Location, Location, Location: How Structural Embeddedness Affects Project Success in Open Source Systems

By

Rajdeep Grewal
Gary L. Lilien
Girish Mallapragada
The Pennsylvania State University

ISBM Report 8-2004

Institute for the Study of Business Markets
The Pennsylvania State University
402 Business Administration Building
University Park, PA 16802-3004
(814) 863-2782 or (814) 863-0413 Fax
Location, Location, Location: How Structural Embeddedness Affects Project Success in Open Source Systems

Rajdeep Grewal
Gary L. Lilien
Girish Mallapragada

September 2004

Rajdeep Grewal is Assistant Professor of Marketing, Gary L. Lilien is Distinguished Research Professor of Management Science, and Girish Mallapragada is Doctoral Student in Marketing, all at the Smeal College of Business, the Pennsylvania State University, University Park, Pa. The authors contributed equally to this research and are listed alphabetically. The article benefited from the feedback of Bill Ross and Raji Srinivasan and the support of Penn State’s Institute for the Study of Business Markets (ISBM).
Location, Location, Location:  
How Structural Embeddedness Affects Project Success in Open Source Systems

ABSTRACT

With the community-based model for software development followed in open source environments becoming a viable alternative to traditional firm-based models, we examine the effects of structural embeddedness-- or the nature of the relationship among projects and developers-- on the success of open source projects. We find that considerable heterogeneity exists in the embeddedness of open source projects and developers. We use a visual representation of the affiliation network of projects and developers to demonstrate this heterogeneity and to develop insights about how these structures differ across projects and developers. Our main results surround the effect of this differential structural embeddedness on project success. We find that structural embeddedness has strong and significant effects on both technical and market success, but that those effects are quite complex. We use latent class regression analysis models to show that multiple regimes exist, and that some of the effects of structural embeddedness are positive under some regimes and negative in others. These findings show that different aspects of structural embeddedness have powerful but subtle effects on project success and suggest that this is a rich environment for further study.
1. INTRODUCTION

With IBM endorsing Linux as a viable operating system option and contributing its source code for
speech recognition and relational database software to various open source initiatives (e.g., Lohr 2004),
along with Microsoft explicitly recognizing its competitive rivalry with Linux (e.g., Spencer and Greene
2003), the community-based model for software development of the Open-Source movement has clear
legitimacy as a potent competitor of the traditional firm-based model for software development (e.g., von
Hippel 2001; von Hippel and von Krogh 2003). The primary emphasis of open source systems is on
developing software such that the source code is public. The level of success of the resulting new code
from open source software development projects will likely determine the survival likelihood of this
community-based movement (e.g., Lakhani and Wolf 2003; von Hippel and von Krogh 2003). The
legitimacy of this model of software development provides both an opportunity and a challenge. The
opportunity is that this self-generating, collaborative model may provide new templates that can enhance
the efficiency and effectiveness of the new product development process. The challenge is to see if (a)
there are sufficient differences in the types of collaborative structures that have thus far emerged to infer
which models work and which don’t and (b) to measure and quantify the relationship between these
structural differences and the success and failures of the associated development projects. We report
here on two studies that rely on data on multiple projects and developers collected from a consortium of
open source projects, specifically SourceForge.net, to address these two challenges.

The naturally-evolving structure of the relationships between the developers involved and the
projects that they are working on—the social capital involved in the system—provides a critical focus for
the distinction of the open source movement from more traditional development mechanisms.

Recognizing the criticality of social capital (e.g., Granovetter 1985; Lin, Cook, and Burt 2001; Portes
1998), organizational researchers have highlighted the importance of structural embeddedness—the
nature of the relationships-- in organizational activities such as receiving financing (e.g., Uzzi 1999),
distribution of power in interfirm relationships (Yamagishi, Gillmore, and Cook 1988), and hiring top
managers (Granovetter 1995). Building on this research in organizational sociology, we suggest that social capital and the ensuing structural embeddedness (e.g., Granovetter 1985; Uzzi 1996) is likely to influence the success of open source software development projects. Thus, our research first identifies the nature of structural embeddedness in open source systems and then relates this embeddedness to the success of open source projects.

We study a foundry (a related set of projects) and associated projects at SourceForge.net, comprising 108 projects and 490 developers, and find in Study 1 that the organizational structure—the structural embeddedness—differs significantly across these projects. In Study 2 we find that after controlling for more standard descriptors such as project age, the degree and nature of structural embeddedness of both projects and developers does indeed influence project success. The pattern of this influence is quite complex, however, in that greater embeddedness (centrality) is not always beneficial. In other words, we find that embeddedness leads to higher levels of technical and/or commercial success for some projects, while embeddedness for other projects leads to lower levels of technical and/or commercial success. These results help determine what organizational structures help and what structures harm open source projects.

We proceed as follows. In Section 2, we provide our conceptual background and research hypotheses. There we show why it is appropriate to view open source systems as networks, provide an overview of two-mode affiliation networks, discuss the relevant literature on centrality in social networks, and introduce our measures of project success. In Section 3 we outline our strategy for collecting data on 108 projects listed on SourceForge.net, rely on sociometrics to provide a visual representation of relationships among open source projects and developers, and present results (Study 1), from a latent class cluster analysis, to statistically establish heterogeneity in the structural embeddedness of projects and project managers. In Section 4, we relate structural embeddedness of projects and project managers to project success (Study 2). We conclude, in Section 5, by discussing our findings and providing directions for further research.
2. CONCEPTUAL BACKGROUND AND RESEARCH HYPOTHESES

We argue that social capital varies across projects and developers and social capital plays a critical role in the success of open source projects. We view social capital as the relations among developers (projects) that provide developers with access to information and perhaps embedded resources (e.g., Lin 2001; Portes 1998). The analysis of social capital focuses on what is referred to as the network effect (e.g., Ruef, Aldrich, and Carter 2003) or structural embeddedness (e.g., Granovetter 1985). The emphasis is this line of investigation is to examine the importance of developers’ (projects’) location, i.e., how central is that location (e.g., Portes 1998), and the strength of the ties that the location provides the developers (e.g., Granovetter 1973). Central locations with stronger ties increase structural embeddedness and social capital. We begin by justifying our use of social networks to study open source systems and then develop hypotheses for the effects of structural embeddedness on project success.

2.1. Open Source Systems as Networks

Software development in the community-based model of the open source movement involves collaboration among developers working in teams (for more details, see Appendix A). Often developers work on multiple software development projects and thus belong to multiple teams. The importance of teams in new product development is a well-researched area (e.g., Wind and Mahajan 1997), demonstrating the critical role of team leaders, the importance of team composition, and the criticality of team chemistry (e.g., Sarin and Mahajan 2001).

Hence, the structure of software development teams should be important in the open source environment. These software development teams are largely self organized, i.e., the hierarchical structure that exists within firms does not directly manifest in the community based model (e.g., Lakhani and Wolf 2003). Social capital is likely to substitute for the positional power that comes from hierarchical structure within firms. Specifically, developers with social capital should find it easier to put together teams with requisite skills sets and the projects initiated by these more central developers should be more sought after because of the gain in reputation from social capital (e.g., Ruef et al. 2003). Thus, consistent with
research on structural embeddedness (e.g., Granovetter 1985), we use centrality measures to assess social capital and relate it to project success.

2.2. Two-Mode Affiliation Networks

To evaluate the presence of heterogeneity in structural embeddedness (measured as variation in centrality) and the consequences of this heterogeneity for project success, we rely on two-mode affiliation networks that are used to study the relationship between actors and events (e.g., Faust 1997) where, in our case, the actors are developers and the events are projects. Developers are related to one another because they work together on projects and projects are related to one another because they share developers.

To illustrate, consider six developers Adam, Bob, Chris, Jean, Joan, and Dave and three projects Deskpro, Screenpro, and Keypro shown in Figure 1. No two developers share a relationship with each other directly and no two projects are linked to each other directly. However, developers share an indirect relationship through the common projects that they work on: Adam, Chris, and Jean are developers for Deskpro and thus are related to each other. Similarly the two projects share an indirect relationship through the developers that work on both the projects. Thus, Deskpro and Screenpro have one common developer, i.e., Adam. Adam is also the most central developer in that he has ties with three developers (with Chris and Jean due to Deskpro and with Bob due to Screenpro). Bob, a developer who works with Adam on Screenpro, is linked to Chris and Jean because of his relationship with Adam, who, in turn, shares relationships with Chris and Jean (developers of Deskpro). The affiliation graph given in Figure 1 is not fully connected, i.e., we cannot move from a project to all the other projects or from a developer to all the other developers. From this graph one can develop the affiliation matrix shown on the upper right hand side of Figure 1, where a 1 indicates that a developer works on the project and the number 0 indicates that he does not. We use this structure in the analysis that follows.
2.3. Centrality and Structural Embeddedness

We use the approach outlined above to develop measures for structural embeddedness of projects and developers. To capture the embeddedness of projects and developers, we use the notion of centrality that captures the “importance” or “visibility” of projects and developers (e.g., Faust 1997; Freeman 1979). As our unit of analysis is a project and a project can have many developers, we use the project manager’s embeddedness to operationalize actor centrality.

The literature on centrality considers three elements of centrality (e.g., Faust 1997) i.e., project managers are central if they (1) are connected to several projects – *degree centrality* (i.e., the number of projects in which the manager participates), (2) can potentially mediate the flow of information and/or resources among other developers – *betweenness centrality* (i.e., the number of paths between other nodes on which the manager lies), and (3) have ties with other central projects – *eigenvector centrality* (i.e., the manager participates in important projects). In a similar manner one can define centrality for projects. A manager with high degree centrality works on projects with many developers, and projects with high degree centrality have many developers. A manager with high betweenness centrality serves as a gatekeeper and other developers rely on this manager for technical and non-technical information. Managers with high eigenvector centrality have ties to projects, which occupy a central location. Therefore, we examine the consequences for project success for the degree, betweenness, and eigenvector centrality of actors and events (e.g., Freeman 1979). In Appendix B, we present the operationalization of these three embeddedness constructs and illustrate this operationalization using the stylized example presented in Figure 1. To relate structural embeddedness to project success, we must first define what we mean by project success, which we address next.

