Stability and Efficiency of Communication Networks in Open Source Software Development

Param Vir Singh

Yong Tan

Dept. of Management Science

School of Business

Box 353200

University of Washington

Seattle, WA 98195-3200

Abstract

Open source software (OSS) development teams use informal communication to coordinate their work towards common goals. According to software engineering folklore, the architecture and the organization of the final product depend on the communication patterns of the contributors. Unlike in a formal organization, the communication network structures in an OSS project evolve unrestricted and unplanned. Little is known about the stability and efficiency of the communication structures that would evolve in OSS projects. In this paper, we use the connections model of the social networks theory by incorporating the salient features of OSS development to study the communications structures that might emerge in OSS projects. We characterize the stable and efficient structures. We find that for a given scenario there may exist several stable structures which are inefficient. We also find that there does not always exist a stable structure that is efficient. This can be explained by the fact that the stability of the structure is dependent on individual's maximization of self utility whereas the efficiency of the structure is dependent on maximization of group utility. In general, tension exists between stable and efficient structures because the players act in their self-interest rather than the group-interest. We discuss the results of the model in the context of OSS development. We also provide numerical simulation to illustrate the tension between stable and efficient networks. We further discuss implications of our results and provide directions for future research.

Keywords: Open source software development, Social networks, Efficiency, Stability, communication structure, coordination structure.

1. Introduction

Software development is an unstructured and non routine task (Kraut et al. 1995) and typically requires informal communication for effective coordination (Ahuja et al. 1999; Van de Ven et al. 1976). According to Conway's law, the architecture and the organization of software depends on the communication needs of the contributors(Weber 2004).

In traditional organizations there exist formal structured communication networks (DiMaggio et al. 1983), which can be changed in the advent of changes in technology or environment. However in open source software development these communication structures are informal and evolve unplanned. Developers are free to organize themselves in any network structure they prefer. There is no formal mechanism by which a specific structure can be imposed on the developers. This raises several important questions: (1)What kind of communication structures would emerge in an open source software development project? (2) Would these structures be efficient? (3) Are there any mechanisms by which efficient structures be achieved in this informal environment?

Metaphors like "*bazaar*", "*clique*" and "*town council*" are identified with open source software development team structure. These structures represent varying patterns of communication among developers and have been observed in various open source projects. During early days of development of Linux8086, the communication structure represented a "clique" of core developers because of huge quality difference between "core" and "average" developers (Cox 1998; Crowston et al. 2004). "Town Councils" structures have been associated with failed open source projects because of their ineffectual carping (Cox 1998). Another school of thought based on "given enough eyeballs all bugs are shallow" suggests that the bazaar structure takes benefit of the law of large numbers and should be the preferred structure in open source projects (Weber 2004). "Cathedral" team structures have been associated with software

development in formal organizations (Krishnamurthy 2002; Raymond 1998). Using Social Network Analysis (SNA) visualization techniques, (Crowston et al. 2004) show that the communication structures are decentralized for large teams and centralized for small teams. (Raymond 1998) finds that with growth of the code the team structure shifts its preference from "cathedral" to "bazaar". These observations have been made at different stages of development process and on projects of varying complexities. However it is not clear as to under what scenario would any of these structures be observed? Are these structures efficient for the scenarios for which they emerge? We try to address these questions in this paper.

In this research we develop an analytical framework of communication network for an open source development project where self interested individuals can form or sever links. We show that several kinds of structures are possible in open source software development under different conditions. We are primarily interested in studying the stable and efficient communication structures. It is shown that under certain conditions the efficient structures are not necessarily stable. Moreover there exist several stable structures under certain conditions.

2. Literature Review

This work draws from both the open source software literature and connections model of social network theory. OSS researchers have focused primarily on the motivations of a contributor to an open source project. These motivations have been attributed to intellectual curiosity (Group 2003), labor economics (Hann et al. 2004; Lerner et al. 2002), needs to improve one's own specific programming skills (Lakhani et al. 2003), and promises of higher future earnings (Haruvy et al. 2003).

Recently researchers have used social network theories to investigate the open source phenomenon both analytically and empirically. This stream of research has focused primarily on

how the developers contribute or communicate in an open source software project. The positions and relationships among players in a social network play a very important role in the efficiency of the network. (Dalle et al. 2005) study the social mechanisms by which individual software developers' efforts are allocated within large and complex open source projects. Using this allocation mechanism they provide an analytical model to simulate the growth of an open source product. Social network analysis (SNA) techniques also provide tools that allow inquiries into the patterns of interaction empirically. In an investigation of 120 Free/Libre Open Source Software (FLOSS) projects (Crowston et al. 2004) find that many of the supposed strengths of the FLOSS development are closely related to the communications structure. The communication structures tend to be centralized for small projects and decentralized for large projects. (Raymond 1998) anatomizes a successful open source project "fetchmail" and contends that with the growth of a project the social structure of the contributors shifts its preference from cathedral to bazaar.

In general social network theories have been utilized to study the formation of network structures and their impact on performance of the players and the whole network. Some of the applications of these network theories range from communication among consumers (Wellman et al. 1988), exchange of employment opportunities (Montgomery 1991), social connections among individuals and collaboration among researchers (Jackson et al. 1996), system compatibility (Katz et al. 1994), information transmission, and internal organization of firms.

