member's cognitive belief in the project's success. In this paper, we describe in detail the connection between the OSS development community and public goods theory, and we explore the hypothesis of open source software as a public good by using agent-based modeling and simulation.

Keywords: Open Source Software, Public Goods Theory, Agent-based Simulation, Computational Social Theory

This work is supported in part by the National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829

INTRODUCTION

Creation of open source software (OSS) can be considered a collective action by individuals with the mutual self-interest of creating and maintaining a public good. Public goods theory (Samuelson, 1954) attempts to explain the factors that encourage people to contribute to a public good; four such factors (Marwell et al., 1993; Monge et al., 2003) include 1) identified features of the public good, 2) characteristics of the individuals, 3) characteristics of the group, and 4) the action process of contribution. We add an additional factor which is the characteristics of the environment. For an OSS project, the software itself can be considered the public good, but it would be more accurate to extend the good to also include the project's web pages, discussion forums (email lists, newsgroups), and management tools (bug/feature tracking, source control) because they provide communality (Fulk et al., 1996); that is, they are mechanisms for people to collectively share information and knowledge related to the public good. OSS projects generally have full communication connectivity because of the discussion forums; each individual can easily communicate with every other individual by posting a message, but this does not mean there is full social connectivity as individuals may not read the forums or may ignore messages on those forums. An important notion in public goods theory is that of critical mass (Marwell et al., 1988)

which refers to a point in time when enough individuals have committed resources to make the public good a realization. For OSS, such a critical mass is one criteria, but not the only, for a project to be successful.

In this paper, we describe in detail the connection between public goods theory and the OSS development community. We discuss the various factors of public goods in relation to OSS projects and examine how OSS diverges from the classical theory both in its interaction with the environment and how critical mass and project success are defined. We describe an agent-based model and simulation, especially the challenges involved with correlating social theory with empirical data into a valid simulation. Lastly, we offer some conclusions and directions for future work.

OPEN SOURCE SOFTWARE AS A PUBLIC GOOD

Characteristics of an Open Source Software Project

Open source software projects have all of the features of a public good. As a digital product, it can be easily and cheaply copied for each individual which allows for jointness of supply; that is, multiple individuals can be using the public good at the same time and use by an individual does not limit usage by another. OSS projects also have the impossibility of exclusion because each user has his own individual copy with the right to modify and to distribute, and the OSS licenses disallow any single user to take away usage rights from other users. Open source software projects have the free rider phenomena in that many individuals download and use the software without contributing to the software project. Lastly, open source software is developed through collective action by numerous individuals. These four features are the core characteristics associated with classical physical public goods as described by Marwell et al. (1993). However, Fulk et al. (1996) put forward that communality and connectivity are additional characteristics of public goods for interactive communication systems; OSS projects have these additional features. Connectivity refers to the ability for any member to communicate with any other member; this capability is supplied for OSS projects by technology such as email lists and discussion forums, so there is a very low cost to communicate directly to any and all members. Communality is the notion of a shared body of information or knowledge held by the members, so for OSS projects, this shared knowledge pertains to the project's web pages, frequently asked questions (FAQ) documents, user and reference documentation, wiki's, discussion archives, etc. These additional features mean that the computer software code alone is not the public good, but all of these communal artifacts that surround the software is encompassed by the definition. The broadening of what is included in the public good definition for OSS projects has the implication that the good is more than a simple, classical physical good; it is in fact a complex, informational good. By complex good, we mean the good provides a heterogeneous set of benefits to its users in contrast to the often homogeneous benefit supplied by classical public goods. Open source software as a complex good is put forward by Bessen (2001) as a hypothesis for why OSS provides functionality that is not provided by proprietary software companies. As a complex good, OSS provides multiple features whereby individuals have interest in different subsets of those features; we will expand upon individual interests and benefits in the next section.

Software has a life-cycle that is different from physical goods. Early in the software life-cycle, few features are available to users and little or no benefit can be obtained from the software. As the software develops, additional features are added which broaden the potential users of the

software; also the communality aspects increase as contributors, who are not software programmers, enhance the public good by writing web pages, FAQ documents, report bugs, and suggest new features. This middle period of the life-cycle sees an increase in software complexity as features become interdependent; this causes a corresponding increase in heterogeneous resource requirements for the project. Besides just programmers, the project needs testers to find and report bugs and writers to draft FAQ and other documentation. When resources are scarce, competition appears to determine what features to implement. It is during this period that the critical mass will arise or not to determine if the project will receive sufficient resources to grow into a successful and self-sustaining project. Later periods in the life-cycle are marked by maturity and maintenance tasks; the resource requirements of the software diminishes as few new features are added and most software development is due to fixing bugs or porting the software to different environments.

