

Toward an Agent Based Model of Open Source Software Development

Chris Hoffman
MIS Undergraduate
University of Notre Dame
choffma1@nd.edu

Abstract

Open Source Software (OSS) development maintains the interest of researchers worldwide. A number of teams have begun investigating OSS as a self-organizing, collaborative social network. Of particular concern are its implications for proprietary software development firms who may adopt open source practices should they prove superior to centrally managed corporate structures. Also thought provoking is the possibility of extending discoveries of the underlying mechanisms of OSS development to explain the mechanics of social networks in general. Decision or communication structures in OSS forums may correlate to those in such varied social networks as the Internet Movie Database or even terrorist cells.

As the basis for further research of OSS and other social networks, I have coded an agent-based simulation to model the development of the OSS network. I discuss here the design of my simulation, as well as recommendations for future improvement.

Open Source Software Development

Open source software conforms to a two-part definition that stipulates: 1) the software is openly distributed at little or no cost, and 2) the software's source code is open to viewing and modification. Both characteristics of OSS run contrary to the limitations of proprietary software licensing. Proprietary vendors compose price tags and user license agreements to prohibit both such "open" activities.

Free distribution of OSS means its developers receive no monetary compensation for their efforts. Yet, the OSS movement has produced widely renowned successes even in the absence of profit motive. Apache, Linux, GNU, MySQL and a host of other crucial internet infrastructures are open source. No single developer earns royalties from their widespread use (as permeating in Apache's case as a 60% market share according to a Netcraft.com survey, [Netcraft, 2002]).

Some researchers tout the process of OSS development as intrinsically superior to closed proprietary development. They argue that OSS development generates more stable code, more quickly [O'Reilly, 1999; Wu, 2001]. Their assertions, however, are yet unproven.

Nonetheless, a process beneficial to so many economic actors, but devoid of a primary economic motivator deserves attention. Proprietary software developers may adapt and adopt all or some aspects of OSS development should the process emerge truly superior. Governments may discover sufficient interest to subsidize the OSS movement. However, understanding the promise of OSS development requires better understanding of the OSS movement itself, which is precisely our aim.

Social Network Theory

Social network theory is a perceptual framework through which we view the OSS movement. The theory, together with graph theory, depicts interrelated social actors as nodes or points on a graph and their relationships as links or edges drawn between nodes [Wasserman, 1999]. Imagine the circles labeled ‘A,’ ‘B,’ and ‘C’ in Figure 1 represent three people as a highly simplified social network. Each is a node. The lines or edges connecting them could represent such relationships as friendship. Thus, Figure 1 is a graphed social network that describes a three-person friendship. At complexities closer to real world networks, actors connect indirectly through paths of links. In Figure 2, we may consider ‘D’ connected to ‘A’ and ‘B’ through its link to ‘C.’ The relationship illustrated is analogous to ‘D’ being a “friend of a friend” (‘C’) to both ‘A’ and ‘B.’

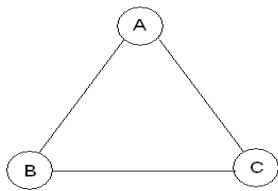


Figure 1 Simple Social Network

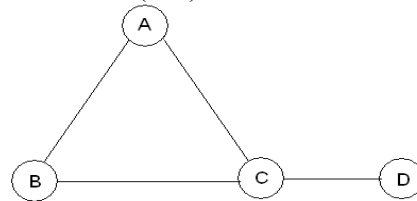


Figure 2 Social Network w/ Indirect Link

Networks of any serious interest are more complex, encompassing thousands of actors and the myriad links possible between them. Despite their complexity, social scientists have discovered common characteristics in social networks. Even as network populations grow exceedingly large, the most direct path between any two actors remains short. Researchers have termed this behavior the Small World Phenomenon [Watts, 1999].

Social networks in which links represent mutual endeavor are collaborative networks. Returning to Figure 1, should we instead imagine that the links represent each person’s contribution to a three-person rock band, the figure becomes that of a collaborative social network. More famous studies of collaborative social networks have examined co-stars in Hollywood films [Watts, 1999; Tjaden 1996] and co-authors on research papers [Barabasi, 2002].

Self-organizing is another classification ascribed to many social networks. A self-organizing network grows and develops structure without centralized decision making. Instead, macroscopic properties evolve from the multitudinous interactions of individual actors. From a vast number of local decisions, global properties emerge. In society, aggregate demand and supply are macroscopic economic properties we see emerge in a

self-organized fashion from the independent purchasing decisions of consumers and producers.