We use the connections model of the social network theories to study the questions at hand. Connections model has been applied successfully to study patterns, flow and effect of communication in social network of individuals. Connections model was formally proposed by (Jackson et al. 1996). It is based on graph theory and models the social communications among

individuals. In the graph, the nodes represent the agents and an arc exists between the two nodes if the corresponding agents interact bilaterally. Each agent possesses some information that has some value to other members of the graph. Individuals directly communicate with the ones they are linked and also benefit from indirect communication with the ones their adjacent nodes are linked. Direct communication is costly and the value of communication from the other nodes depends on the distance to those nodes. Each agent undergoes a cost-benefit analysis before indulging in link formation. This setting is then used to study stability, efficiency, and reliability of network structures. Several variations of connections models exist. In an earlier version of connections model, (Goyal 1993) considers that individuals can form links unilaterally. (Galeotti et al. 2005) introduce heterogeneity in individuals based on their communication and social skills. They contend that individuals can often be classified into groups and communication within a group is cheaper as compared to across group. They derive the conditions under which center-sponsored and periphery-sponsored stars are efficient.

In the context of open source software development, the communication could be about code development, request for new features, bug reports, patch management, documentation, licensing, and etc. The heterogeneity in agents would amount to having skill sets of varying levels and context. This implies that in addition to heterogeneity in cost of link formation, there is also heterogeneity in individuals' intrinsic values. Using the connection model and the underlying ideas of open source software development process, we develop a stylish model of communications in OSS. We show how the observed network structures of OSS development are possible in various scenarios.

This paper proceeds as follows. In section 3 we provide the definitions comprising the general model. In section 4 we describe efficient networks and the networks that are stable. In

section 5 we show a numerical example of a network consisting of two high-quality developers and three low-quality ones. In addition, we demonstrate the incompatibilities between efficient and stable networks.

3. Model

We consider an open source software development team. The team members are heterogeneous not only in the information they possess but also in the value of information. The heterogeneity in the value of information is a concept ingrained in open source software development processes. Prior studies (Cox 1998) have revealed that the difference between a good and average developer is 30 to 1. Here forth, the "good" and "average" developers would be referred as "high" and "low" respectively. The information possessed by "high" type players is valued higher than that by "low" type players. For instance, the high type players are generally the ones who have higher skills set and experience in the development process of the project under consideration. These high type developers usually contribute to the designing and core code development. Hence it would be rational to assume that they possess information more relevant to the project. This combined with their greater skill set in the context of the project at hand would imply that in any information exchange among contributors in their project they can contribute more than the low types. It must be noted that the low types contribute less in information only in comparison to the high types. In absolute terms the value provided by the low types can be quit high. For instance, in comparison to the value provided by a code developer's information, the value provided by the bug reporters' information is marginal. But the bug reporter's information alone is quite significant. A player can augment his information by interacting with other players with formed links. This interaction involves time and effort which can also be denoted by cost. "High" players are assumed to incur costs at least as high as

incurred by "low" players. Since most of the OSS developers participate in any project only part-time (Weber 2004) and hence they might like to spend more time on code development than helping lesser skilled developers via communication (Cox 1998). Also, in general, most of the code development and designing of the software is done by the high types (Krishnamurthy 2002; Raymond 1998) hence the high types may have less time for participating in communication. A Player derives full value from the ones he is directly linked and discounted value from the ones he is indirectly linked (i.e. linked through his adjacent nodes). Each player is rational and weighs the value obtained from forming a link against its cost. Now we provide the definitions which are consistent with previous literature and allow us to characterize stable and efficient networks.

3.1. Definitions

Let $H = \{1, \ldots, n_h\}$ and $L = \{1, \ldots, n_l\}$ be the sets of players having "high" and "low" information value levels respectively. Let $N = \{H, L\}$ represent the finite set of players. The network relation between the players is represented by a graph whose nodes represent the players and the arcs represent the pairwise relations.

Graphs. The complete graph denoted as g^N is a set of all subsets N of size 2. The set of all possible graphs on N is then $\{g \mid g \subset g^N\}$. The link ij is a subset of N containing players i and j. The graphs obtained by adding and severing a link ij to g are denoted by $g + (ij)$ and $g - (ij)$ respectively.

Values. The value function V of the graph is represented as $V : \{g \mid g \subset g^N\} \to R$. We consider the total value to be the aggregate of individual utilities $V(g) = \sum_i u_i(g)$, where $u_i : \{g \mid g \subset g^N\} \to R$.

Efficiency. A graph $g \subset g^N$ is strongly efficient if there exists no graph $g' \subset g^N$ such that $V(g) < V(g')$.