---

1 The literature also discusses closeness centrality, only defined for fully connected graphs, which is not relevant with our data.
As open source projects are devoid of direct profit motives (e.g., Lakhani and Wolf 2003), defining criteria for success of open source projects is not evident. Nonetheless the criteria for success of open source projects should encompass both the technical achievements of a project, which depends on the scope and the complexity of the projects and the technical skill with which the projects are developed, as well as an indicator of market or commercial success. This pair of criteria for project success is consistent with the literature in information systems on software success (e.g., Rai, Lang, and Welker 2002) and literature on R&D success in new product development going back to Mansfield and Wagner (1976). Hence, we seek measures of open source project success that relate to (1) the technical know-how created in the project and (2) the commercial/economic success of the project.

Software development teams use the Concurrent Versioning System (CVS) to manage the software development process. CVS enables teams to store source code at a central location, thus enabling team members to retrieve the source code to make changes. CVS also helps the team to keep track of every change, including what was changed, when it was changed and who made the change, and helps in blending changes made by different developers, including making sure that developers do not accidentally overwrite each others’ alterations. A commit occurs when a developer uploads the altered source code file, where the CVS tool updates the changed files automatically. As CVS commits reflect meaningful changes to the source code, we treat the count of CVS commits as an indicator of successful technical refinement.

To assess commercial/economic success, we use the number of downloads over the life span of a project. Number of downloads is a market-based measure of popularity, which should relate to product use, particularly when software is distributed through a single channel as in the case of SourceForge (e.g., Crowston, Annabi, and Howison 2003). When a software product is freely available, researchers have used downloads as a surrogate for “sales” (e.g., the case of shareware in Chandrashekaran et al. 1999).
2.5. Research Hypotheses

We propose four hypotheses, i.e., two on the influence of project centrality on technical and commercial success and two on the influence of project manager centrality on technical and commercial success. Given the nascent stage of theory development in the open source area, we draw from several literatures including social networks, signaling, word of mouth influences, and firm reputation to develop the hypotheses.

2.5.1. Project Centrality and Technical Project Success

When project centrality is high, projects have access to greater resources due to the larger number of developers (degree centrality) and the better information quality due to developers’ linkages with other projects in general (betweenness centrality) and other important projects in particular (eigenvector centrality) (e.g., Freeman 1979). Thus, high degree centrality implies that the complex tasks associated with software development can be spread over more developers, resulting in better organization and hence higher productivity. The development process, which involves tasks such as code development, debugging, document writing, translation, and consulting can be better handled with greater resources and should lead to more technical success. Access to higher quality information should also increase the technical success of projects, as it tends to be more relevant, has greater accuracy and reliability and tends to be timely (e.g., O’Reilly 1982). Research in diverse contexts such as on stock returns (e.g., Veronesi 2000) and decision quality (e.g., Raghunathan 1999) shows that high quality information tends to be used more frequently and results in better outcomes than does low quality information (e.g., Maltz and Kohli 1996). Indeed, research in social networks shows that centrality is an important indicator of group performance, i.e., the extent to which groups are efficient in solving a problem (e.g., Freeman, Roeder, and Mulholland 1980). Therefore, we propose the following hypothesis:

H1: The centrality of a project positively influences the technical success of the project.
2.5.2. Project Manager Centrality and Technical Project Success

A project manager plays the key role of coordinating overall project development activity, i.e., s/he delegates roles to developers based on their skill sets, determines the timing of version releases, and allows entry of new team members. Project manager centrality is higher when the manager is working on a larger number of projects (degree centrality), serves as a conduit for information exchange among project teams (betweenness centrality), and participates in important (central) projects (eigenvector centrality). Indeed, the larger number and important linkages of project managers should result in the managers having access to higher quality information and this information quality should result in a higher technical success. In contrast, high centrality also implies that the project manager is working on a larger number of projects and has access to too much information, leading to the possibility of cognitive overload and poorer work performance (e.g., Rosa et al. 1999; Weick 1979), leading to lower technical success. Thus, we propose: 2

H2: For some projects, the centrality of a project manager will positively influence the technical success of the project and for other projects the centrality of the project manager will negatively influence the technical success of the project.

2.5.3. Project Centrality and Commercial Project Success

We rely on the literature in signaling theory (e.g., Spence 1974), word-of-mouth influences (e.g., Rogers 1995), and corporate reputation (e.g., Fombrun and Shanley 1990) to link centrality with software adoption. Signaling theory suggests that project centrality signals project quality such that greater centrality would imply higher quality, i.e., the users are likely to infer that more connected projects should be of higher quality (e.g., Spence 1974). Similarly, if project centrality is a signal of software quality being developed, then project centrality should increase the commercial success of the project.

The logic behind the possible effects of word of mouth influences and reputation are related.

---

2 We will rely on latent class regression analysis, which admits the possibility of multiple regimes to test our hypotheses. This approach permits us to uncover one regime in which project manager centrality has a positive influence on technical success and another in which project manager centrality has a negative effect on project success (see Appendix D for details).
The literature on social networks and diffusion of innovations shows that network structures influence the rate at which innovations diffuse (e.g., Abrahamson and Rosenkopf 1997), suggesting that central projects would be able to more successfully disseminate information concerning the projects. Clearly, the effect would depend on the valence of the information communicated, i.e., positive or negative (e.g., Mahajan, Muller, and Kerin 1984), where the valence of the word of mouth communication may depend on the reputation of the developers. As in the case of corporate reputation (e.g., Fombrun and Shanley 1990), reputation in the open source environment should be a multidimensional construct. For example, a project manager may have the reputation of developing technically sophisticated (good reputation) software that is not user friendly (bad reputation). Project centrality would facilitate the dissemination of this information. The effect, positive or negative, would depend on the valence of the information disseminated. When the valence of the salient reputation dimension is positive (negative) word-of-mouth should increase (decrease) the commercial success of the project. Thus, project centrality can have a positive or a negative effect on commercial project success.

**H3:** For some projects, the centrality of a project will positively influence the commercial success of the project and for other projects the centrality of the project would negatively influence the commercial success of the project.

### 2.5.4. Project Manager Centrality and Commercial Project Success

The reasoning behind the effect of project manager centrality on commercial project success is similar to that for the effect of project centrality. Specifically, if project manager centrality signals project quality, then project manager centrality should positively influence project success (e.g., Spence 1974). Project manager centrality should also facilitate the dissemination of word of mouth information concerning the project (e.g., Deroian 2002). Again, the valence of information disseminated, which would depend on the reputation of project and its developers, would determine whether commercial success is enhanced or reduced. Thus, parallel to the previous hypothesis, we suggest that project manager centrality can either increase or decrease commercial project success.
For some projects, the centrality of a project manager will positively influence the commercial success of the project and for other projects the centrality of the project manager will negatively influence the commercial success of the project.

3. STUDY 1: STRUCTURE OF OPEN SOURCE NETWORKS

To better understand the nature of structural embeddedness in the open source environment, we rely on two approaches: a visual approach relying on sociometrics to develop a rich, in-depth description of the relationships among projects and developers (for review see Wasserman and Faust 1999) and a statistical approach based on latent class cluster analysis to formally assess the number of groupings of project structures. Both approaches rely on the same data to generate the two-mode affiliation network structure: membership data on the project development teams for projects listed on SourceForge.net

3.1 Data Source and Data Collection Procedure

Based on the suggestions of von Hippel and von Krogh (2003), we use the website [http://www.SourceForge.net](http://www.SourceForge.net) to collect our data. Sourceforge.net is an open source initiative that provides web space to organize and co-ordinate open source product development. The site itself has been developed on open source software and the technology platform that it utilizes to facilitate services for the open source initiatives are all based on open source technology. The site hosts more than 85,000 projects with over 900,000 registered users. For the purpose of this study, we sought to sample from this population to generate a reasonable and manageable set of data.

The projects in SourceForge.net are classified under broad technology platforms called project foundries. To keep the data collection manageable, we sought a foundry with 8-15 active projects. We randomly selected the “Perl” foundry, comprising projects that share the Perl programming language as the platform technology. The foundry has 10 active projects that represent a wide range of applications such as databases, system administration, text processing, and development tools. These projects have 72 members, resulting in an affiliation matrix of 72 rows (developers) and 10 columns (projects), where each entry is a 1 if a developer worked on a project and 0 otherwise.
To view this foundry in the framework of the more complete project-developer network, we listed all non-Perl projects that these 72 developers were members of, resulting in 108 projects, including the 10 projects in the *Perl Foundry*. We also identified all other developers aside from the 72 Perl developers who were members of these additional 98 projects, resulting in a total of 490 developers including the 72 Perl developers. The resulting sociomatrix has 490 rows (developers) and 108 columns (projects), providing an appropriate sample of projects to represent the Perl affiliation networks (e.g., Faust 1997).

3.2. **Heterogeneity in the Embeddedness of Open Source Projects**

3.2.1. **Visual Representation of Network Structure**

We used the Fruchterman-Reingold algorithm (Fruchterman and Reingold 1991) in the network software package *Pajek 1.00* to develop the Perl developer membership bipartite graph. We use squares to represent the projects and triangles to represent the developers (see Figure 2).

![Figure 2](image-url)  
Note that the *Perl Foundry Network* in Figure 2 is not fully connected; i.e., there are six clusters (labeled A to F) of projects and developers that do not have connections to other clusters of projects and developers. Cluster A represents the largest connected part of the graph while Cluster F consists of three projects (“wxperl,” “bayespam”, and “dailystrips”) from the *Perl foundry* that have one developer each and do not seem to have a connection with the rest of the network. Developer number 41 in Cluster A is strategically positioned and serves as a link for project “esmf” (which has 40 developers including Developer 41). Developer 41 also works on project “pdl,” which is the largest project from the *Perl foundry* with 21 developers. Similar to Developer 41, Developer 44 working on project “pdl” has a strategic position. The second largest *Perl foundry* project, “misterhouse,” with 14 developers, also belongs in Cluster A. In contrast, “slashcode,” a Perl foundry project with 9 developers, is in Cluster D. Indeed, Figure 2 strongly suggests that considerable heterogeneity exists in the embeddedness of project and developers in an open source environment.
3.2.2. Latent Class Cluster Analysis

Although Figure 2 visually demonstrates heterogeneity in the embeddedness of projects and developers, we sought to establish statistical differences using latent class cluster analysis. We present the technical details on this statistical methodology in Appendix C along with an associated test to determine if there is statistically significant heterogeneity within this population, which rejects the homogeneity hypothesis. Specifically, we use the likelihood dominance criterion (Pollak and Wales 1991) to show that a model with either two clusters or six clusters (our optimal solution) is superior to a model with a single cluster (p < .01) (for details see Appendix C). To identify the optimal number of clusters, we estimated models with 1 to 8 clusters and use the Bayesian Information Criteria (BIC) and the Consistent Akaike Information Criteria (CAIC) to select the optimal solution (see table in Appendix C). While these two criteria do not agree in our case, we found that going beyond six clusters lead to the identification of separate, albeit large, idiosyncratic projects. Indeed, the entropy of separation of this six cluster model was .98, suggesting good separation among the clusters (Wedel and Kamakura 2000 recommend ES > .90). Note that the fact that a multi-cluster solution is statistically superior to a one-cluster solution supports the notion of heterogeneity in structural embeddedness of projects and project managers.