Characteristics of an Individual

One of the key characteristics is the heterogeneity of individuals involved in OSS projects. Heterogeneous interests play both cooperative and competitive roles; individuals cooperate to attract and to accumulate resources for a particular OSS project; yet within a project, individuals compete for which features to implement, the architecture and design of the software, the specific license to use, and the technology. Each individual holds beliefs about these issues and communicates them publicly to enforce or to direct these beliefs onto other members of the group. The resource capability, be it time or skill, is also heterogeneous for individuals. Some people have the skills to write software code while others may have skills in web design, graphics, or just the ability to install and use the software. Besides having the skill, individuals must also have the time to contribute to a project, and the time availability must match the temporal resource requirements as the project evolves over time. Resources are generally considered fungible (Marwell et al. 1993) which means that they can be reduced to a single metric like money, but this assumption is not valid for OSS projects because resource interdependencies play an integral part in the evolution of the project. Lastly the benefits that individuals gain from using the software as well as the costs they incur from contributing to a project varies from individual to individual. All of these factors play a role in positive and negative feedbacks which attract or repel individuals.

Current research by Xu et al. (2005) has categorized individuals into project leaders, core developers, co-developers, active users, and passive users according to their role and type of contribution to the project. Passive users are the free riders who use the software but do not contribute to the public good in any way whether it be software contributions or just posting messages to the discussion forums. Active users participate in discussions, report bugs, and request features but they do not contribute code. Co-developers, core developers, and projects leaders contribute source code but only the core developers and projects leaders can commit code into the source repository, so co-developers must communicate their code to somebody of the two other categories. Project leaders have the additional authority to perform administrative duties like adding or remove core developers, but they can also delegate some administrative duties to the core developers.

Characteristics of a Group

The group is defined as the set of people associated with an OSS project; the group's communication is characterized by full connectivity and one-to-many messages through the

project's mailing lists and discussion forums. One-to-one private communication exists, often for functional purposes like sending patches to maintainers, but the public forums is considered the primary communication medium. Group norms come into play regarding acceptable usage of these forums for both the form and content of messages. These norms are heterogeneous across projects; some projects may consider posting patches to the forum as acceptable while others do not, or responsiveness to newbie help requests may provoke a detailed answer or a blunt response to read the documentation. These norms also extend to the source code for formatting standards and naming conventions. Each project enforces its norms through an often undocumented and informal action process of correcting an individual whenever they have violated a norm.

Of particular interest are group norms that appear to be shared across many projects; one such norm is the maintenance of ownership responsibility for code a developer has written. When new code is committed into the project's source code, the author of that code is expected to respond and fix bug reports and features requests related to that new code. Developers who do not follow through with their responsibilities may be looked down upon with disfavor, and project maintainers may be less inclined to accept code submissions from that developer in the future. This norm is enforced by many open source licenses which legally require that authors and their modifications are clearly documented with the source code. Such a norm also appears to act as an implementation of division of labor for the project; instead of a manager dividing and assigning tasks, individual developers acquire responsibility in a decentralized and self-selecting manner.

Just like an individual can have cognitive beliefs about an OSS project, the group has a cognitive understanding of the purpose of the project. Such a cognitive structure is inherently dynamic and changes over time as users join and leave the project and as users inject new ideas into group discussions. Constraints imposed by this group cognitive structure feeds back to individuals' self-interests as they compete among themselves for features to implement in the project; this can also be viewed as attempts to shift the group's cognitive goals so that they align more closely to an individual's interests.

Action Processes of Open Source Software Development

In keeping with our definition of OSS as a complex good, the action processes of contribution undertaken by individuals display a great amount of heterogeneity. By action process, we refer to any action undertaken by an individual that translates into collective action for the public good; such actions alter the public good by adding, removing, or reallocating resources. There are many ways that an individual can contribute to an OSS project, and each way requires differing resource commitments of skill and time by the individual. Actions can be categorized as either individual or project actions; however, a project as a socially constructed informational good cannot actually perform an action, so an individual is require to perform the actions on behalf of the project. The type of project action performed by an individual often signifies the role or authority that individual has within the project; conversely, only individuals with the right authority can perform project actions. Project actions include creating a project, adding or removing core developers, committing source code, and releasing file distributions. Individual actions include writing source code, writing documentation, posting messages, reporting bugs, and requesting features. There are many more individual and project actions, but not all can be listed here, and all of them are considered collective actions for the public good.

An individual's position within the group is not fixed; in time, a person can acquire more responsibility and authority for the group, or the person can become more distant with less interaction and contribution to the project. The evolution of an individual's role in an OSS project is a key action process. A current hypothesis is that projects which can increase the responsibilities of members, which implies increased resource commitment, and can retain those members are more fit and have a better chance of success. Likewise, the disengagement of individuals with a high level of responsibility from the project signals a significant loss of resources and diminished survival prospects for the project.