Allocation Rule. The allocation rule $Y : \{g : g \subset g^N\} \times V \to R^N$ in the graph describes how the value generated in the graph is distributed among players. $Y_i(g,V)$ is the value received by player *i* from graph *g* under *V*. The utility of player *i* from a graph *g* under the allocation rule $Y_i(g,V)$ is then given by

$$u_i(g) = \sum_{j \neq i} \delta^{t_{ij}-1} w_{ij} - \sum_{i:ij \in g} c_{ij}$$

where t_{ij} is the number of links in the shortest path between *i* and *j*, and δ is the discount rate.

$$c_{ij} = \begin{cases} c_h & if \quad i \in H \\ c_l & if \quad i \in L \end{cases}, \; w_{ij}(g) \;=\; \begin{cases} v_h, & \text{if} \quad i \in H \quad j \in H; \\ v_l, & \text{if} \quad i \in H \quad j \in L; \\ v_h, & \text{if} \quad i \in L \quad j \in H; \\ v_l, & \text{if} \quad i \in L \quad j \in L, \end{cases} \text{ and } \begin{matrix} v_h \geq v_l \\ c_h \geq c_l \end{matrix}$$

where $\{H, L\} \subset N$ and $ij \subset g$. Here w_{ij} is the intrinsic value of individual *j* to individual *i*, and c_{ij} is the cost of maintaining the link *ij* for *i*. Note that only direct links are costly. Both players party to a link incur a cost of maintaining the link.

Stability. We are primarily concerned about the relationship between stable and efficient network structures. We define a graph *g* to be pairwise *stable* with respect to *V* and *Y* if

(i) for all $ij \in g$, $Y_i(g,V) \geq Y_i(g-ij,V)$, and $Y_j(g,V) \geq Y_j(g-ij,V)$;

(ii) for all $ij \in g$, $Y_i(g,V) \geq Y_i(g+ij,V)$, and $Y_j(g,V) \geq Y_j(g+ij,V)$.

The formation of a link requires the consent of both parties, but the severance can be done unilaterally.

In the following, we consider a special case where there is only one high type player and n_l $_{>3}$ low type players. This allows us to obtain analytical solutions and to show the ranges of intrinsic value and cost where different types of structures are efficient.

3.2. Efficient or Stable Networks

Proposition 1. *For* $|H|=1$ *and* $|L|=n_l$, *the unique efficient structure is:*

(1) a completely connected network if

$$v_l > \frac{c_l}{1-\delta} \text{ and } v_l + v_h > \frac{c_l + c_h}{1-\delta};$$

(2) all low type nodes completely connected and one of these low type nodes connected to the high type node if

$$v_l > \frac{c_l}{1-\delta} \text{ and } \frac{c_h + c_l}{1-\delta} > v_h + v_l > \frac{c_h + c_l}{1+(n_l-1)\delta};$$

(3) a completely connected network of only the low type nodes if

$$v_l > \frac{c_l}{1-\delta} \text{ and } v_h + v_l < \frac{c_h + c_l}{1+(n_l-1)\delta};$$

(4) a star structure encompassing everyone with a high node at center if

$$v_l < \frac{c_l}{1-\delta},\ v_h - v_l > \frac{c_h - c_l}{1-\delta}, \text{ and } v_h + v_l + (n_l-1)\delta v_l > c_h + c_l;$$

(5) a star structure encompassing everyone with a low node at center if

$$v_l < \frac{c_l}{1-\delta},\ v_h + v_l < \frac{c_h + c_l}{1+(n_l-1)\delta},\ v_h - v_l < \frac{c_h - c_l}{1-\delta}, \text{ and}$$

$$v_h + (2n_l-1)v_l + (n_l-1)^2\delta v_l + (n_l-1)\delta v_h > c_h + (2n_l-1)c_l;$$

(6) a star structure encompassing only the low type nodes if

$$v_l < \frac{c_l}{1-\delta},\ v_h + v_l < \frac{c_h + c_l}{1+(n_l-1)\delta}, \text{ and } v_l > \frac{2c_l}{2+\delta(n_l-2)};$$

(7) no links if

$$v_l < \frac{2c_l}{2+\delta(n_l-2)},\ v_h + v_l + (n_l-1)\delta v_l < c_h + c_l, \text{ and}$$

$$v_h + (2n_l-1)v_l + (n_l-1)^2\delta v_l + (n_l-1)\delta v_h < c_h + (2n_l-1)c_l.$$

Figure 1 provides a visualization of the network structures (with n_l = 3) stated in Propositions 1 and 2. The blue square represents the high type developer, while the low type ones are depicted by red circles.