In Table 1 we present descriptive statistics for the structural embeddedness variables for the six-cluster solution. Cluster 1 consists of projects with low levels of embeddedness for nearly all the centrality variables. The mean values for five of the six-centrality measures for Cluster 1 are lower than the overall mean, representing a large number of drifter (disconnected) projects (60% of total projects) that have low structural centrality. Cluster 2, which comprises 16% of the projects, has medium levels of project degree (mean of 10.62 compared with the overall mean of 5.75) and betweenness centrality (mean of 1527 compared with the overall mean of 907), but scores low on the other centrality measures. Cluster 3, which comprises 12% of the projects, is the mirror image of Cluster 2. The projects in this cluster have medium levels of degree centrality (mean of 10.0 compared with the sample mean of 7.7) and betweenness centrality (mean of 8610 compared with the sample mean of 2114) for project
managers, but score low values on the remaining centrality measures. Cluster 4, comprising 4.7% of the projects, has project managers with high centrality (mean of 9.8, 5099, and 3.2 for degree, betweenness, and eigenvector centrality respectively compared with the sample means of 7.7, 2114, and .28 respectively) and on project eigenvector centrality (mean of .32 compared with the sample mean of .13). Cluster 5 projects (3.8% of the total projects) have high levels of project degree and betweenness centrality (mean of 33.2 and 9846 for degree and betweenness centrality respectively compared with sample mean of 5.6 and 908 respectively) but low levels of project manager centrality (mean values of 3.0, .01, and .02 for degree, betweenness, and eigenvector centrality respectively compared with the sample mean of 7.7, 2114, and .28 respectively). Thus, some developers for the projects, but not project managers may have high levels of structural embeddedness. Finally, the smallest cluster (Cluster 6 comprising 2.9% of the projects), scores high on three centrality measures across both projects (mean values of 21.0, 4003, and 4.08 for degree, betweenness, and eigenvector centrality respectively compared with the sample mean of 5.75, 907, and .13 respectively) and project managers (means of 13.0, 5875, and 4.51 for degree, betweenness, and eigenvector centrality respectively compared with the sample mean of 7.72, 2114, and .28 respectively). Heterogeneity in structural embeddedness also seems to exist for the ten Perl foundry projects: while six the ten Perl foundry projects belong to Cluster 1, the other four are spread among three different clusters.

To summarize these results, we classified the six clusters along two grids: one for project centrality and the other for project manager centrality (Figure 3). We use degree/betweenness centrality (which generally covary) on the x-axis and eigenvector centrality on the y-axis. Clusters 1 and 3 are similar to each other and score high on project manager degree/betweenness centrality only, while Cluster 2 scores high on project degree/betweenness centrality only. Clusters 4 and 5 are mirror images of one another, where Cluster 4 is high on project eigenvector and project manager degree/betweenness centrality, while Cluster 5 is high on project degree/betweenness and project manager eigenvector centrality. Here again,
Cluster 6 is high on all centrality measures. Thus, study 1 shows that there is considerable heterogeneity in the structural embeddedness of projects and developers.

[Insert Figure 3 about here]

4. STUDY 2: OPEN-SOURCE PROJECT SUCCESS

Given the heterogeneity we found in Study 1 in the embeddedness of both projects and developers, in Study 2, we examine the performance consequences of differences in structural embeddedness. However, we must control for other correlates of project success in our model, an issue we address next.

4.1. Other Correlates of Project Success

All of the “new products” that emerge from the Perl Foundry are in the same general market; hence, most of the differentiators of new product success that Cooper (2001) and Griffin (1997) have identified are likely to be common across these projects. There are differences, however, in the age of the project, its market potential or interest level, and the role that users, lead users in particular, play; factors that we can measure relatively easily. Number of page views directly signals the general interest level in the project and its market potential. And as the number of CVS commits and downloads are likely to increase with the age of a project, we use number of months since the inception of the project to control for the age of the project. Users often play a critical role in the development of new products in general (e.g., von Hippel and Katz 2002), with lead users being particularly effective in driving success (e.g., Lilien et al. 2002). Bugs and support requests represent user and lead user input in the open source world, with those requests often having directed solutions associated with them. In Table 2 we provide the descriptive statistics and correlation coefficients for the variables examined in Study 2.

[Insert Table 2 about here]

4.2. Modeling and Analysis Approach

Although both our dependent measures, i.e., number of CVS commits and downloads, are count measures, their mean and standard deviations are fairly large (see Table 2), and heavily skewed. Thus, we took the logarithm of these two variables and approximate them as continuous variables for our analysis.
To evaluate the distribution of these two continuous variables, we developed kernel density plots that showed a bi-modal distribution, indicating multiple regimes or multiple relationships between each dependent and the independent variables (see Figure D1 of Appendix D). Latent class regression analysis (e.g., Wedel and Kamakura 2000), which is based on finite mixture theory (e.g., Titterington, Smith, and Makov 1985), provides an appropriate methodology to simultaneously estimate multiple relationships among dependent and independent variables (for technical details see Appendix D). Thus, using latent class regression analysis, we first determine the number of regimes (using BIC and CAIC) and then estimate the coefficients for the effect of each independent variable on the dependent variable for each of these regimes.

4.3. Model Selection

We generated both BIC and CAIC for the model for CVS commits and downloads (for detailed results see table in Appendix D). The results clearly suggest two regimes for number of downloads, which is consistent with the kernel density plots (ES = .99). The results are a bit ambiguous for CVS commits with BIC suggesting two regimes and CAIC suggesting a single regime. Given the bimodality in the kernel density plot, we pursued the two-regime solution, and the high entropy of separation for the two-regime solution (ES = .98) provides support for this two-regime solution. Thus, we explored two-regime solutions for both CVS commits and downloads.

4.4. Results

4.4.1. CVS Commits

In Table 3, we present the results for the two-regime solution for CVS commits as well as the mean and standard deviations for the variables in each of these two regimes. The results on CVS commits permit us to test our first two hypotheses on the technical success of open source projects. We find support for H1, as degree centrality in Regime 1 (b = .88, p < .05) and betweenness centrality in Regime 2 (b = 3.81, p < .01) positively influence technical project success.
In H2 we suggested that on one hand the centrality of the project manager would result in access to quality information and thus would result in technical project success, while on the other hand, the demands for time placed on a central project manager may compromise the quality of work on the project and thus result in lower technical success. We find support for both these assertions: in Regime 1 we find negative effects, where coefficients for degree (b = -.49, p < .05) and eigenvector (b = -.96, p < .10) centrality are both negative, and in Regime 2 we find positive effects, where the coefficient for degree (b = .37, p < .01) and betweenness (b = .26, p < .01) project manager centrality are positive.

In terms of the control variables, the results on project age suggest that older projects tend to have more CVS commits for both regimes, but the effect for Regime 1 (b = .97, p < .01) is over six times higher than the effect for Regime 2 (b = .16, p < .10). Number of page views has a positive but statistically non-significant effect in Regime 1 (b = .66, p > .10), but has a negative and statistically significant effect in Regime 2 (b = -.47, p < .01). We find positive and statistically significant effects for number of bugs closed (b = .54, p < .01) and number of support request answered (11.98, p < .01) for Regime 2, but the effects for these variables are not statistically significant in Regime 1.

The descriptive statistics in Table 3 show that, compared with Regime 1, Regime 2 has (1) more technically successful projects, as shown by greater number of CVS commits, (2) projects with greater market potential (as shown by higher number of page views) and activity levels (more bugs closed in Regime 2 than Regime 1, although projects in Regime 1 have more support requests answered), and (3) greater eigenvector centrality (degree and betweenness centrality are fairly similar across the two regimes). The 10 Perl foundry projects were equally split between the two regimes, with “dailystrips,” “guido,” “misterhouse,” “pdl,” and “wxperl” belonging to Regime 1 and “amphetadesk,” “apachetoolbox,” “bayesspam,” “slashcode,” and “spamassassin” in Regime 2.
4.4.2. Downloads

In Table 4, we present the results for the two-regime solution for number of downloads in a format that parallels Table 3. With these results on downloads, we can test the last two hypotheses on the commercial success of open source projects. We find support for H3, where degree centrality positively influences the commercial success of projects in Regime 1 (b = .66, p < .01), while betweenness centrality negatively influences commercial success of projects in both regimes (Regime 1: b = -.12, p < .05; Regime 2: b = -.57, p < .01) and eigenvector centrality negatively influences the commercial success of projects in Regime 1(b = -2.22, p < .05). The support for H4 is much weaker; showing that eigenvector centrality positively influences the commercial success of projects in Regime 1 (b = 1.46, p < .10) and negatively influences the commercial success of projects in Regime 2 (b = -.14, p < .10).

The results also show that number of CVS commits increases the commercial success of projects in Regime 2 (b = .38, p < .10). As project age increases, the number of downloads increases in Regime 1 (b = .63, p < .01), while page views positively influences number of downloads in both regimes (Regime 1: b = 8.35, p < .01; Regime 2: b = 1.25, p < .01). Number of bugs closed positively influences number of downloads in Regime 2 (b = 1.09, p < .01), while number of support requests answered negatively influences number of downloads in Regime 1 (b = -.77, p < .01) and positively influences number of downloads in Regime 2 (b = .82, p < .01).

