Success in Online Production Systems: A Longitudinal Analysis of the Socio-Technical Duality of Development Projects

Claudia Mueller-Birn[1], Marcelo Cataldo[1],
Patrick Wagstrom[2], James D. Herbsleb[1]

November 2010
CMU-ISR-10-129

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

[1] Institute for Software Research, School of Computer Science, Carnegie Mellon University
[2] IBM Research

ABSTRACT

Online production systems represent a new and innovative approach for producing information goods. However, the success of such endeavors depends on a careful interrelationship between their social and technical dimensions. In this paper, we explore how various aspects of those dimensions impact the success of online production systems. We collected data from the open source community GNOME and we used the inclusion of a product into an official release as indicator for the success of a project. Our analyses revealed that structural characteristics of the individual project's communication and task dependency (coordination needs) networks, the position of individuals in the overall ecosystem communication network as well as the technical structure of the product, are all significantly associated with project success. Our novel results represent an important step in understanding the success drivers of online production systems as well as a starting point for reshaping traditional models for producing information goods typically used in corporate settings.

Keywords: Online production systems success, Product structure, Collaboration networks, Coordination requirements networks, Dynamic network analysis.

INTRODUCTION

The World Wide Web (WWW) has not only shaped social relationships but has also changed processes of production information-age goods such as content and software. New and innovative approaches have emerged for people to share their ideas, their experiences, their knowledge, and collaboratively produce information goods in so-called online production systems (cp. [3], [26]).

Online production systems have a number of properties that differentiate them from traditional ways of producing goods. They have relatively low barriers to entry [27] and, consequently, contributions are mostly done by volunteers who cooperate in a geographically distributed and decentralized way forming open collaborative communities. Online production systems rely on key infrastructure enablers such as the Internet and a host of communication, coordination and collaboration tools to produce information goods such as an online encyclopedia, open source software, a shared taxonomy (e.g., del.icio.us), or compiled news and information offerings (e.g., Slashdot). These systems also share similarities with traditional production settings found in companies. The development of any type of product involves a close relationship between the technical and the social dimension. The technical dimension refers to properties of the developed product and the social dimension refers to the collection of organizational norms, governance and coordination mechanisms that allow people to produce the desired object.

Research in product development and software engineering has examined the relationship between the social and technical dimension and project success in traditional corporate environments. A key finding is that the ability to identify the relevant task dependencies and the use of appropriate coordination mechanisms to manage those dependencies are major drivers for project success (e.g., [40], [8], [9], [6] [20]). We know that certain open collaborative communities such as Wikipedia are able to produce high quality outcomes by utilizing particular coordination mechanisms such as interactions on talk pages and a relatively small number of editors [25]. In other types of online production systems such as software development projects, the landscape of coordination needs that emerge from technical properties of the product and the ability of the project participants to identify those task dependencies might be a major challenge. In this paper, we study drivers of success in online production systems using data collected from the open source community GNOME covering about 8 years of activity. Because of the availability of artifacts that contain an explicit and detailed record of dependencies in the product, open source software development projects represent a unique opportunity to study the role of task dependencies in project success.

Our study examines the relative impact of factors that capture the communication and coordination patterns of different projects and the whole ecosystem as well as factors that consider task dependencies as determined by the technical dimension. We find that structural characteristics of the individual project's communication and coordination needs networks, the position of individuals in the overall ecosystem communication network as well as the technical structure of the product, are all significantly associated with project success.

The remainder of the paper is organized as follows: First, we discuss the socio-technical nature of online production systems and its implications for project success. Second, we describe our research setting and the measures as well as statistical models used in our analysis. We conclude with a discussion of our results, the limitations of our study and its implications for future work.

SUCCESS DRIVERS IN ONLINE PRODUCTION SYSTEMS

Researchers in the area of online communities gave an important impetus for our understanding which factors impact success. One of the earliest contributions differentiates usability and sociability as main areas to evaluate success of virtual community sites [34]. The former considers human-technology interactions (e.g., information design, navigation, and access) whereas the latter is especially concerned with human-human interactions by developing policies and practices that are socially acceptable and practicable. Sociability determinants are of particular interest because they give us insights on how social interactions might influence success. Past work [35] suggests that factors such as the number of participants who communicate, the number of exchanged messages, interactivity, and reciprocity are relevant success drivers. In recent years, increased attention has been given to the structure of those exchanges of information and interactions and its role for the success of online communities. However, most of that line of work has been descriptive and qualitative in nature [22]. Adopting a social network perspective allows for describing social interactions in online communities [43]. Hinds and Lee [22] proposed a set of hypotheses that link certain structural properties of communication networks with online community success, but no empirical evaluation of how these properties impact success in those settings has been reported. This leads to our first research question:

> *RQ 1. Are structural properties of project communication networks associated with success of online production systems?*

Members of a project do not operate in isolation from other projects in the ecosystem; rather their interactions can impact other members' motivation to participate, future decisions and coordination processes as well as solutions to technical problems. For example, recent work has shown that personal relationships play an important role in project's contributions. Hahn and colleagues [21] showed that participation in projects within the Linux development community tended to increase when people had close personal relationships with other project members or when they had good experience based on past collaborations. Therefore, the position of project members within a community's social network might have an important impact on its outcomes [32]. For example, individuals that bridge core members of a community with those in the periphery tend to have higher innovative output [11]. Those results lead to our second research question:

> *RQ 2. Does the structural position of projects' members within the communication network of the whole ecosystem have an impact on the success of online production systems?*

The production of information goods depends on a collection of well-coordinated activities in order to be successful. Coordination, defined as the act of managing dependencies between activities [33], has been shown to be an important driver for high quality outcomes in certain online production systems. Kittur and Kraut [25] explored the relationship between coordination activities and the quality of articles in Wikipedia. The authors found that the scale of coordination needs was a major challenge and when contributors were able to adopt the appropriate coordination mechanism, the quality of articles improved. For example, in case of a high number of contributors who edited an article, article quality was positively impacted if a core group of contributors existed and they were responsible for most of the editorial work. In addition to an issue of scale, recent work in geographically distributed development has shown that the structure of those coordination needs could impact project outcomes such as quality as well [10]. Taking a social network perspective, coordination needs among members of an online production system can be thought of a network where the patterns and the scale of the coordination needs are captured by structural properties of such a network. That leads to the following research question:

> *RQ 3. How do structural properties of the coordination needs network influence the success of online production systems?*

The technical dimension of products being constructed or developed in an online production system can also be a major success driver. One key element of the technical dimension is the structure of a product. Researches have long argued that certain product structures such as modular designs have a number of benefits over other structures because they make task dependencies more manageable (e.g., [1],[31]). For example, modular structures allow projects to experiment with different design alternatives and those designs evolve more efficiently resulting in higher returns for projects or firms. Work in the area of technical dependencies has shown that product structures have also important impact on traditional project outcomes such as performance and quality. Highly coupled parts of a system tend to be more difficult to change and maintain [29] and are more likely to exhibit lower levels of quality [37], [7]. However, recent work has shown, for instance, that the relationship of the structural properties of a system and project outcomes is more intricate. For example, certain types of technical relationships among a product's constituent parts matter more than others in terms of project performance [9] or product quality [7]. Furthermore, certain structural patterns of those different types of technical relationships impact product quality differently [10]. Those results suggest the following research question:

> *RQ 4. Are network properties of the technical structures of the system under development associated with success of online production systems?*

METHOD

In order to answer our research questions an appropriate representative of an open production system is needed that allows us to tackle the two dimensions of an online production system. An open source community is ideal for examining our research questions because the various repositories of data used in open source software development represent a rich source of data that capture the social as well as the technical dimension of an online production system. For example, the social dimension of an open source project is represented by the communication on the mailing lists. The coordination processes can be understood from the task and defect tracking systems and the technical structure of the software systems can be described by using data from the source code management system.

We examined our research questions using data collected from the open source community GNOME that is focused on the development of software for a graphical user interface system. The remainder of the section is organized as follows. First, we present a description of GNOME. Then, we describe the various measures collected and we conclude the section with a description of the statistical model used in our analyses.

Description of the Research Setting

GNOME is an open source software development community that has developed a graphical user interface for operating systems such as Linux. It was initiated in 1997. Over the following decade, volunteers around the world contributed to this software in order to create a freely available desktop application. The constellation of software development projects is organized within a federated system [41]. Core programmers such as those responsible for certain key components, manage their projects and oversee the whole software development process within these projects. For instance, core programmers produce releases and integrate changes submitted from other contributors through so-called “patches”. Projects adjust their development roadmaps and goals to the overall GNOME plans and strategy. In particular, such alignment means that projects which become part of GNOME releases or that are linked to projects that are part of GNOME releases have to follow specific development guidelines and milestones. In order to get a project hosted by GNOME’s technical infrastructure, projects have to meet specific requirements such as using an open source license, employing GNOME core technologies, and having at least one public release. Hence, the GNOME source code repository consists of various projects either included into official releases or not. We are using this fact by considering both “kinds” of projects in our data set, projects that are included into an official release and projects that are not included.