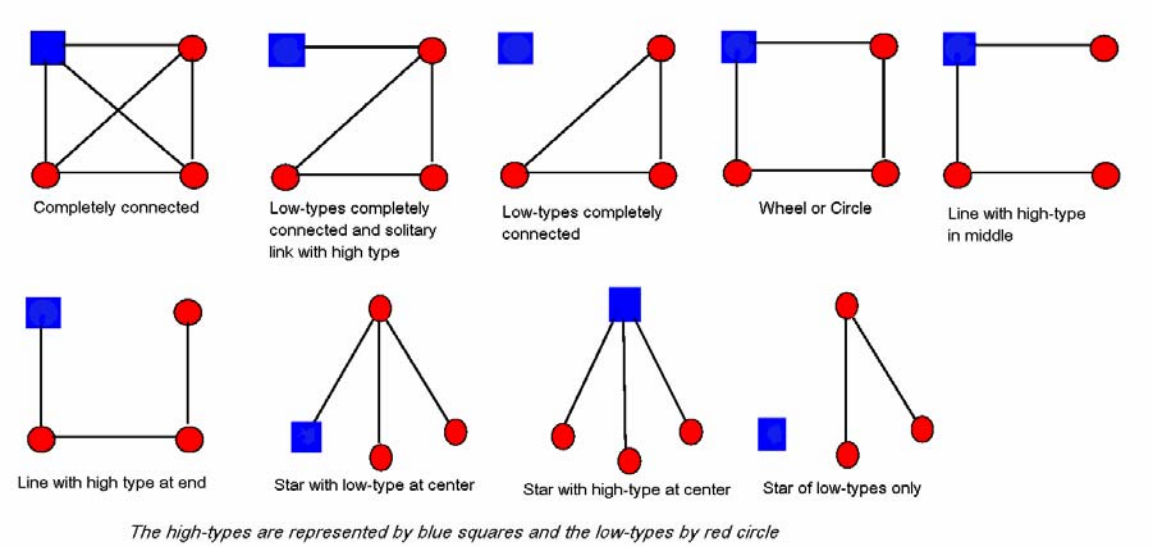


Figure 1. Network structures

Proposition 2. *For* $|H|=1$ *and* $|L|=n_l$,

(1) a pairwise stable network has at most one non-empty components;

(2) a completely connected network is the only stable network if

$$v_l > \frac{c_h}{1-\delta};$$

(3) all low type nodes completely connected and one of these low type nodes connected to the high type node is the only stable network if

$$\frac{c_h}{1-\delta} > v_l > \max\left[\frac{c_l}{1-\delta}, \frac{c_h}{1+(n_l-1)\delta}\right];$$

(4) a completely connected network of only the low type nodes is stable if

$$\frac{c_h}{1+(n_l-1)\delta} > v_l > \frac{c_l}{1-\delta};$$

(5) a star structure encompassing everyone with a high type node at center is stable if

$$\frac{c_l}{1-\delta} > v_l > c_h;$$

(6) a star structure encompassing everyone with a low type node at center is stable if

$$\frac{c_l}{1-\delta} > v_l > \max\left[c_l, \frac{c_h}{1+(n_l-1)\delta}\right];$$

(7) a star structure encompassing only the low type nodes is stable if

$$\min\left[\frac{c_l}{1-\delta}, \frac{c_h}{1+(n_l-1)\delta}\right] > v_l > c_l;$$

(8) for $v_l < c_l$, any pairwise stable network that is non empty is such that each player has at-least two links and hence inefficient.

Note that Proposition 2 does not necessarily restrict the stable structure to be *unique* in the corresponding range. For example, for the range specified in Point 7, if n_l = 2, a line encompassing everyone with high type node at the end is stable if

$$c_l < v_l < \min\left[\frac{c_l}{1-\delta}, \frac{c_h}{1-\delta^2}\right];$$

a line encompassing everyone with low type nodes at both ends is stable if

$$c_h < v_l < \frac{c_l}{1-\delta^2};$$

and a circle encompassing everyone is stable if

$$\frac{c_l}{1-\delta} < v_l < \frac{c_h}{1-\delta^2}.$$

3.3. Discussions

As is obvious from the above propositions, the efficient or stable networks are generally completely connected (i.e. decentralized) or star shaped (i.e. centralized). Other cases are minor variations of these two types of structures. The completely connected network is observed when the low type nodes make high value contributions, as compared to the cost of link formation for high type nodes. This might be the case where qualified developers get together on a project and the high type is some one who has already worked on some similar projects. The high type feels the discussions with low types to be useful, enriching, and productive to the development of the project. In this scenario, the low type does not really mean someone with very low skill set in an absolute sense, but rather a low skill set relative to that of the high type in the context of the project at hand. This kind of structure may be observed during initial development phase. This

structure is a form of bazaar structure where everybody communicates with each other. Note that for this structure the intrinsic value of the low type player needs to be quite high.

The completely connected low node network with solitary link with high node is observed in the case where the low types individually do not provide much useful information, but as a group can provide reasonable information. Moreover here the low types can help each other more by answering questions or queries. This might be the case where a high type is a very advanced developer and the low types are beginners with information about enhancing the usage of the product but little information about the minute details of code development. These low types can discuss about the type of new features that might be useful, reach a consensus, or develop small patches and provide the information to the high type node. The high type can then incorporate their suggestions into the code. This might be observed in scenarios where code development and bug reporting goes on simultaneously. The high type might be the only one who has the skill set to understand and develop the code whereas the low types may be the ones who can contribute marginally by reporting bugs or by developing small patches.

The completely connected network of the low type nodes is observed in the case where even the low type nodes all together as a group are not able to provide information that can outweigh the cost of link formation for the high type node. However as in previous case, the low type nodes are competent enough to help each other. In the case where high type node is the main developer, none of the suggestions of the low type nodes will be incorporated in the code. The implications of this structure can not be appreciated in this setting (with just one high type node). We will show that this structure is equivalent to a “clique” of high type nodes in the next section. The low type players might represent something like a town council and the high type is better off not communicating with them.