The descriptive statistics in Table 4 show that projects in Regime 1 have more downloads than those in Regime 2, but have fewer CVS commits, page views, bugs closed, and support requests answered than projects in Regime 2. Overall, the structural embeddedness of projects in Regime 2 is much higher than of projects in Regime 1. Specifically, project eigenvector centrality and project manager betweenness and eigenvector centrality are higher for projects in Regime 2 than Regime 1. However, project betweenness centrality is higher for projects in Regime 1 than for projects in Regime 2. Eight of the 10 Perl foundry projects belonged to Regime 1, with “slashcode,” and “spamassassin” in Regime 2.
4.4.3. Comparing Regimes

For both the dependent variables, i.e., number of CVS commits and downloads, we found a two regime solution. One might assert that these two regimes should contain the same projects, i.e., that regime identity should hold across both CVS commits and downloads, a constraint we did NOT impose on the models, which were calibrated independently. To explore this issue, we show these cross-tabulation results in Table 5, which strongly suggest different drivers for regime membership for CVS commits and downloads. The cell sizes range from 19 to 39 and there is no difference among them in terms of number of developers. The table provides some commentary on the characteristics of these cells, with Cell B (Regime 1 for Downloads and Regime 2 for CVS commits) highest on average downloads.

[Insert Table 5 about here]

5. DISCUSSION

We have studied how structural embeddedness of projects and developers relate to the success of open source projects in two studies, one focusing on the differences between the structure of the networks surrounding such projects and the second relating that network structure to two measures of project success.

Study 1 demonstrated that considerable heterogeneity exists in the embeddedness of projects and developers in an open source environment. Using the Perl foundry as a starting point and building an affiliation network around it, we found three independent Perl foundry projects (Cluster F in Figure 2) that were not linked with rest of the network. The largest cluster (Cluster A in Figure 2) was built around the largest Perl foundry project (“pdl”) and consisted of other important Perl foundry projects such as “misterhouse” and “amphedesk”. We were also able to identify centrally located developers such as 41 and 44 in cluster A, who might be monitored in the future to see what their role is or could be in facilitating project success. This representation of the affiliation network of the Perl foundry provides a rich, visual understanding of the structure of relations in an open source environment. Our latent class
cluster analysis provided formal support for the observed heterogeneity in structural embeddedness of projects and project managers based on measures of centrality. Here a six-cluster solution showed considerable variation in structural embeddedness of projects and project managers across developers. The 10 Perl foundry projects belong to five of the six clusters, further strengthening the notion of heterogeneity in structural embeddedness.

Study 2 relates the heterogeneity in structural embeddedness to the success of software development projects in an open source environment. We focused on both technical success, viewed in terms of number of CVS commits, and commercial success operationalized as number of downloads. Our results suggest that project centrality improves the technical success of projects, but can either enhance (as shown by degree centrality in Table 4) or reduce (as shown by betweenness and eigenvector centrality in Table 4) the commercial success of projects. Thus, the availability of greater resources (as operationalized by degree centrality, i.e., number of developers) and higher quality of information can enable improvements in technical success. We relied on the literature in signaling theory to suggest that project centrality would serve as a signal of project quality and thus greater centrality would imply greater project success, an assertion supported in the case of project degree centrality. Indeed, project degree centrality, which equals the number of developers, serves as a strong signal of project importance and thus becomes a strong driver of commercial project success. We also reasoned that the perceived reputation of a project and its developers would be diffused at a higher rate for more central projects and that the valence of the reputation or the most salient reputation dimension would determine the direction of the effect on commercial project success.\(^3\) The results for betweenness and eigenvector centrality support these assertions.

\(^3\) The saliency of the reputation dimension is important, as users are unlikely to weigh every dimension of reputation equally. For example, users may prefer easy-to-use software that is not technically that advanced, to hard-to-use software that is technically advanced. As a result, easy-to-use takes on salience and the technically competent hard-to-use software gets negative word of mouth.
Project manager centrality seems to have both negative and positive effects on technical project success: the number of projects the project manager is working on can provide access to a richer knowledge base and thus greater technical success or could distract the manager, thereby reducing the technical success of projects. We find support for both assertions concerning degree centrality (Table 3). Thus, in Regime 2 we find effects from gains associated with access to richer knowledge stores and in Regime 1 we find support for pitfalls associated with working on multiple projects. The effects of project manager centrality on commercial project success are rather weak (p < .10 for eigenvector centrality; Table 4). Here again, we had relied on literature on signaling theory to suggest a positive effect and on word of mouth influences to suggest effects consistent with valence of salient reputation dimension. The results suggest that project centrality that is based on the number of developers working on a project and the linkages of these developers serve as much stronger signals and enable better diffusion of reputation information than project manager centrality, which is based on the projects the managers are working on.

The results for the effects of embeddedness are much stronger for technical success (number of CVS commits) than those associated with commercial success (number of downloads), implying that structural embeddedness has a greater role to play in technical success than in commercial success. Such a conclusion may follow from the fact that embeddedness enables projects to attract talented developers, but is invisible to the users, who drive commercial success. In fact, we find a weak link (p < .10) between technical projects success and commercial project success (Table 4 – results for Regime 2), which seems to be consistent with the literature on new product development in general (e.g., Mansfield and Wagner 1976).

Although our results across the two studies are informative on heterogeneity in structural embeddedness and the effect of this heterogeneity for project success, we must stress the exploratory nature of our research. As research on open systems environments is new, theoretical insights in this domain are just emerging (von Hippel and von Krogh 2003). Nonetheless, research streams such as the
importance of lead users in new product development (e.g., Lilien et al. 2002), on embeddedness in organizational sociology (e.g., Granovetter 1985), and signaling theory in economics (e.g., Spence 1974) among other, provide bases for speculating about the potential for and the most effective management of open source software development projects. In this research, we find that significant heterogeneity exists in the embeddedness of open source projects and there seems no reason to expect this result not to hold for other open source projects. We also find that the structure of projects and project managers strongly affects technical and commercial project success, a result that should encourage further research in the area.

Besides simple replications of our research, enriched perhaps by more direct observation (via diary or survey or the like), future research should examine other measures of embeddedness, such as those related to resources (e.g., Wasserman and Faust 1999) and of performance such as rate of innovation in projects (e.g., Chandrashekaran et al. 1999) and the nature of the innovations (e.g., radical versus incremental; Tushman and Anderson 1986). Building in dynamics by examining the effect of structural embeddedness over time should also provide new insights; we have studied this process via a static view while the dynamics of the network and the environment may have even more powerful effects. For that purpose one could rely on evolutionary theories in economics (e.g., Nelson and Winter 1982) and/or sociology (e.g., Carroll and Hannan 2000). It is our hope that our initial results encourage researchers studying open source systems to embrace a social capital perspective and that researchers in diverse social sciences will focus on this domain to provide richer insights into open source systems. Clearly, community based models, such as those of the open source environment, have the potential to become more prevalent as technological advances create a global marketplace and a careful study of how they evolve and their relative effectiveness, will likely gain in importance over time.
Figure 1
Bipartite Graph for an Affiliation Network

Affiliation Matrix for Network on the Left

<table>
<thead>
<tr>
<th></th>
<th>Deskpro</th>
<th>Screenpro</th>
<th>Keypro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Bob</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Chris</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jean</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Joan</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Key Features

- Projects share a relationship with one another due to common developers and developers share a relationship with each other due to common projects.
- Adam is the most central developer as he is the only developer on two projects and has direct relationships with three developers (Bob, Chris, and Jean).
- The graph on the left is not fully connected because we cannot move from a project to all the other projects or from a developer to all the other developers. Thus, we cannot go from Keypro to either Deskpro or Screenpro or from Jean or Dave to the first four developers.
- The affiliation matrix on the top right can be used to summarize the relationships seen in the graph on the left, where 1 signifies that a developer works on the project and 0 suggests that the developer does not work on the project.
Figure 2
Bipartite Graph of the “Perl Network”

Key Features

- The graph is not fully connected with five major clusters (A to E) with Cluster A being the largest and a cluster (F) of three independent projects (“wxperl,” “bayespam”, and “dailystrips”), which happen to be from the Perl foundry.
- Some observations:
  - Developer number 41 in Cluster A works on the largest *Perl foundry* project “pdl” with 21 developers and seems to be strategically positioned, as s/he serves as a link for project “esmf” (that has 40 developers including Developer 41).
  - Developer 44 also works on a project the *Perl foundry* “pdl” which seems to have a strategic position.
  - The second largest *Perl foundry* project “misterhouse” with 14 developers also belongs in Cluster A, while “slashcode” a Perl foundry project with 9 developers is in Cluster D.
Figure 3: Centrality and Cluster Membership

Panel A: Cluster Distribution Based on Project Centrality Measures

<table>
<thead>
<tr>
<th>Project Eigenvector Centrality</th>
<th>Project Degree/Betweenness Centrality</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Cluster 1</td>
<td></td>
<td>Cluster 2</td>
</tr>
<tr>
<td></td>
<td>Cluster 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Cluster 4</td>
<td></td>
<td>Cluster 5</td>
</tr>
<tr>
<td></td>
<td>Cluster 6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Panel B: Cluster Distribution Based on Project Manager Centrality Measures

<table>
<thead>
<tr>
<th>Project Manager Eigenvector Centrality</th>
<th>Project Manager Degree/Betweenness Centrality</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Cluster 2</td>
<td></td>
<td>Cluster 1</td>
</tr>
<tr>
<td></td>
<td>Cluster 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Cluster 5</td>
<td></td>
<td>Cluster 4</td>
</tr>
<tr>
<td></td>
<td>Cluster 6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OBSERVATIONS
- Clusters 1 and 3 are high only on project manager degree\betweenness centrality.
- Cluster 2 is high only on project degree\betweenness centrality.
- Cluster 4 is high on project eigenvector and project manager degree\betweenness centrality.
- Cluster 5 is high on project degree\betweenness and project manager eigenvector centrality.
- Cluster 6 is high on all centrality measures.
Table 1: Network Variable Means for the Six Clusters

<table>
<thead>
<tr>
<th>Variable Category</th>
<th>Variable Name</th>
<th>Cluster Numbers</th>
<th>Full Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Degree Centrality</td>
<td></td>
<td>2.66</td>
<td>10.62</td>
</tr>
<tr>
<td>Betweenness Centrality</td>
<td></td>
<td>159.10</td>
<td>1527.42</td>
</tr>
<tr>
<td>Eigenvector Centrality*</td>
<td></td>
<td>.00</td>
<td>.01</td>
</tr>
<tr>
<td>Degree Centrality</td>
<td></td>
<td>8.06</td>
<td>4.41</td>
</tr>
<tr>
<td>Betweenness Centrality</td>
<td></td>
<td>862.34</td>
<td>954.79</td>
</tr>
<tr>
<td>Eigenvector Centrality*</td>
<td></td>
<td>.02</td>
<td>.04</td>
</tr>
<tr>
<td>Cluster Size (% cases)</td>
<td></td>
<td>60.02</td>
<td>16.43</td>
</tr>
</tbody>
</table>

*The coefficient for eigenvector centrality was multiplied by 100 for ease of presentation of the results.