Our data set (cf. [41]) comprises the mailing list repository and the source code repository of GNOME. We extended this data set by release information and by the actual source code for each of the selected projects. The data covered a period of about 8 years of activity from November 1997 until July 2005. At the end of this period, the source code repository contained 735 *modules* but the size, the number of developers, and the age of these modules varied extensively, for example because a module can be a single project or can be part of a larger project. We define a *project* as a single piece of software in GNOME that may comprise a number of different modules but at least one. Additionally, a project should have at least one dedicated mailing list.

Our first step was to identify all projects that have been part of the official GNOME release, i.e. distribution, in the period under investigation. For this, we retrieved each distribution with the corresponding source code and compared each contained module with available project data in our data set. If a project started separately but was later merged with another project we treated both projects as a single project.

Only for 18 officially released projects user mailing lists were available and we were able to identify 9 unreleased projects in the data set with the same set of available information. All other modules in our dataset had no dedicated mailing list. Of these 27 projects, all had general mailing lists, and only 3 projects also had developer-specific mailing lists. In 2 of these 3 projects, more developers participated in the general mailing list than in the developer-specific list. In all three cases, developers participated in more threads on the general mailing list than on the developer-specific list. Across all the general mailing lists, the developers had very significant participation, initiating 28% of the e-mails, and participating in 51% of the threads. For these reasons, and to maintain comparability in the project data sets, we decided to use the general mailing lists for our study.

The projects exhibit a different level of maturity in terms of their lifetime (e.g., 5 projects have less than 5 years of data), their number of contributors (e.g., number of committers ranges between 10 and 976), their intensity of communication (e.g., number of mail threads is between 11 and 15,000), and their product sizes (e.g., the source code has between 50,000 and 14,000,000 lines of code). One reason for these differences can be seen in the type of the products. There are for example projects that deal with user tools such as an email client (Evolution) or web browser (Epiphany) but also projects that are part of the underlying technical infrastructure, such as the GTK+ library which is the primary library used to construct user interfaces in GNOME.

During the second step, we determined an appropriate unit of analysis for our longitudinal data (all defined temporal networks are introduced in the following section). The goal was to determine a time interval that exhibits as much information as possible (or that loses as little information as possible, respectively). We tested the changes of the temporal development of the density and average shortest path length in the communication and coordination needs network with intervals of 15, 30, 45 and 60 days [28]. Finally, we constructed a stream of longitudinal data for each project aggregating the data on a 30-days basis because this interval balanced best the information gain versus the information loss.

Description of the Measures

In the following section, we describe our dependent and independent variables as well as the set of control factors we used in our analyses. Besides these measures, we discuss our defined temporal networks and how we constructed them based on our data set.

Measure of Success

Success in online production systems as well as in traditional development projects is a widely discussed concept and its past operationalizations have varied significantly. For example in Wikipedia, the success of an article can be seen as its quality [25]. An article has to meet certain requirements in order to get assigned into a six-level quality system, ranging from "stub" (almost no content) to "featured-article" (best quality). Researchers have used measures such as number of edits and unique editors [44] to explore the interrelatedness of article quality and the creation process. Measures of success in online communities and, in particular, open source projects have typically revolved around quantifications of volume related to number of contributors or participants or number of access to the particular project's product or outcome [15], [24]. Also, the product development literature has considered "success" of a development project across a wide range of conceptualizations such as market performance of the product, project cycle time, efficiency of the development process and product quality [12], [17], [38].

In the case of a community such as GNOME, projects that become part of the community distribution have been vetted as having a certain level of quality or value because they have to meet the aforementioned technical quality requirements. The criteria for inclusion in an official release include 1) continuous project activity; 2) a judgment of quality based on fixing all high-priority bugs, all major features completed, and all bugs fixed at a reasonable rate; 3) technical characteristics of the project such as using GNOME technology, accessibility, usability, and internationalization; and 4) the developers work well with the community [18]. Therefore, we expect that projects that are part of a release exhibit differences in their technical structure as well as in their communication and coordination structure compared to projects that has not been included.