Star structures would generally be observed during bug reporting and fixing stages of the project. Information about bugs is more valuable to the bug fixers or code developers than to other bug reporters. Hence the periphery will consist of bug reporters or fixers and the center will have someone who can organize and assign tasks to them. The star network with high type node at the center might be observed in the case where the high type node is the one which has most information about the project and the individual information provided by the low type nodes outweighs his cost of link formation. This kind of structure might be efficient during bug reporting or patch assignment phase of the project. The individual low type nodes can report bugs to the central node (high type) and the high type can decide which task to assign to which node and help them in developing the patch.

The star with low type node at the center is efficient in the case where the high type node's cost of link formation is worth more than the individual intrinsic value of low type nodes. This might amount to a high type node deputing a low type node to communicate with other low type nodes. This might also be observed in bug reporting and fixing. The star comprising of low type nodes only can be observed in the case where the high type developer's link formation cost is too high. The high type does not communicate with the low types. This may have interesting implications on the architecture of the software which will depend on which segment (high or low) contributes to the code.

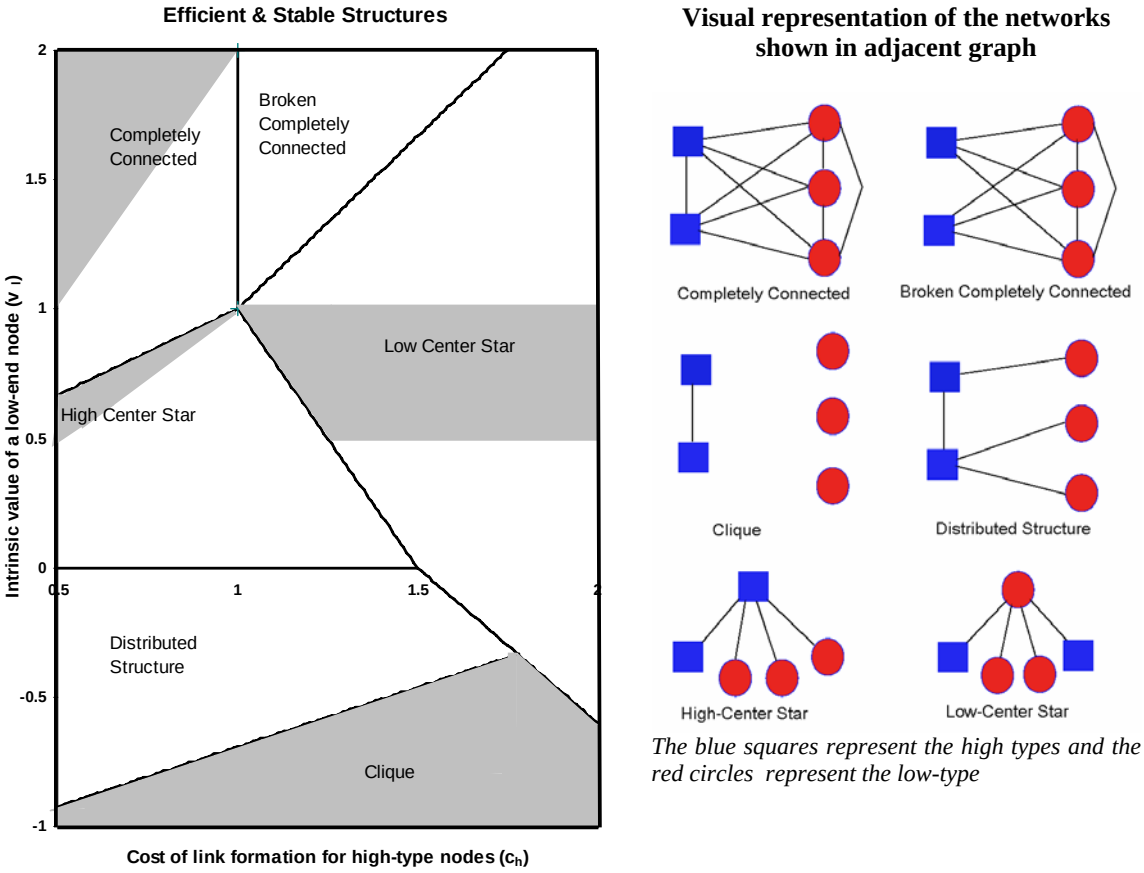
It must be noted that same structure might be observed across different stages of a project. It would depend on the intrinsic values and costs of the two types at different stages. In the discussion we have just provided few examples where these structures might be observed. Moreover, the communication structure of projects which have fewer numbers of developers is more likely to see a high type contributing more to the communication. However, as the size of

the development team increases the heterogeneity among the value levels of the team members also increases and there is greater possibility of having communication structures with marginal contribution from the high type nodes.

One of our main aims of this paper is to illustrate the incompatibility between stable and efficient networks. At the end of proposition 2 we provided some examples of structures that are stable and inefficient. However at this point it may not be clear how tension exists between stable and efficient networks. In the next section, we provide results of numerical simulation for a case with two high type nodes and three low type nodes to illustrate the incompatibility between stable and efficient structures.