Characteristics of the Environment

Though not generally considered as part of the theory of public goods; the environment is one of the central concepts in agent-based modeling, and it plays an important role in the OSS community. By environment, we refer to everything external to our unit of interest, the OSS project, to things like proprietary software companies, political and economic climate both nationally and globally, legal issues especially regards to intellectual property rights and the difference of those rights between nations, and technological advances and standardization efforts of technologies. Of interest are commercial organizations that are injecting resources into the community either by hiring programmers to work on OSS projects or investing money in organizations that develops open source software. There is the trend of proprietary software companies to support their software on OSS systems as well as to bring open source software internally as components of their proprietary software or as part of their IT infrastructure. These activities have lent endorsement and validity to OSS which has enabled customers to seriously consider open source alternatives to proprietary software. Most of this activity has occurred after the relevant OSS projects have become successful; and while it can provide stability to those projects, such environmental affects are not the cause of the projects' initial success.

Technology plays an important role for how desirable an OSS project is to potential developers and users; such technologies include the programming language, supported operating systems, integration with other tools and libraries, support for standard data formats and protocols, and even the look-and-feel for graphical programs. For OSS projects, technology is part of the environment, and projects may have to adapt to different technologies if they want to attract a larger user base. The environment is not static for OSS projects; it is dynamically changing by bringing new programming languages, new architectural paradigms, new software engineering practices, new standards and protocols, and new modes of communication into existence. Such a changing environment makes new software features appear as well as making some existing features more or less desirable than others.

Critical Mass and Project Success

Critical mass is the point when a public good has received enough interest and resources to be self-sustaining. For a classical public good, critical mass indicates a success point when the public good is realized. For software, the notion of critical mass as a specific event is not so clear because the definition of success is not homogeneous for all software projects. Each project has its own definition of success that is a subjective rating by users of the software; it cannot be computed with a simple numerical measure. The potential user base of an agent-based simulation toolkit is significantly smaller that for a web browser or a word processor, so direct comparison between

projects is not possible. Besides projects having a heterogeneous definition of success, each member within a project has a different cognitive belief in the project's success. This goes back to OSS being a complex, socially-constructed informational good that evolves through interactions between its members and the environment. Critical mass is not a singular point in time but an ongoing adaptive process. Projects which have attained a critical mass can lose it later in time, not just due to people leaving the project, but people joining the project can shift the group's cognitive purpose or goal of the software and increase the resource requirements to make the project successful.

Data mining research (Gao, et al., 2004) has indicated that five temporal factors are significant for clustering projects into categories of success. The factors are the number of developers, the number of file releases, the number of help requests, the number of tasks opened, and the number of tasks closed; all of these factors are expressed as rates of change over time. K-means clustering by these factors partitions the projects into failed, normal, and good projects with decreasing confidence, and the remaining unclustered projects are categorized as excellent projects. The significance of these results is that while standard metrics can categorize failed projects accurately, excellent projects are outliers in the data set. So while these factors offer a good description of what excellent projects are not, they offer little description of what excellent projects are. Our focus on agent-based simulation is then to produce outlier scenarios; analysis of the time evolution and dynamics of those outlier projects should provide hypotheses for why the projects succeed, and we expect that empirical surveys can test those hypotheses.

AGENT-BASED MODEL AND SIMULATION

We explore the hypothesis of open source software as a public good by using agent-based modeling and simulation. Individuals and projects are modeled as agents interacting in a virtual environment, specifically a social network. The environment is also an agent in our model, and the social network is dynamic because both individuals and projects appear and disappear from the network and network links appear and disappear as people join and leave projects. We focus on the time series and evolutionary aspects of the community as individuals join and leave projects as their interests and resource commitments change, and projects are created, abandoned, and mature through a software life cycle.

We are currently in the process of designing and implementing an agent-based simulation for our public goods theory of the OSS development community. There are serious challenges associated with such a simulation for input modeling, parameter estimation, and validation. The primary set of empirical data available to us is a database dump of the Sourceforge community; and while this data set is large, it is missing key attributes of individuals that can usually be obtained for smaller communities through surveys. Likewise, the database dump is a snapshot in time of the community, so not all temporal information is available for us to understand the evolution of the projects and the individuals. We do, however, have very good data analysis about the project and developer networks indicating the existence of scale-free and small world network properties.