Our measure of success, "released", is a dichotomous variable that indicates whether a project is included in the GNOME distribution during a specific time interval (indicated with a 1) or not. Such an inclusion followed different patterns. For example, a group of 10 projects were never included in the distribution. There were also persistent inclusions where projects were always part of all nine releases and finally, there were temporary inclusion where projects were included in at least one release but the inclusion was not persistent.

Independent Measures

In order to address our research questions, we collected a series of measures which can be grouped into four categories each one corresponding to a particular research question.

Measuring the Project's Communication Structure

We constructed a collection of communication networks for each project from the exchange of emails as stored in the projects' mailing lists. Each network covered a 30-day period of activity. Each network was constructed as follows: For each email, we added the sender into the communication network. If there was a response to the email, we added an edge in the network between the sender and the respondent. Based on regular expressions we identified such a response to an email by the same subject. If a third person responded to the same email thread all three senders were connected by an edge. We considered the communication networks as undirected graphs because emails in a mailing list are sent to the list and not to a single person.

In order to assess the structural properties of the communication networks, we computed a set of network-level measures on each communication network for each project at each time interval. Density is a measure between 0 and 1 that captures the ratio of existing number of edges in a network to the maximal possible number of edges in a network. A completely connected network has a density of 1. Clustering coefficient is based on the average local clustering coefficient for the entire network proposed by Watts and Strogatz [42]. The local clustering coefficient of a node is the average probability that a pair of neighbors of this node is connected to each other. We use this measure on a network level in order to capture the average strength of connectedness between all nodes in a network. This measure is standardized on the network size and ranges therefore between 0 and 1. The average shortest path length measures the average number of edges that connects any pair of nodes in the network. If the network consists of components, only the lengths of existing paths are considered. Combined, these last two measures can be used to determine whether a network follows a small-world structure or not. High levels of clustering coefficients and lower levels of average shortest path characterize a small-world structure [42]. The number of components captures the number of subsets of nodes where there is no path connecting those subsets. The measure was adjusted by the number of isolates in the network. This procedure allowed us to precisely measure only disconnected subsets containing at least two actors. Finally, we also computed the network degree centralization which captures the difference between the node with the highest degree centrality and each other node in the network. It is a measure between 0 and 1 and it is the higher the more central one node is compared to all other nodes in the network.

Measuring the Interplay between Project and Ecosystem

It has long been argued that the structural position of individuals within a communication or information flow network has a range of individual and collective benefits [5], [13]. For example, the structural holes argument [5] suggests that individuals, groups or organizations could gain from information access and flow control by bridging disconnected sets of individuals or entities. Recent work on innovation has argued that it is not just about the position of an individual within a network, but the position of an individual within networks with particular structural properties. Cattani and Ferriani [11] suggested that individuals that bridge the core and periphery in a network are more likely to have higher innovated capabilities.

Motivated by the above-mentioned line of work, we constructed communication networks of the whole ecosystem by aggregating the project-level communication networks into one. Since the communication network of the projects capture a 30-day period of interaction activity, the community wide networks also covered a 30-day period of activity. We then ran a core-periphery (CP) analysis on each network using the continuous model proposed by Borgatti and Everett [4]. The model assigns a core-periphery score to each node in the network. The closer the score is to 1, the higher the likelihood that the node is part of the network's core. On average, each monthly network had a correlation with an idealized core-periphery model of 0.520 (s.d.=0.081, min=0.261, max=0.764).

In order to examine the interplay between the interaction patterns of the ecosystem and each project's member structural position within the communication networks, we computed the following measures for each 30-day network. Mean CP score captures the average core-periphery score of the projects' members. The CP score dispersion captures the standard deviation of the core-periphery scores of the project members. Higher values of this measure suggest that projects' members tend to be more dispersed between the core and the periphery of the network. Finally, we computed the proportion in core measure that captures the ratio of project members that have core-periphery scores that are higher than the network-wide average. The higher the value of this measure, the greater the proportion of project members that are positioned in the core of the network.