4. Numerical Simulation

Based on above definitions, we conduct numerical simulations for a team of 5 developers of which 2 are high type and 3 are low types. Figure 2 depicts the efficient and stable structures in the value and cost domain. We have fixed $v_h = 2$, $c_l = 0.5$, $\delta = 0.5$, and varied c_h from 0.5 to 2 and v_l from -1 to 2. The whole area is divided into several parts bounded by solid black lines. In each area inside black lines the name of the structure represents the efficient structure in that area. The shaded region in each area shows the region where the efficient network is also stable in that area.



The blue squares represent the high types and the red circles represent the low-type

Figure 2. Efficient and Stable structures in the value and cost domain

Here several interesting observations can be made. First, it demonstrates the tension between the efficient and stable networks. The structures which are efficient are not necessarily stable in the complete range for which they are efficient, i.e. unstable structures can be efficient in certain ranges. For instance, the broken completely connected network is efficient in certain range but it is never stable. This means that the communication structure that evolves in an open source software development team will never stabilize to a broken-completely-connected structure. Also note that most of the efficient structures are stable only in a very limited range.

The second point to consider is the stability and efficiency of the clique structure. A clique would be observed when the low type players create lot of noise (as v_l is negative in this range). In this scenario, the high type developers would like to wall themselves off the low types. This

occurs for projects of high complexity. As mentioned earlier, this structure was observed among developers during the early days of development of Linux.

There is a third interesting observation. As the low type developers' value contribution increases, it is better for the network to have a low type developer at the center of a star structure. This would correspond to a situation where the high type developer deputizes a low type developer to communicate with all other. Note that the low types are not really novices here but have skill sets only marginally smaller than that of high types. This can also be observed in scenarios where core code development and patch management goes on simultaneously.

We now show the tension between stable and efficient structures. Figure 3 shows the completely connected structure (a) and the network with one less link (b). The structure in (a) is efficient for the values of $v_h = 2$, $c_l = 0.5$, $\delta = 0.5$, $c_h = 0.75$ and $v_l = 1.45$. The structure in (b) is used to show why the structure in (a) is unstable.

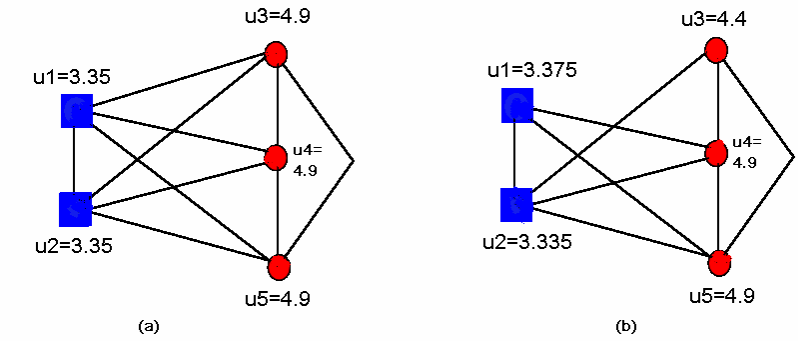


Figure 3. Tension between stable and efficient networks

The value of the utility for each node is presented adjacent to it. The total value generated by the structure in (a) is 21.4 as compared to 20.975, the total value generated by the structure in (b). As is obvious from the previous graph, structure (a) is efficient. However it is not stable. Compare the utilities of node 1 (u1) in both the structures. It is higher in structure (b). Therefore in structure (a), node one can increase its utility by severing the link with node 3. This implies

that structure (a) is not pairwise stable for the given parameter values. This situation arises because node 1 is concerned about self utility rather than the group utility; and the group can not enforce it to not to sever the link.

5. Conclusion

In this section we discuss several implications for the OSS development team administrators and research directions for academicians. One of the main implications is that in addition to managing the codes submitted by the developers it is important, for the administrator, to monitor and manage the communication among developers. It is generally the high-type developers who communicate less than what is needed for efficiency. Appropriate incentive mechanisms should be designed to lure the high-type developers to contribute more to communication. Equal bargaining rule allows the benefits of a connection to be split equally between the two nodes(Jackson et al. 1996) and induces the players who contribute less to contribute efficiently. A similar rule/mechanism can be developed to ensure that the high types get reasonable value when communicating with low types. Since most of the developers are motivated by reputations among the developer community. A ranking mechanism could be developed for the contribution by communication. However one should be careful while designing this ranking system because it could invite a lot of cheap talk also.

For the sustainability of the project, it is also important to retain the individuals who are at the center of the information exchange networks. This is not an easy thing to do because the developers are free to come and go. However the communication in OSS teams takes place online and the information exchange is preserved in the email lists. Hence with some training any individual is replaceable at least in communication network. A list of important communications that took place can also be maintained for future references.

The structure of the software depends on the communication structure of the developers. The structure of the software should be influenced more by the high-type developer than a low-type developer. Hence in centralized structures it is important to have a high type developer at the center.

Finally we conclude by pointing out our main results and future research directions. We showed that several kinds of network structures would emerge in OSS communication teams. In case of heterogeneous players the over all value generated by these networks depend not only on the architecture but also on the relative positioning of players. Since players are motivated by self-interest rather than group-interest, the structures that are efficient are not stable for certain ranges of variables. We also show that for a given set of variables there exist several structures that are stable but only one which is efficient.