Social networks that are indeed self-organizing tend to inherit another property: a highly skewed distribution [Madey, 2002]. When graphed on logarithmic axis, such distributions map a straight line. Data that produces such a distribution obeys a power-law [Schroeder, 1991]. The source of power-laws is still unknown, though some attest that preferential attachment, or nonrandom attachment, of the nodes in a system is responsible [Barabasi 1999].

OSS as a Self-Organizing, Collaborative Social Network

I adopted the view of my predecessors that the OSS movement is understandable as a self-organizing system and a collaborative social network [Madey, 2002]. I conceptualize OSS developers as its social actors. Their collective progress is self-organizing. Developers schedule contributions individually. No boss or authority directs their activities. As their relationships embody their cooperation towards a common end, the OSS developer's network is indeed collaborative.

In terms of a graphed social network, OSS developers would be its nodes. Collaboration on a software project would be visualized in its links between the nodes. However, my goal was to surpass visualization with actual simulation. Rather than limit our modeling of the OSS movement to legible graphs, we instead sought to develop a virtual representation of a network with similar complexity.

Data Sampling

Of course, no appreciation of the intricacy of the OSS network was attainable without statistical analysis. The Data I inherited examines 16 months of collaborations between January 2001 and May 2002 at SourceForge.net, an online project support site sponsored by VA Software. While not the exclusive repository for open source software, SourceForge does host projects for a large enough population of OSS developers to be considered representative of the movement as a whole. At the time of this writing, SourceForge hosts no less than 56,000+ developers and their 44,000+ projects [Sourceforge, 2002].

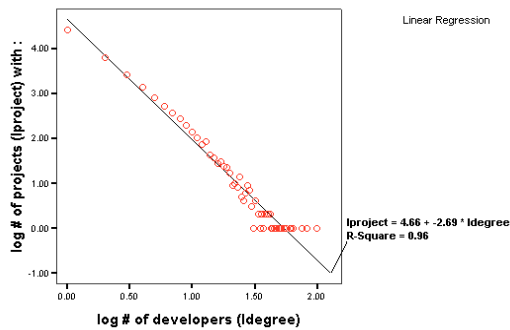
Before manipulation beyond the initial collection, the data consists of a two-column table. The first field contains a developer's identification number, unique to each developer; the second contains a similar number attributed to a single project. This format factored in the design of my simulation.

From the initial dataset, scripts extracted statistics describing both developers and projects to text files. These in turn fed data analysis programs like Mathematica and MATLAB, with which previous researchers examined project and developer distributions based on degree, or level of involvement. A developer's degree is, thus, the number of projects to which he contributes. For a project, "degree" refers to the number of

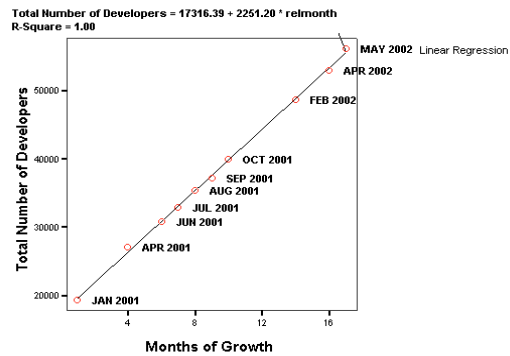
developers it employs. A power law was discovered for each distribution. This revelation, too, affects my simulation design. Graph 1 shows a power law for project degree distribution in OSS.

Apart from distributions, I also studied trends apparent in the data. I analyzed and graphed the values of nine variables that classify projects and developers across the 16 months of data collection. Linear regression of each trend produced growth rate constants. Eventually, verification of simulation data will depend on these. Graph 2 below shows example output from trend analysis.

Graph 1 Project Degree Distribution



Graph 2 Sample Trend Analysis



Simulation Design

Accurately simulating OSS development as self organizing relied on agent-based modeling [Terna, 1998]. In an agent-based model, each individual data object, or agent, contains its decision logic. Agents behave according to this logic alone. No central authority commands or controls them. Thus, agents self organize; agent-based modeling simulates self organization. Hence, I modeled OSS development using agent-based modeling.

In my model, open source software developers are the agents. Each is an instance of a Java class with methods that encapsulate a real developer's possible daily interactions with the development network. Developers can create, join, or abandon a project each day or continue their current collaborations. A separate Java method models each of the first three possibilities. A fourth method encapsulates a developer's selection of one of the three alternatives. Here, three model parameters appear. Each represents the probability of one of the three developer activities. Comparison of a randomly generated number to these probabilities determines which behavioral method the agent will enact. The trend analysis discussed in the previous section provides a benchmark by which to judge calibration of these parameters and determine future adjustments. The table below describes parameters in the current program version.