**Highlights:**

- Cluster 1 consists of larger number of projects with low levels of structural embeddedness.
- Cluster 2 projects have medium levels of project degree and betweenness centrality, but score low on the other four embeddedness measures.
- Cluster 3 projects are mirror image of Cluster 2 projects with medium levels of project manager degree and betweenness centrality, but score low on the other four embeddedness measures.
- Cluster 4 projects score well on four of the six structural embeddedness measures.
- Cluster 5 projects score well on project embeddedness measures but poorly on the project manager embeddedness measures.
- Cluster 6 projects are highly embedded on all the centrality measures.
Table 2: Descriptive Statistics and Bivariate Correlation Coefficients

<table>
<thead>
<tr>
<th></th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
<th>X9</th>
<th>X10</th>
<th>X11</th>
<th>X12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Downloads (X1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of CVS Commits (X2)</td>
<td>.034</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Page Views (X3)</td>
<td>.13</td>
<td>.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Age (X4)</td>
<td>.20*</td>
<td>.19</td>
<td>.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Bugs Closed (X5)</td>
<td>.33**</td>
<td>.22*</td>
<td>.09</td>
<td>.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Support Requests</td>
<td>.11</td>
<td>.08</td>
<td>.13</td>
<td>- .00</td>
<td>.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Answered (X6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Degree Centrality (X7)</td>
<td>.15</td>
<td>.57**</td>
<td>.13</td>
<td>.19*</td>
<td>.18</td>
<td>.32**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Betweenness Centrality (X8)</td>
<td>.16</td>
<td>.27**</td>
<td>.19</td>
<td>.18</td>
<td>.05</td>
<td>.21*</td>
<td>.64**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Eigenvector Centrality (X9)</td>
<td>- .04</td>
<td>.94**</td>
<td>.01</td>
<td>.14</td>
<td>.19*</td>
<td>.00</td>
<td>.47**</td>
<td>.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Manager Degree</td>
<td>- .08</td>
<td>.08</td>
<td>.07</td>
<td>.12</td>
<td>.24**</td>
<td>- .12</td>
<td>- .15</td>
<td>- .13</td>
<td>.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centrality (X10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Manager Betweenness</td>
<td>- .01</td>
<td>.14</td>
<td>.06</td>
<td>.01</td>
<td>-.04</td>
<td>-.10</td>
<td>-.05</td>
<td>-.01</td>
<td>.14</td>
<td>.28**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centrality (X11)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Manager Eigenvector</td>
<td>- .08</td>
<td>.65**</td>
<td>.13</td>
<td>.11</td>
<td>.11</td>
<td>-.02</td>
<td>.30**</td>
<td>.11</td>
<td>.77**</td>
<td>.18</td>
<td>.28**</td>
<td></td>
</tr>
<tr>
<td>Centrality (X12)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Mean                           | 10398.36 | 1320.83 | 219870.54 | 1118.64 | 60.01  | 1.48   | 5.75   | 906.74 | 0.01   | 7.72   | 2113.69 | .03   |
| Standard Deviation             | 28517.92 | 6208.50 | 1070492.27 | 405.55  | 385.39 | 8.17   | 7.81   | 2374.19 | 0.10   | 5.37   | 2875.89 | .11   |

*p < .10
**p < .05
### Table 3: Results for the Two-Regime Solution Model for CVS Commits

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable Name</th>
<th>Regime 1</th>
<th>Regime 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Estimate (SE)</td>
<td>Mean (SD)</td>
</tr>
<tr>
<td>Project Age</td>
<td></td>
<td>.97*** (0.27)</td>
<td>1,161.55 (393.68)</td>
</tr>
<tr>
<td>Control</td>
<td>Page Views</td>
<td>.66 (.55)</td>
<td>83,030.15 (216,419.97)</td>
</tr>
<tr>
<td></td>
<td>Bugs Closed</td>
<td>.26 (.31)</td>
<td>22.22 (85.13)</td>
</tr>
<tr>
<td></td>
<td>Support Requests</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Answered</td>
<td>.06 (.20)</td>
<td>2.23 (10.88)</td>
</tr>
<tr>
<td>Project Centrality</td>
<td>Degree</td>
<td>.88** (.40)</td>
<td>5.87 (7.23)</td>
</tr>
<tr>
<td></td>
<td>Betweenness</td>
<td>-.04 (.27)</td>
<td>1,057.44 (2,903.67)</td>
</tr>
<tr>
<td></td>
<td>Eigenvector</td>
<td>.67 (.61)</td>
<td>.01 (.03)</td>
</tr>
<tr>
<td>Project Manager</td>
<td>Degree</td>
<td>-.49** (.25)</td>
<td>7.35 (5.48)</td>
</tr>
<tr>
<td>Centrality</td>
<td>Betweenness</td>
<td>.29 (.25)</td>
<td>2,038.76 (2,847.69)</td>
</tr>
<tr>
<td></td>
<td>Eigenvector</td>
<td>-.96* (.62)</td>
<td>.02 (.07)</td>
</tr>
</tbody>
</table>

* p < .10
** p < .05
*** p < .01

NOTES: We report one-tail tests for statistical significance. For each regime we have two columns of results. In the first we report the regression coefficient and its standard error in parenthesis and in the second we report the mean of the explanatory variable with its standard deviation in parenthesis.

OBSERVATIONS:
- H1 is supported, as project centrality tends to increase technical project success for both the regimes.
- Consistent with H2, in Regime 1 we find that project manager centrality reduces the technical success of projects, while in Regime 2 project manager centrality increases the technical success of projects.
- The notion of aggregation bias is evident from these results (e.g., DeSarbo, Jedidi, and Sinha 2001). For example, the effect of project manager degree centrality is negative for Regime 1 and positive for Regime 2. In the aggregate model we would expect these to cancel out.
- In Regime 1, 992.18 and 1896.26 were the mean and standard deviations for CVS commits, while in Regime 2, 1731.65 and 9106.50 were the mean and standard deviations. Thus, projects in Regime 2 have greater number of CVS commits than projects in Regime 1.
- Regime 2, when compared with Regime 1 has projects with greater market potential (as shown by higher number of page views) and activity levels (larger number of bugs closed on Regime 2 than Regime 1, although projects in Regime 1 has a larger number of support requests answered).
- Regime 2, when compared with Regime 1 has greater eigenvector centrality (degree and betweenness centrality seems to be fairly similar across the two regimes).
Table 4: Results for the Two-Regime Solution Model for Downloads

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable Name</th>
<th>Regime 1</th>
<th>Regime 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Estimate (SE) Mean (SD)</td>
<td>Estimate (SE) Mean (SD)</td>
</tr>
<tr>
<td>CVS</td>
<td></td>
<td>.87 (.89) 929.28 (1,854.58)</td>
<td>.38* (.24) 1,775.04 (8931.84)</td>
</tr>
<tr>
<td>Project Age</td>
<td></td>
<td>.63*** (.25) 1,126.10 (425.87)</td>
<td>-.01 (.07) 1,109.98 (384.73)</td>
</tr>
<tr>
<td>Page Views</td>
<td></td>
<td>8.35*** (1.73) 83,423.31 (139,712.30)</td>
<td>1.25*** (.05) 378,149.32 (1,559,51)</td>
</tr>
<tr>
<td>Bugs Closed</td>
<td></td>
<td>-.13 (.37) 10.88 (25.50)</td>
<td>1.09*** (.06) 117.00 (563.38)</td>
</tr>
<tr>
<td>Support Requests Answered</td>
<td></td>
<td>-.77*** (.33) .50 (1.03)</td>
<td>.82*** (.08) 2.62 (11.92)</td>
</tr>
<tr>
<td>Degree</td>
<td></td>
<td>.66*** (.31) 5.98 (7.30)</td>
<td>.00 (.18) 5.48 (8.43)</td>
</tr>
<tr>
<td>Project Centrality</td>
<td></td>
<td>-.12*** (.24) 1,113.94 (2878.28)</td>
<td>-.57*** (.23) 666.39 (1,600.42)</td>
</tr>
<tr>
<td>Betweenness</td>
<td></td>
<td>Eigenvector</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.22*** (.35) .00 (.01)</td>
<td>.32 (.26) .027 (.14)</td>
</tr>
<tr>
<td>Project Manager Centrality</td>
<td></td>
<td>Degree</td>
<td>Project Manager Centrality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00 (.25) 6.79 (5.67)</td>
<td>-.04 (.085) 8.80 (4.84)</td>
</tr>
<tr>
<td>Betweenness</td>
<td></td>
<td>.20 (.28) 1,546.46 (2,625.25)</td>
<td>.04 (.07) 2,771.68 (3,036.59)</td>
</tr>
<tr>
<td>Eigenvector</td>
<td></td>
<td>1.46* (1.11) .00 (0.03)</td>
<td>-.142* (.09) .06 (.15)</td>
</tr>
</tbody>
</table>

* p < .10
** p < .05
*** p < .01

NOTES: We report one-tail tests for statistical significance. For each regime we have two columns of results. In the first we report the regression coefficient and its standard error in parenthesis and in the column we report the mean of the explanatory variable with its standard deviation in parenthesis.