Each type of communication network architecture has implications for the coordination, negotiation, and growth of individuals and social capital. The implications of the network structures observed in this study on the final output of an OSS project can also be studied. Certain incentive mechanisms can also be designed for releasing the tension between stable and efficient structures. Whether the OSS phenomenon evolves such mechanisms on its own remains an open question and can be tested empirically.

References

1. Ahuja, M.K., and Carley, K.M. "Network Structure in Virtual Organizations," *Organization Science* (10) 1999, pp 741-757.
2. Cox, A. "Cathedrals, Bazaars, and the Town Councils," Slashdot.org, 1998, pp. 1-4.
3. Crowston, K., and Howison, J. "The Social Structure of Free and Open Source Software Development," School of Information Studies, Syracuse University, 2004.
4. Dalle, J.-M., and David, P.A. "Simulating Code Growth in Libre (Open-Source) Mode," University Pierre-et-Marie-Curie & IMRI-Dauphine and Stanford University & Oxford Internet Institute, 2005.

5. DiMaggio, P.J., and Powell, W.W. "The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields," *American Sociological Review* (48:2) 1983, pp 147-160.
6. Galeotti, A., and Goyal, S. "Network Formation with Heterogeneous Players," in: *Games & Econ Behavior*, 2005.
7. Goyal, S. "Sustainable Communications Model," Tinbergen Institute Discussion Paper, Rotterdam-Amsterdam.
8. Group, B.C. "Boston Consulting Group/OSDN Hacker Survey," Boston Consultancy Group, Boston.
9. Hann, I.-H., Roberts, J., Slaughter, S., and Fielding, R. "An Empirical Analysis of Economic Returns to Open Source Participation," Carnegie Mellon University, 2004.
10. Haruvy, E.E., Wu, F., and Chakravarty, S., "Incentives for Developers' Contributions and Product Performance Metrics in Open Source Development: An Empirical Investigation," University of Texas at Dallas, 2003.
11. Jackson, M.O., and Wolinsky, A., "A Strategic Model of Social and Economic Networks," *Journal of Economic Theory* (71) 1996, pp 44-74.
12. Katz, M., and Shapiro, C. "Systems Compatibility and Network Effects," *Journal of Economics Perspective* (8) 1994, pp 93-115.
13. Kraut, R., and Streeter, L. "Coordination in Software Development," *Communications of the ACM* (38:3) 1995, pp 69-81.
14. Krishnamurthy, S., "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects," *First Monday* (7:6) 2002.
15. Lakhani, K., and Hippel, E.V., "How Open Source Software Works: 'Free' User-to-User Assistance," *Research Policy* (32) 2003, pp 923-943.
16. Lerner, J., and Tirole, J., "Some Simple Economics of Open Source," *Journal of Industrial Economics* (52) 2002, pp 197-234.
17. Montgomery, J., "Social Networks and Labor Market Economics," *American Economic Review* (81) 1991, pp 1408-1418.
18. Raymond, E.S., "The Cathedral and the Bazaar," *First Monday* (3:3), March 3 1998.
19. Van de Ven, A.H., Delbecq, D., and Koening, R.J., "Determinants of Coordination Modes within Organizations," *American Sociological Review* (41:2) 1976, pp 322-338.
20. Weber, S., "*The Success of Open Source,"* Harvard University Press, Cambridge, Massachusetts, and London, England, 2004.
21. Wellman, B., and Berkowitz, S., "*Social Structure: A Network Approach,"* Cambridge university Press, Cambridge, 1988.

Appendix

A1. Proof of Proposition 1

Points (1), (2), and (3)

For $v_l > \frac{c_l}{1-\delta}$ any two indirectly connected nodes belonging to *L* can increase their utilities as well as the overall value by forming a direct link. Hence for $v_l > \frac{c_l}{1-\delta}$ the efficient structure must have completely connected low type nodes. This completely connected low type node

structure can form a single link to the high type node. The value change of the graph in this case is $v_h + v_l + (n_l - 1)\delta v_l + (n_l - 1)\delta v_h - c_h - c_l$. This new graph will be efficient only if the value change is positive. This structure can further form more connections with the high type node. The value change for forming any new connection with the high type node is $(1-\delta)v_l + (1-\delta)v_h - c_h - c_l$ per new connection henceforth. Hence the completely connected structure will be efficient if this value change is positive.

Points (4), (5), and (6)

We will provide an intuitive proof for these parts. A more robust proof is eliminated due to space limitations. For $v_l < \frac{c_l}{1-\delta}$, direct connections are un-preferable and indirect connections are preferred. Indirect connections with minimum number of nodes in between are the most preferred because $v_l - c_l < v_l\delta \geq v_l\delta^2$. A star structure minimizes the number of direct connections but maximizes the number of indirect connections with only one intermediary node. Three different star structures are possible: (a) star with high type node at center (b) star with low type node at center (c) star with no high type node.