Table 1. Descriptive statistics

Variable	Mean	Std. Dev.	Min	Max	Skewness	Kurtosis
Released	0.361	0.480	0	1	0.581	1.337
Platform module	0.186	0.389	0	1	1.607	3.584
Productivity tool	0.046	0.211	0	1	4.291	19.405
Other module	0.439	0.496	0	1	0.245	1.061
Committers	11.830	11.204	0	73	1.451	5.063
Changed LOCs	51,041.700	139,041.300	0	2,018,314	6.633	63.511
Density (communication)	0.118	0.196	0	1	0.568	2.027
Num. components (communication)	1.820	3.847	0	47	2.721	11.087
Density (coordination)	0.746	0.311	0	1	-0.674	1.732
Num. components (coordination)	0.881	0.467	0	15	1.743	6.648
Density (technical-logical)	0.196	0.146	0	0.835	1.294	3.658
Num. components (technical-logical)	24.230	31.41	0	267	2.277	9.531
Density (technical-syntactic)	0.018	0.018	0.0007	0.117	-0.182	2.108
Num. components (technical-syntactic)	1.373	2.638	1	21	2.667	11.105
Proportion in core	0.051	0.131	0	1.000	4.856	30.035
CP score dispersion	0.894	0.183	0	0.994	-4.122	12.017

Measuring the Project's Coordination Needs Structure

A second set of factors that might impact success in online production systems relates to the structure of dependencies among the development tasks, which determine a set of coordination requirements among the project's members. We used the socio-technical congruence method proposed by Cataldo and colleagues [8, 9] to compute coordination needs networks for each project. The congruence approach combines information about how developers were involved in the development tasks performed over a particular period of time with information about how the software system was modified over the same period of time to determine the degree to which each developer is interdependent with the others. In our case, we considered the set of development tasks that were completed within a 30-day period of time. Then, identifying the set of individuals involved in those tasks and constructed a Task Assignment matrix. A task is defined as a commit of a developer to at least one file in the source code. The set of source code files modified as part of the development tasks completed in each 30-day period allows us to construct a Task Dependency matrix. Following Cataldo and colleagues, we computed the coordination needs networks through the following matrix multiplication: (Task Assignment x Task Dependency) x Transpose(Task Assignment). In order to assess the impact of the structural properties of the coordination needs networks on project success we computed the same network-level measures described in the previous section: density, clustering coefficient, average shortest path length, number of components, and network degree centralization. We consider the coordination needs networks as undirected graphs.

Measures the Project's Technical Structure

In open source projects, the technical structure of a software system can have major implications on the success of projects [1], [2]. A traditional way of extracting the structure of a software system is to examine its source code and identify programming language constructs that represent points of relationships between different entities of the system such as source code files, modules or components. We refer to this structural representation as syntactic. Researchers have also suggested other structural representations of a software system. Recent work by Cataldo and colleagues [9], [7], [6] has shown the value of considering interrelationships among software entities based on the patterns of modification of those entities as part of the development tasks. We refer to this structural representation as logical since it allows the identification of relationships that are semantic in nature, which is typically not easily captured by the syntactic approach. For our analysis, we constructed 30-day technical networks. In the case of the syntactic relationships, we extracted them using a tool called Doxygen[1] that examines the source code tree of each project at the time of each release of the GNOME distribution. In the case of the logical dependencies, we constructed them by extracting the set of source code files that were modified as part of development tasks performed during the period of time between two releases of the GNOME distribution. In order to assess the impact of the structural properties of the syntactic and logical dependency networks on project success we computed the same network-level measures described above: density, clustering coefficient, average shortest path length, number of components, and network degree centralization. We consider the technical networks as undirected graphs.

Additional Control Factors

Besides the already introduced measures, we defined six additional control factors that are not related to the defined networks. The variable Mails indicates the number of mails submitted to the mailing lists of the projects during a period of 30-days. The variable Senders indicates the number of distinct individuals that submitted emails to the mailing lists of the project during the defined time interval. The variable Committers indicates the number of distinct developers that made changes to the project's source code during the time interval. The variable Commits represents the number of changes made to the project's source code during 30 days. Files represent the number of source code files modified in the project the time interval. Changed lines of code (LOCs) indicates the number of lines of code added, deleted and modified in the source code of a project during the defined period of time. Furthermore, the software implemented in each project differs in their technical properties as well as in their role within the overall "vision" of the GNOME community. Therefore, we used the existing classification system in GNOME and measured project type as a categorical variable distinguishing projects that were "desktop modules", "platform modules", "productivity tools" and "other" [19]. The categorical measure was converted for inclusion in the regression models to a set of three dichotomous variables: (a) platform module indicates with a 1 if the project is a platform module, (b) productivity tool indicates with a 1 if the project is a productivity tools like an email or a word processor and (c) other module indicates with a 1 if the project type fell into the "other" category. When all three variables are zero, we have a project that is categorized as a desktop module.