Input modeling refers to the process of coming up with an analytical distribution for empirical data that can be used to synthesize input data for a simulation; while, parameter estimation is determining appropriate values to use for the parameters of those analytic distributions like mean and standard deviation. For our model, we need input distributions for attributes of individuals, projects, and the environment. The primary individual attributes are interest, skill, and time. None

of those attributes are directly available in the empirical data, but we can infer skill and time distributions based upon the quantity and type of activities performed by individuals, and interest can be inferred from project attributes for members of those projects. The environment is more difficult because the Sourceforge data has no real measurement of outside influences, but we can estimate the flow of people in and out of the OSS community. More research is needed on the effect of singular events like Oracle supporting Linux or advocacy events like the announcement of a project on Slashdot. Besides the attributes of the agents, the rates for action processes must be estimated. The rate of new projects being created, the membership rate of people joining projects, and the activities within a project like posting messages, reporting bugs, committing source code, release files, etc. are all relevant action processes. As mentioned before, data mining results have shown some factors to be more significant than others, so we will initially focus on actions related to those significant factors in our implementation.

Validation is the process of comparing the simulation's behavior to the behavior of the real system. Here lies the greatest challenge because it is not possible to acquire complete information about the real system in order to perform extensive statistical testing. The approach we take towards validation will utilize global network properties and data mining. If the social network created by the simulation has the same network properties, e.g. statistically comparable clustering coefficient and scale-free parameter, as the empirical social network then we claim that our social network structure is similar to the real social network. Of course many processes can produce a scale-free network, so we cannot claim we have found the correct process, only that we have found a social theoretic process which produces the correct structural properties. 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 the algorithms produce 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. Data mining provides us with a validation technique not just on the raw data but on the meta-data. Though, we must be cautious about this claim as data mining of incomplete information in the empirical data may produce incomplete factors, so our simulation may just be duplicating significant factors in the empirical data instead of the truly relevant factors in the real system.

CONCLUSION

Public goods theory offers a rich social theory for studying the open source software development community. We have described OSS in terms of the four theoretical features of classical public goods: the characteristics of the good, the characteristics of the individuals, the characteristics of the groups, and the action processes of contribution. We have also described OSS in terms of communality and connectivity as additional features for public goods that are interactive communicative systems. Likewise, we have extended public goods theory for OSS by incorporating the characteristics of the environment as a new feature and by redefining critical mass as an ongoing adaptive process. Transformation of the conceptual model into an agent-based simulation poses numerous challenges for input modeling, parameter estimation, and validation; we describe some of these challenges and suggest that data mining algorithms offer a novel form of validation that operates on the meta-data instead of the raw data. Going forward, we look to complete our agent-based simulation and to provide additional insight about the open source development community.

REFERENCES

Bessen, J.E., 2001, "Open Source Software: Free Provision Of Complex Public Goods", available at http://ssrn.com/abstract=278148

Fulk, J., A.J. Flanagin, M.E. Kalman, P.R. Monge, and T. Ryan, 1996, "Connective and Communal Public Goods in Interactive Communication Systems," *Communication Theory*, 6:60-87.

Gao, Y., Y. Huang, and G. Madey, 2004, "Data Mining Project Histories in Open Source Software Communities," in *North American Association for Computational Social and Organizational Science (NAACSOS 2004)*, Pittsburgh, PA.

von Hippel, E. and G. von Krogh, 2003, "Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science," *Organization Science*, 14(2):209-223.

Hunt, F. and P. Johnson, 2002, "On the Pareto Distribution of SourceForge Projects", in C. Gacek and B. Arief (eds.), *Proc. Open Source Software Development Workshop*, pps. 122-129, Newcastle, UK.

Madey, G., V. Freeh, and R. Tynan, 2002, "The Open Source Software Development Phenomenon: An analysis based on social network theory", in *Americas Conference on Information Systems (AMCIS2002)*, Dallas, TX.

Madey, G., V. Freeh, and R. Tynan, 2004, *Modeling the F/OSS Community: A Quantitative Investigation, Free/Open Source Software Development, Edited by Koch S.*, Idea Publishing.

Marwell, G., P.E. Oliver, and R. Prahl, 1988, "Social Networks and Collective Action: A Theory of Critical Mass," *American Journal of Sociology*, 94:502-534.

Marwell, G. and P. Oliver, 1993, *The Critical Mass in Collective Action: A micro-social theory*, Cambridge University Press, Cambridge, UK.

Monge, P.R. and N.S. Contractor, 2003, *Theories of Communication Networks*, Oxford University Press, New York, NY.

Samuelson, P.A., 1954, "The Pure Theory of Public Expenditure," *Review of Economics and Statistics*, 36:387-389.

Xu, J., S. Christley, Y. Gao, and G. Madey, 2005, "A Topological Analysis of the Open Source Software Development Community", in Hawai'i International Conference on Systems Sciences (HICSS 2005).