OBSERVATIONS

- Consistent with H3, we find that degree centrality positively influences the commercial success of projects in Regime 1, while project betweenness centrality negatively influences commercial success of projects in both the regimes and project eigenvector centrality negatively influences commercial success of projects in Regime 1.
- We find weak support for H4. Specifically, eigenvector centrality positively influences the commercial success of projects in Regime 1 and negatively influences the commercial success of projects in Regime 2.
- Here again, the notion of aggregation bias is evident from the results (e.g., DeSarbo et al. 2001). For example, the effect of the number of support requests answered is negative for Regime 1 and positive for Regime 2. In the aggregate model we would expect these to cancel out.
- Regime 1 (mean = 16158.33, Standard Deviation = 35000.18) projects tends to have larger number of downloads than projects in Regime 2 (mean = 3716.80, Standard Deviation = 16311.95).
- Regime 1 projects seem to have few number of CVS commits, page views, bugs closed, and support requests answered than projects in Regime 2.
- Overall, the structural embeddedness (centrality) of projects in Regime 2 is much higher than projects in Regime 1.
### Table 5: Cross Classification of Regimes across Downloads and CVS models

<table>
<thead>
<tr>
<th></th>
<th>CVS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Regime 1</td>
<td>Regime 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Regime Statistics</td>
<td>No. of Developers</td>
<td>CVS Commits</td>
<td>Downloads</td>
</tr>
<tr>
<td>CELL A</td>
<td>Number of cases: 39</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>5.9</td>
<td>1,138.7</td>
<td>11,766.5</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>6.4</td>
<td>2,056.3</td>
<td>22,020.9</td>
</tr>
<tr>
<td></td>
<td>Exemplars:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dailystrips</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>octave</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>misterhouse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pdl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>modinfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CELL B</td>
<td>Number of cases: 19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>6.0</td>
<td>499.2</td>
<td>25,139.1</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>9.0</td>
<td>1,294.3</td>
<td>52,246.5</td>
</tr>
<tr>
<td></td>
<td>Exemplars:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gtk2-perl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>biblook</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>renai</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>plplot</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>amphetadesk</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CELL C</td>
<td>Number of cases: 21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>5.7</td>
<td>720.1</td>
<td>2,196.3</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>8.7</td>
<td>1,566.5</td>
<td>10,046.6</td>
</tr>
<tr>
<td></td>
<td>Exemplars:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tidy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>esmf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbeating</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dbschema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>shuck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CELL D</td>
<td>Number of cases: 29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>5.3</td>
<td>2,539.1</td>
<td>4,840.2</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>8.4</td>
<td>11,679.8</td>
<td>19,757.1</td>
</tr>
<tr>
<td></td>
<td>Exemplars:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>htdig</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mac-ae-simple</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lsb</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spamassassin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>slashcode</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:** For each cell, we have provided the overall mean and standard deviation along with a few exemplar projects. The *Perl foundry* projects are shown in bold.

**OBSERVATIONS**

- There is no difference among the four cells in terms of number of developers.
- Cell B has the highest number of downloads.
- Cell D has the highest number of CVS commits.
- Cell A has the second highest number of both CVS commits and downloads.
- Cell C does not have any *Perl foundry* projects. This cell is relatively on the low side on downloads, but has medium levels of CVS commits.
REFERENCES


APPENDIX A
OPEN SOURCE SOFTWARE AND SOURCEFORGE.NET

Open Source Software

The Open Source software movement has contributed to the birth of a number of successful products, the most notable among them being the Linux OS, Apache Web Server, MySQL Database and Sendmail. However, there is still some debate as to what exactly is “Open Source”. The very definition of “Open Source” says that it is software whose source code is available to its users for further modification and refinement. A review of the various related open source movements reveals that broadly there are two powerful camps, namely, the free software movement popularized by The Free Software Foundation and open source movement of which the Apache Software Foundation is the most popular example.

Proponents of the free software movement have argued that while theirs is a social movement, “open source” is a development methodology. Richard Stallman, the founder of The Free Software Foundation puts it aptly by saying that “For the Open Source movement, non-free software is a suboptimal solution whereas, for the Free Software movement, non-free software is a social problem and free software is the solution” (Gay 2001). We adopt the notion that “Open Source” is a development methodology and acknowledge that our interest lies in studying the different variables that affect the efficacy of this knowledge development methodology, which is primarily based on the community model, rather than discussing the philosophy of the movement itself.

There are various licensing mechanisms such as the GPL (GNU Public license - http://www.gnu.org/licenses/gpl.html), OSL (Open Software License- http://opensource.org/licenses/osl-2.1.php) and Apache License (http://www.apache.org/licenses/LICENSE-2.0) that are available for distributing open-source software but the underlying common theme is that the end-users are free to modify
and further refine the source code of the software for their own use, as long as they acknowledge the original license. A license that gives users great flexibility in modifying the software to fit their needs, is said to be “copylefted”. The huge popularity of products such as the Apache Web Server and the Linux OS when compared to commercial products can be explained by the flexibility these copylefts give users. Copyleft is a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well (Gay 2001).

The Open Source Initiative (http://www.opensource.org) is a non-profit corporation that is geared towards promoting the open source movement. Most of the open-source projects however have their own websites and project level activity is coordinated at these websites. For instance, the Linux OS project is coordinated through http://www.linux.org and the Apache Web Server through http://www.apache.org.

The Open-Source movement has also found proponents in the commercial sector. Companies such as IBM have expressed strategic interest by making their products compatible with open source offerings such as Linux, and participate in a number of open source projects (150 and growing in IBM’s case). The Open source movement also received a boost when IBM donated more than half-million lines of relational database code to the Apache Software Foundation. Some of the other companies which are actively involved in the open-source movement include Sun Microsystems with its OpenOffice product suite (http://www.openoffice.org), BEA Systems with its contributions to Apache Beehive and XMLBeans technologies (http://incubator.apache.org/beehive/).

SourceForge.net

Sourceforge.net (http://www.sourceforge.net) is the world’s largest Open Source collaborative software development website. Individuals interested in contributing to the
open source movement can register themselves as developers for any of the more than
85,000 projects that are currently hosted on the website. There are over 900,000 registered
members on the website, including contributors and users. The website is owned and
operated by OSTG Inc. (Open Source Technology Group, a subsidiary of VA Software
Corporation), a community driven media network on the Internet. The company also owns
and operates other initiatives such as www.slashdot.org (for news discussion) and
www.thinkgeek.com (innovative e-commerce products for the masses).

The collaborative technology that drives Sourceforge.net is itself based on open source
technology and is a registered project titled “alexandria” on the website
(https://www.sourceforge.net/projects/alexandria). The website provides tools to look at
the activity and usage statistics of the projects, their descriptions and information on
registered developers. The co-ordination of development at the project level is supervised
by the project administrator (typically the initiator of the project), who delegates tasks to various
developers and assigns them the roles of developer, consultant, document writer, translator,
and packager.

The projects can be grouped based on a wide range of variables including technology
platform, application category, and type of users, programming language, and stage of
product development. Apart from customizable classifications that one can do, the website
provides something called the “foundry system” for grouping projects. There are currently
nineteen foundries on the sourceforge.net website, each focusing on a popular language (e.g.,
Java, Perl), technology (e.g., clustering, distributed computing) or type of software (e.g.,
databases, messaging clients). Foundries provide a common platform for developers and
end-users to communicate with each other and further improve the development process.
Foundries also allow users and developers to identify complementary technologies, thereby facilitating further integration across projects.

Foundries are managed by at least one foundry guide who has good technical experience in the area covered by the foundry. These guides act as the direct personal link between visitors and the foundry’s development community. Articles written by top developers are available in the featured articles section in each foundry. Apart from the featured articles, the foundries also provide discussion forums and email based mailing lists, which can be used for queries related to the foundry topic, obtain feedback regarding a specific project under development and also obtain support from the open-source community for technologies that are covered by the foundry. This process enables dissemination of knowledge to a much wider audience and developers can access such systematic knowledge stores for enhancing their development skills and further contribute to the open source development community.

Exhibit A1 is a screen capture of the home page of the Perl Foundry, which we use in our data analysis. The projects that fall under this foundry are classified based on their application categories such as software used in database technology (e.g., slashcode and pdl) and development tools (e.g., apachetoolbox, wxperl), among other things. The homepage also serves as a common platform for developers interested in developing their skills in the technology associated with the foundry (in this case the Perl programming language).

Exhibit A2 is a screen capture of the home page of the project titled, “pdl”, which stands for perl data language. The tool basically allows basic perl programmers to compactly store and speedily manipulate large multi-dimensional data sets. The home page also lists all the developers associated with the project on the right hand corner of the page. The tabs provided on the top of the page provide access to the other parts of the project such as
support issues, feature requests, bugs etc. Details regarding the development stage of the project, various classification schemes, such as intended audience and operating system compatible with, are provided at the bottom of the page. This homepage serves as the central page for people interested in collaborating on the project.

Exhibit A3 is a screen capture of the page on which the developers associated with the project “pdl” are listed. The page also lists the roles associated with the developers and by clicking on the developers’ usernames, one can access the profile of the developer, which would then give information on all the projects that particular developer is associated with.

**Exhibit A1: The Home Page of the Perl Foundry on Sourceforge.net**

(http://perl.foundries.sourceforge.net/repository.pl?section=perl&op=list)
Exhibit A2: The Home Page of project “pdl” in the Perl Foundry

(http://sourceforge.net/projects/pdl)
Exhibit A3: The developers page in project “pdl”
(http://sourceforge.net/project/memberlist.php?group_id=612)

References

APPENDIX B
TWO-MODE AFFILIATION NETWORKS

Consider an affiliation network $A$ in which the rows represent the actors (project managers) and columns the events (projects) with 1 when an actor belongs to an event and 0 otherwise (similar to the one shown in Figure 1). From this non-valued (i.e., the elements of the matrix are either 0 or 1) affiliation matrix, we can obtain the valued matrix (where higher value indicate greater strength of relationship) for actors ($X^A$) and events ($X^E$) as:

(B1a) $X^A = AA'$

(B1b) $X^E = A'A$

For the illustrative example represented in figure 1, the affiliation matrix $A$ will be,

$$A = \begin{bmatrix}
1 & 1 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1
\end{bmatrix}$$

And its transpose $A' = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}$

And therefore,

$$X^A = AA' = \begin{bmatrix}
2 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}$$

$$X^E = A'A = \begin{bmatrix}
3 & 1 & 0 \\
1 & 2 & 0 \\
0 & 0 & 2
\end{bmatrix}$$

We define degree centrality for actors ($C_D(X^A)$) as (e.g., Faust 1997):

(B2) $C_D(X^A) = \sum_{i=1}^{I} X^A_{ii}$
where, the network has \( I \) actors. Thus, we calculate degree centrality as the sum of the diagonal elements of \( X^A \). The degree centrality for events is calculated in a similar manner.