In case (a) the number of direct links is n_l and the number of indirect links is $n_l(n_l - 1)/2$. The value from each direct link is $v_h + v_l - c_l - c_h$ and from each indirect link is $2\delta v_l$. Hence the total value of the star in case (a) is $n_l(v_h + v_l - c_l - c_h) + n_l(n_l - 1)\delta v_l$

In case (b) the number of direct links involving high node is 1 and yields a value of $v_h + v_l - c_l - c_h$. The number of indirect links involving high node are $(n_l - 1)$ and yield a value of $\delta(v_l + v_h)(n_l - 1)$. The number of direct links involving only low nodes is $(n_l - 1)$ and yields a value of $(n_l - 1)(2v_l - 2c_l)$. The number of indirect links involving only low type nodes is $(n_l - 1)(n_l - 2)/2$ and yields a value of $2\delta v_l(n_l - 1)(n_l - 2)$. Hence the total value of the star in case (b) is $(v_h + v_l - c_l - c_h) + \delta(v_l + v_h)(n_l - 1) + (n_l - 1)(2v_l - 2c_l) + \delta v_l(n_l - 1)(n_l - 2)$.

In case (c), the number of direct links is $(n_l - 1)$ and yields a value of $(n_l - 1)(2v_l - 2c_l)$. The number of indirect links is $(n_l - 1)(n_l - 2)/2$ and yields a value of $2\delta v_l(n_l - 1)(n_l - 2)$. Hence the total value of the star in case (c) is $(n_l - 1)(2v_l - 2c_l)2\delta v_l(n_l - 1)(n_l - 2)$

A direct calculation using final values of the three stars can easily show that the value of star comprising of *m* + *n* individuals is greater than the value of two separate stars of *m* and *n* individuals respectively.

Start with (4). Case (a) is better than case (b) and case (c). Comparing total values of the three stars, case (a) is better than both case (b) and (c) if $v_h - v_l > \frac{c_h - c_l}{1-\delta}$; However it might happen that case (a) provides negative value in some range that satisfies $v_h - v_l > \frac{c_h - c_l}{1-\delta}$ as well as $v_l < \frac{c_l}{1-\delta}$. Hence it will be efficient till it satisfies both the above conditions and provides a non-negative value. To avoid non-negativity, we equate value of case (a) to zero and get $v_h + v_l + (n_l - 1)\delta v_l > c_h + c_l$.

For (5), case (b) is better than both case (a) and (c). Comparing total values of the three stars, case (b) is better than both case (a) and (c) if $v_h - v_l < \frac{c_h - c_l}{1-\delta}$ and $v_h + v_l < \frac{c_h + c_l}{1+(n_l-1)\delta}$ respectively. However it might happen that case (b) provides a negative value in some range that satisfies the above two conditions as well as $v_l < \frac{c_l}{1-\delta}$. Hence it will be efficient till it satisfies the three above conditions and provides non negative value. To avoid non-negativity, we equate value of case (b) to zero and get $v_h + (2n_l - 1)v_l + (n_l - 1)^2 \delta v_l + (n_l - 1)\delta v_h > c_h + (2n_l - 1)c_l$.

For (6), case (c) is better than both case (a) and (b) if $v_h + v_l < \frac{c_h + c_l}{1+(n_l-1)\delta}$ and is non-negative if $v_l > \frac{2c_l}{2+\delta(n_l-2)}$.

A2. Proof of Proposition 2

(1). Consider that *g* is a pairwise stable network and has two or more non empty components. (a) Let u^{ij} be the value to component $i \in L$ from link *ij*, given graph *g*. Then $u^{ij} = u_i(g) - u_i(g - ij)$ if $ij \in g$ and $u^{ij} = u_i(g + ij) - u_i(g)$ if $ij \notin g$. Let us assume $ij \in g$ then $u^{ij} \geq 0$. Consider a pair *kl*, $k \in L$ which belongs to another component. Then $u^{kj} > u^{ij} \geq 0$ because *k* also gets the discounted value of node *i*. Following this argument it can be easily seen that all the low type nodes are connected. This does not preclude the possibility that the high type node is not connected to the low type nodes. But the definition of component is such that it does not consider a node alone to be a component.

(2). This follows from the argument that all the nodes that are not directly connected can increase their utility by forming direct links.

(3). $v_l > \frac{c_l}{1-\delta}$ implies that the low type nodes are completely connected. $v_l > \frac{c_h}{1+(n-1)\delta}$ implies that the high type node can get a non negative utility by forming a single link to the completely connected low type nodes. However, $v_l < \frac{c_h}{1-\delta}$ implies that the high type node prefers indirect connections to the direct ones and hence will form only one direct connection.

(4). In the range where $\frac{c_h}{1+(n-1)\delta} > v_l > \frac{c_l}{1-\delta}$, we can easily see that the high type node will get a negative utility by forming a link to the completely connected low type nodes.

(5), (6), (7). It is easy to verify that these structures are stable. The low type nodes prefer indirect connections for $v_l < \frac{c_l}{1-\delta}$. The different types of stars are formed only because of the high type nodes cost.

(8). $v_l < c_l$ precludes any loose ends in the structure and hence any form of star is not stable. Hence by Proposition 1 the structure is inefficient.