Parameter Name	Description of Value
probCreateInitially	Probability that a developer creates a project during his first time slice
probCreateEachPeriod	Probability that developer creates a project during any time slice after his first
probJoinEachPeriod	Probability that developer joins a project during any time slice after his first
probAbandonEachPeriod	Probability that developer abandons a project during any time slice after his first
developersPerPeriod	The number of developers introduced to the network each period
endTime	The duration of the simulation in time slices (days)

The agents' virtual network of collaborations is a database. The database is designed to catalogue all collaborations in the simulated network. It consists of three tables, "DEVELOPERS," "PROJECTS," and "LINKS." A record in each represents a developer, a project, or a developer's commitment to a project, respectively. Together, the primary keys from "DEVELOPERS" and "PROJECTS" form a composite primary key for entries in "LINKS." Thus, "LINKS," which archives every simulated collaboration, is designedly similar to our data source at SourceForge. Consequently, we can analyze simulation data with the same scripts used on the historic data. To date, no such analysis has occurred, but, once done, its results will direct parameter calibration and program revision.

Unlike developers, projects and links are not also Java agents, since they are passive elements of the social network. Projects and links are not self-motivated. They change only as developer decisions so dictate. Objectification of projects and links, however, is crucial to the flexibility of my design. As database objects, projects and links may gain attributes through table alterations adding fields. Based on polling of actual OSS developers, we hope to discover very specific factors that affect their collaboration choices. Should we uncover characteristics like project popularity as attractors for developer contribution, we can update the database schema to include them.

An open source modeling package, Swarm [Swarm], provided my simulation engine. Swarm instantiates developer instances, storing them in a list. The Swarm activity scheduler then traverses that list, prompting each developer to choose a daily activity. Swarm automates model iteration, removing cumbersome looping structures from the program code.

My simulation is thus a Java/JDBC/Swarm hybrid, utilizing the capabilities of all three.

Simulation Revision

Before the simulation approaches an accurate portrayal the OSS movement, revision is needed. Some aspects of the program code are too simple.

The current model simulates random attachment. Developer agents join a randomly selected project indiscriminately. However, power laws in the distribution of project degree suggest developers may employ preferential attachment, that developers identify

projects with certain qualities as more attractive. In aggregate, their choices begin to favor, or prefer, projects with the given qualities, skewing project membership to fit a power law [Schroeder, 1991]. The next simulation version should test the power law-preferential attachment correlation.

According to the present scheme, developer age determines activation order for each period. Developers join a list at birth. The current Swarm code traverses that list from start to finish, prompting each developer to act in order. The eldest developers, those nearest the top of the list, enjoy privileged activation. They always act first. No set of real developers maintains such a birthright. OSS developers create, join, and abandon projects at will, without uniform scheduling. The simulation's activation regime must change to accommodate that freedom. [Axtell, 2002] discusses possible approaches to doing so.

References

- Axtell, R. "Effects of Interaction Topology and Activation Regime in Several Multi-Agent Systems," Center on Social and Economic Dynamics, Working Paper 12, (July, 2002), 2002.
- Barabasi, A.L., Albert R. "Emergence of Scaling in Random Networks," *Science* (286), 1999, pp. 509-512.
- Barabasi, A.L. *Linked: The New Science of Networks*, Perseus Publishing May, 2002.
- Madey G., Freeh, V., Tynan, R "The Open Source Software Development Phenomenon: An Analysis Based on Social Network Theory," University of Notre Dame, 2002, pg. 2.
- Netcraft "Netcraft.com Web Server survey." (2002:Feb. 16, 2002), 2002.
- O'Reilly, T "Lessons from Open-Source Software Development," *Communications of the ACM* (42:4), 1999, pp. 44-45.
- Schroeder, M.R. *Fractals, Chaos, Power Laws*, W. H. Freeman and Company, New York, 1991.
- SourceForge.net, "SourceForge.net Statistics" <http://www.sourceforge.net>, July, 2002.
- Swarm Development Group, <http://www.swarm.org>, Santa Fe Institute, May, 2002.
- Terna, P "Simulation Tools for Social Scientists: Building Agent Based Models with SWARM," *Journal of Artificial Societies and Social Simulation* (1:2), <http://www.soc.surrey.ac.uk/JASSS/1/2/4.html>, March, 1998, pg. 2.
- Tjaden B. "The Kevin Bacon Game." (2001:July), 1996.
- Wasserman, S., K Faust *Social Network Analysis: Method and Applications*, Cambridge University Press, Cambridge UK, 1999.
- Watts, D. *Small Worlds: The Dynamics of Networks between Order and Randomness*, Princeton University Press, 1999.
- Wu, M.W., Lin, Y.D. "Open Source Development: An Overview," *IEEE Computer*, 2001, pp. 33-38.