For the illustrative example, the degree centrality for the actor Adam, will therefore be
\[
\sum_{i=1}^{I} X^A_{ii} = 2, \text{ where } i=1 \text{ (Adam)}. \text{ The degree centrality for the other actors and the projects can be calculated in a similar manner and these values are presented in Exhibit B1.}
\]

**Betweenness centrality** relies on the notion of geodesic paths i.e., shortest path between two actors or events. The two-step procedure for calculating betweenness centrality involves calculating ‘partial betweenness’ of actors first and then using this partial betweenness to calculate actor betweenness (e.g., Freeman 1979). An actor’s partial betweenness \( \langle p \rangle \) is the number of pair of actors whose geodesic paths contain the actor \( i \). In case of ties, i.e., when there are multiple geodesic paths between two actors, partial credit is given to actors. Betweenness centrality \( (C_B(X^A)) \) for this actor is then given as:

\[
(B3) \quad C_B(X^A) = \sum_{j<k} g_{jk}(p_i) / g_{jk}
\]

where, \( g_{jk} \) is the number of geodesic between actors \( j \) and \( k \) and \( g_{jk}(p_i) \) is the number of geodesic between \( j \) and \( k \) that contain \( i \). We define the betweenness centrality for events (projects) in a similar manner.

For the illustrative example, the betweenness centrality for the actor Adam, will therefore be
\[
\sum_{j<k} g_{jk}(p_i) / g_{jk} = 6, \text{ where } i=1 \text{ (Adam) and } j \text{ and } k \text{ are all the other nodes. The betweenness centrality for the other nodes can be calculated in a similar manner and these values are presented in Exhibit B1.}
Eigenvector centrality should be high for project managers that are connected to other central project managers. Thus, the eigenvector centrality for an actor depends on the strength of the ties of other actors to which this actor is connected (e.g., Faust 1997; Friedken 1991). For example, in the studies of corporate board of directors, eigenvector centrality is considered to be most important centrality measure, as the social ties of board of directors provides legitimacy to the organizations and provides access to embedded resources (e.g., Rosenthal et al. 1985; Roy 1983).

In an affiliation network such as ours, projects can only be adjacent to developers and developers can only be adjacent to projects, which implies that the eigenvector centrality of projects is a function of the centrality of the developers associated with it and the eigenvector centrality of developers is a function of the projects they are members in. Specifically, eigenvector centrality \( C_E(D^k) \) for a developer \( D^k \) can be expressed as,

\[
(B4) \quad C_E(D^k) = C_E(P^i) x_{ik}
\]

where \( C_E(P^i) \) is the eigenvector centrality of the project \( i \) that the developer \( D^k \) is a member of and strength of the tie between the developer and the project is given by \( x_{ik} \).

Solving for \( C_E(D^k) \) which satisfies the above equation for all nodes in the graph (actors and events or developers and projects) gives the eigenvector centrality for all the nodes. One can solve this system of simultaneous linear equation system by using standard eigenvector-eigenvalue formulation. Specifically, let:

\[
(B5) \quad Xc = \lambda c
\]

where \( X \) is a \( h \times h \) sociomatrix, \( \lambda \) is the largest eigenvalue, and \( c \) is the vector of centrality scores. Thus, the eigenvector centrality for project \( P^i \), is given as (see Faust 1997):
where, $x_{ik} = 1$ if developer $D_k$ is a member of project $P_i$, 0 otherwise.

Similarly, the eigenvector centrality of developer $D_k$, is given by the equation,

$$ C_{E}(D_k) = \frac{1}{\lambda} \sum_{i=1}^{g} C_{E}(P^i) x_{ik} $$

where, $x_{ik} = 1$ if developer $D_k$ is a member of project $P^i$, 0 otherwise. Note that in case a project had more than one project manager, we added the degree, betweenness, and eigenvector centrality measures of the multiple managers to obtain the degree, betweenness, and eigenvector centrality measures for the project manager respectively.

For the illustrative example, the eigenvector centrality for the actor Adam, will therefore be $\frac{1}{\lambda} \sum_{i=1}^{g} C_{E}(P^i) x_{ik} = .512$, where $i = 1$ (Adam). The eigenvector centrality for the other nodes can be calculated in a similar manner and these calculated values are presented in Exhibit B1.

We present a screen capture of the *perl foundry* data in Exhibit B2. This dataset has 108 rows (projects) x 490 columns (developers). We used the software UCINET 6.0 to calculate the network measures from this affiliation matrix.
Exhibit B1: Network Centrality Measures for the Illustrative Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Degree</th>
<th>Betweenness</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deskpro</td>
<td>3</td>
<td>7</td>
<td>.601</td>
</tr>
<tr>
<td>Screenpro</td>
<td>2</td>
<td>4</td>
<td>.372</td>
</tr>
<tr>
<td>Keypro</td>
<td>2</td>
<td>1</td>
<td>.000</td>
</tr>
<tr>
<td>Adam</td>
<td>2</td>
<td>6</td>
<td>.512</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>0</td>
<td>.195</td>
</tr>
<tr>
<td>Chris</td>
<td>1</td>
<td>0</td>
<td>.316</td>
</tr>
<tr>
<td>Jean</td>
<td>1</td>
<td>0</td>
<td>.316</td>
</tr>
<tr>
<td>Joan</td>
<td>1</td>
<td>0</td>
<td>.000</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>0</td>
<td>.000</td>
</tr>
</tbody>
</table>

Exhibit B2: Screen Capture of Actual Data (Affiliation matrix)
References


For the six structural embeddedness variables (i.e., degree, betweenness, and eigenvector centrality for projects and project managers), we use a multivariate normal latent class cluster model, also referred to as a finite mixture model (e.g., McLachlan and Peel 2000; Wedel and Kamakura 2000). Specifically, for the vector \( x \) of the six structural embeddedness constructs \( x_1, x_2, x_3, x_4, x_5, \) and \( x_6 \) respectively we specify the following probability density function:

\[
(C1) \quad f(x | \pi, \theta) = \sum_{c=1}^{C} \pi_c f_c(y | \theta_c)
\]

where \( c \) is the indicator variable for latent cluster such that \( \sum_{c=1}^{C} \pi_c = 1 \) and \( 0 \leq \pi_c \leq 1 \).

We specify the conditional distribution \( f(y_c | \pi_c, \theta_c) \) as a multivariate normal, i.e., for latent cluster \( c \):

\[
(C2) \quad f(y_c | x) = (2\pi)^{-K_c/2} |\Sigma_c|^{-1/2} \exp\left[-\frac{1}{2}(y_c - \mu_c)^T \Sigma_c^{-1} (y_c - \mu_c)\right]
\]

where, \( \mu_c \) is a vector of the means, \( \Sigma_c \) is the covariance matrix, and \( K_c = 6 \) \( \forall c = 1\ldots C \) in our case.

As a firm can belong to only one cluster, we model \( \pi_c \) with a logit formulation (e.g., Gupta and Chintagunta 1994; Kamakura, Wedel, and Agrawal 1994). Specifically:

\[
(C3) \quad \pi_c = \frac{e^{\delta_c}}{\sum_{c=1}^{C} e^{\delta_c}}
\]
where, $\boldsymbol{\delta}_c$ is the cluster-specific constant term. We then standardize Equation (B3) as by assuming that $\boldsymbol{\delta}_c = 0$. (e.g., Gupta and Chintagunta 1994):

**Estimation and Model Selection**

Consistent with literature on latent class cluster analysis (DeSarbo and Cron 1988; Wedel and Kamakura 2000), which relies on finite mixture distribution theory (Everitt and Hand 1981; Titterington, Smith, and Makov 1985), we specify the likelihood function as:

\[
L = \prod_{n=1}^{N} \sum_{c=1}^{C} \pi_{cn} f_{nc} \]

where we specify $f_{nc}$ in Equation C2, $\pi_{cn}$ in Equation C4, and the subscript $n$ denotes the individual unit of analysis (projects in our case) such that the sample has $N$ units in total. We use the Bayes rule to determine the posterior probability that the unit $n$ belongs to cluster $c$:

\[
P[n \in c | \theta_{nc}] = \frac{\pi_{cn} f_{nc}}{\sum_{c=1}^{C} \pi_{cn} f_{nc}}
\]

To obtain the parameter estimates, we use the expectation-maximization (E-M) algorithm to maximize the natural logarithm of the likelihood function specified in Equation C4 (e.g., DeSarbo and Cron 1988). We used 50 randomly selected starting values for each of the models to ensure convergence (Wedel and Kamakura 2000). To obtain the standard errors for the parameters we inverted the Hessian (or the information matrix) for the final parameter estimates.

To manage heterogeneity, we introduced latent clusters, which represent a non-parametric formulation. To determine the number of the latent clusters, we use the Bayesian
information Criteria (BIC; Schwarz 1978) and the Consistent Akaike Information Criteria (CAIC; Bozdogan 1987). Specifically, we compare the model with c point mass with a model with c+1 point mass ∀ c = 1, 2, … till the model fit stops improving. We calculate BIC and CAIC as:

\[
\text{(C6)} \quad BIC = -2 \ast LL + K \ast \ln(N)
\]

\[
\text{(C7)} \quad CAIC = -2 \ast LL + K \ast (1 + \ln(N))
\]

where \(LL\), \(K\), and \(N\) stand for log-likelihood value, number of parameters, and sample size respectively (see Table C1). We also report an entropy measure of separation (ES) to assess the extent of separation of the clusters (Wedel and Kamakura 2000). We calculate ES as (ES is bounded in the range 0 to 1, such that a value closer to 1 indicates good separation of groups or latent clusters):

\[
\text{(C8)} \quad ES = 1 - \frac{\sum_{n=1}^{N} \sum_{c=1}^{C} - p_{nc} \ln(p_{nc})}{N \ln(C)}
\]

where \(p_{nc}\) is the probability of unit \(n\) belonging to cluster \(c\), which we calculate using Bayes rule specified in Equation C5.

**Testing for Heterogeneity**

We seek a test to determine, statistically, the best number of regimes or, more narrowly, whether more than one regime exists within our data. Although the information criteria (BIC and CAIC) are used most frequently in latent class analysis to select the “best” model (e.g., Wedel and Kamakura 2000), these criteria fail to provide a statistical test for model comparison. Thus, based on the information criteria, we see that in our case the two-cluster model outperforms the model with a single cluster and the model with six-clusters seems to
be the most appropriate, thereby suggesting the existence of heterogeneity in structural embeddedness. However, we need a formal test to say that one model is statistically superior to another. To achieve this objective we use the Likelihood Dominance Criterion (LDC) advanced by Pollak and Wales (1991) to statistically compare non-nested models. For two non-nested models involving the same dependent variable, the likelihood dominance criterion (LDC) involves calculating the $L_2 - L_1$, where $L_2$ is the log likelihood for the model with greater number of estimated parameters ($n_2$) and $L_1$ is the likelihood for model with lower number of estimated parameters ($n_1$). The LDC prefers $H_1$ to $H_2$ if

$L_2 - L_1 < \left[ \chi^2(n_2 + 1) - \chi^2(n_1 + 1) \right]/2$ ; $H_2$ to $H_1$ if $L_2 - L_1 > \left[ \chi^2(n_2 - n_1 + 1) - \chi^2(1) \right]/2$ ; and is indecisive if both of these criteria are not satisfied (Pollak and Wales 1991). For the comparison of the homogenous (one-cluster) solution with the two-cluster solution, we obtain $L_2 - L_1 = 700.37$. The two critical values ($p < .01$) are: (1) $\left[ \chi^2(n_2 + 1) - \chi^2(n_1 + 1) \right]/2 = 8.98$ and (2) $\left[ \chi^2(n_2 - n_1 + 1) - \chi^2(1) \right]/2 = 11.25$. Thus as $L_2 - L_1$ exceeds the second critical value, we can say that in our case, the two-cluster solution outperforms the homogenous single cluster solution ($p < .01$). A comparison of the single-cluster solution with the optimal six-cluster solution in our case gave us $L_2 - L_1 = 979.80$. The two critical values ($p < .01$) are: (1) $\left[ \chi^2(n_2 + 1) - \chi^2(n_1 + 1) \right]/2 = 41.14$ and (2) $\left[ \chi^2(n_2 - n_1 + 1) - \chi^2(1) \right]/2 = 44.50$. Here again, the six-cluster solution outperforms the homogenous single cluster solution ($p < .01$).
### Table C1: Model Selection in Latent Class Cluster Analysis

<table>
<thead>
<tr>
<th>Number of Clusters</th>
<th>Log-Likelihood</th>
<th>Number of Parameters</th>
<th>BIC</th>
<th>CAIC</th>
<th>Size of Smallest Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2526.590</td>
<td>12</td>
<td>5109.37</td>
<td>5121.37</td>
<td>100.00%</td>
</tr>
<tr>
<td>2</td>
<td>-1826.219</td>
<td>25</td>
<td>3769.49</td>
<td>3794.49</td>
<td>12.38%</td>
</tr>
<tr>
<td>3</td>
<td>-1720.993</td>
<td>38</td>
<td>3619.91</td>
<td>3657.91</td>
<td>7.65%</td>
</tr>
<tr>
<td>4</td>
<td>-1651.361</td>
<td>51</td>
<td>3541.51</td>
<td>3592.51</td>
<td>3.94%</td>
</tr>
<tr>
<td>5</td>
<td>-1581.942</td>
<td>64</td>
<td>3463.54</td>
<td>3527.54</td>
<td>2.94%</td>
</tr>
<tr>
<td>6 (Optimal Solution)</td>
<td>-1546.791</td>
<td>77</td>
<td>3454.11</td>
<td>3531.11</td>
<td>2.91%</td>
</tr>
<tr>
<td>7</td>
<td>-1512.31</td>
<td>90</td>
<td>3446.02</td>
<td>3536.02</td>
<td>1.05%</td>
</tr>
<tr>
<td>8</td>
<td>-1473.33</td>
<td>103</td>
<td><strong>3428.93</strong></td>
<td>3531.93</td>
<td>1.03% (three such clusters)</td>
</tr>
</tbody>
</table>

**NOTES:** BIC and CAIC stand for Bayesian Information Criteria and Consistent Akaike Information Criteria respectively. The optimal solution is shown by minimum BIC and CAIC (shown in bold). BIC suggests an eight-cluster solution, while CAIC suggests a five-cluster solution to be optimal. As the number of parameters for the nine-cluster solution exceeds the sample size (108), we were not able to estimate this solution. A comparison of solutions with five to eight clusters revealed that solutions with seven and eight clusters involved splitting up the smallest cluster until it reached a size of 1 project. Such splitting makes sense, as large projects can be very different from each other in terms of their centrality and those of their project managers. However, from our perspective, the splitting of large clusters adds statistical explanatory power but does not change the managerial story. Thus, we use the six-cluster solution to be the optimal solution.
References


APPENDIX D
LATENT CLASS REGRESSION ANALYSIS

To account for the possibility of multiple regimes, i.e., multiple possible relationships between explanatory variables (X) and the dependent variables (Y), we use latent class regression analysis. Specifically for R possible regimes we specify these relationships as:

\[(D1) \quad Y_p = \sum_{r=1}^{R} [X_p \beta_r + \epsilon_r] \]

where, p scripts the projects, and \( \beta_r \) is the regime specific regression coefficient. To estimate this multi-regime model, we use a finite mixture of linear regressions (DeSarbo and Cron 1988; Hutchinson, Kamakura, and Lynch 2000; Wedel and Kamakura 2000) that are based on finite mixture distribution theory (Everitt and Hand 1981; Titterington, Smith, and Makov 1985). We use Bayes rule to calculate the posterior probability for regime \( r \) to be representative of project \( p \), that is:

\[(D2) \quad P[p \in r | Y_p] = \frac{\delta_{r|p} L_{pr}}{\sum_{r=1}^{R} \delta_{r|p} L_{pr}} \]

where, \( \delta_{r|p} \) denotes the prior probability that project \( p \) belong to regime \( r \) and \( L_{pr} \) is the likelihood value that the project \( p \) belongs to regime \( r \). Consistent with extant literature (Dayton and MacReady 1988; Gupta and Chintagunta 1994; Kamakura, Wedel, and Agrawal 1994) and the latent class cluster analysis (Appendix C), we use the logit formulation to specify the prior probabilities as:

\[(D3) \quad \delta_{r|p} = \frac{e^{\kappa_r}}{\sum_{r=1}^{R} e^{\kappa_r}} \]
where, we estimate $\kappa_r$ for each regime. Again, we can standardize Equation (B3) by assuming that $\kappa_r = 1$ (e.g., Gupta and Chintagunta 1994):

Thus, we treat the last group as the base and need to only estimate $R-1$ parameters. Note that consistent with the concomitant variable approach (e.g., Dayton and MacReady 1988; Kamakura et al. 1994), we use firm size (as indicated by sales) as a determinant of regime membership. The likelihood for each regime is specified based on the standard normal density as:

$$(D4) \quad L_{plr} = \phi^*(\varepsilon_r)$$

where, $\phi^*(.)$ is the standardized normal density function and $\varepsilon_r$ is residual error such that $\varepsilon_r \sim N(0, \sigma_r)$. Thus, the likelihood function can be written as:

$$(D5) \quad L = \prod_{p=1}^{P} \sum_{r=1}^{R} \delta_{p|r} L_{plr}$$

where, we have $P$ projects in our dataset and estimate the relationship for $R$ regimes. We maximize the natural logarithm of Equation D5 to obtain parameter estimates for $R$ regime solution. Specifically, we use the E-M algorithm with 50 random start values to obtain the parameter estimates. Here again we rely on BIC and CAIC values and entropy of separation (ES) to determine the appropriate number of regimes and the validity of the optimal solution respectively (e.g., Wedel and Kamakura 2000). Table D1 provides support for a two-regime solution in our case, reinforced by the kernel density plots in Figure D1 signaling bimodality and the existence of multiple regimes.
Table D1: Model Selection in Latent Class Regression

<table>
<thead>
<tr>
<th>Number of Regimes</th>
<th>Log-Likelihood</th>
<th>Number of Parameters</th>
<th>BIC</th>
<th>CAIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-242.329</td>
<td>12</td>
<td>540.844</td>
<td>552.844</td>
</tr>
<tr>
<td>2</td>
<td>-211.679</td>
<td>25</td>
<td>540.412</td>
<td>565.412</td>
</tr>
<tr>
<td>3</td>
<td>-197.252</td>
<td>38</td>
<td>572.425</td>
<td>610.425</td>
</tr>
</tbody>
</table>

Model for Downloads

<table>
<thead>
<tr>
<th>Number of Regimes</th>
<th>Log-Likelihood</th>
<th>Number of Parameters</th>
<th>BIC</th>
<th>CAIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-290.857</td>
<td>13</td>
<td>642.582</td>
<td>655.582</td>
</tr>
<tr>
<td>2</td>
<td>-196.261</td>
<td>27</td>
<td>518.939</td>
<td>545.939</td>
</tr>
<tr>
<td>3</td>
<td>-166.376</td>
<td>41</td>
<td>524.720</td>
<td>565.720</td>
</tr>
</tbody>
</table>

NOTES: The optimal solution for each criterion for each dependent variable is shown in bold. In the case of CVS commits, BIC suggests that a two-regime solution is optimal, while CAIC suggests that a one-regime solution is optimal. Given the bimodality in the kernel density plot (Panel A in Figure 4), we are inclined to consider the two-regime solution to be optimal. The high entropy of separation for the two-regime solution (ES = .98), further bolsters confidence in the two-regime solution. In the case of number of downloads, both BIC and CAIC suggest that a two-regime solution is optimal. The two regimes also seem to be well separated (ES = .99).
Figure D1: Kernel Density Plots for Logarithm of Number of CVS Commits and Downloads

Panel B: Number of CVS Commits

Panel B: Number of Downloads

NOTES: The kernel density plots for the logarithm of both the number of CVS commits and number of downloads shows bimodality, signaling the possibility of two regimes.
References