[1] More information are available at http://www.stack.nl/~dimitri/doxygen/

Description of the Statistical Model
As it is typical in this type of longitudinal analysis, variations of the outcome may be the result of the idiosyncratic aspect of each project as well time (e.g., variation may be due to the current development state of a project). Traditional regression models are not able to properly account for those variations. Therefore, we utilized a multi-level modeling approach, which allows variation at several levels within the model (for a more complete overview of multi-level modeling of longitudinal data see [39]). When developing a multilevel model, initial effects are specified, and then random effects may be applied to multiple variables for a given stream of longitudinal data. For example, the impact of time may vary across projects—a multi-level model allows for an analysis that accounts for this. In our data we have a stream of data for each project and this allows variation of both the intercept and the influence of time (in number of months since the beginning of existence of the GNOME community). In this way, we account for the fact that projects may be different, for example an email client is more likely to have widespread appeal than a library that does processing on a niche file format. Since our dependent variable is dichotomous, we used a multi-level logistic regression model to evaluate the effects of the various factors on project success.

We constructed several multi-level logistic regression models to examine our research questions. Following a standard hierarchical modeling approach, we started our analysis with a baseline model that contains only the control factors. In subsequent models, we added the various aforementioned measures. To enhance interpretability, we report the odds ratios associated with each measure instead of reporting the regression coefficients. Odds ratios larger than 1 indicate a positive relationship between the independent and dependent variables whereas an odds ratio less than 1 indicates a negative relationship. For example, if a dichotomous independent variable has an odds ratio of 2 in a logistic model, such value of the odds ratio means that the probability of having the outcome variable be 1 doubles when the independent variable changes from 0 to 1. Mathematically, the odds ratio is the exponent of the logistic regression coefficient.

RESULTS
This section presents the results of our empirical analyses. We first present a series of preliminary analyses performed on our data followed by a discussion of the various models that examined the role of product success in terms of inclusion into an official release.

Preliminary Analyses
The first analysis consists in performing a collinearity diagnostic among all our independent variables. We did a variance inflation factors (VIF) analysis and as recommended [30] we removed from our models those measures with VIF values above 5. We had high levels of correlation (above 0.7) between the measures of the syntactic representation of the software structure and the logical representation; consequently, we constructed two different regression models for those measures. Table 1 summarizes the descriptive statistics of the set of measures that we included in the regression analyses. Those variables with skewed distributions were log-transformed to be used in subsequent analyses.

Baseline Models
We report the results of our regression analyses in Table 2. Following a traditional hierarchical approach, we started with a baseline model that includes only the control factors. Subsequent models include the various independent measures related to our research questions. Model I in Table 2 is the null model with a log likelihood of -1214:7. We use such model as the baseline comparison to determine the

amount of deviance explained the various groups of independent factors. Model II is the baseline model that includes only the control factors. We observe that the nature of the project had a significant impact on the likelihood of inclusion into an official GNOME release. Those projects categorized as platform modules were about 3.5 times more likely to be included in a GNOME release than those projects categorized as desktop modules. On the other hand, productivity tools or other types of projects were 2.9 times and 6.8 times less likely to be included in an official release than desktop modules projects. These results reflect the module strategy in GNOME. While platform modules are necessary to develop additional functionality for the GNOME project, desktop modules are primarily necessary to ensure the "basic" functionality of the GNOME Desktop. Furthermore, platform modules have to meet certain requirements on code stability; for example, existing interfaces should be stable at least for a period of two releases. Consistent with past results [24], the amount of participation, which is here the number of contributors (committers), increases the likelihood of project success.

Table 2. Odds ratios from multi-level logistic regressions assessing the impact on release inclusion

	Model I	Model II	Model III	Model IV	Model V	Model VI	Model VII
Platform module		3.498*	4.648*	4.752**	4.342**	6.088**	5.253**
Productivity tool		0.346*	0.006+	0.307**	0.338**	0.328**	0.252**
Other module		0.146*	0.003*	0.199**	0.182**	0.143**	0.154**
Committers–log		5.279*	6.858*	5.570**	4.809**	3.704**	4.022**
Changed LOCs–log		0.789	0.439*	0.867*	0.837*	0.872*	0.921+
Density (communication)–log			0.004*	0.347**	0.347**	0.126**	0.211**
Num. components (communication)–log			0.041*	0.924	0.920	0.812	0.923
Proportion in core–log				0.031**	0.041**	0.077**	0.035**
CP score dispersion–log				2.829**	2.712**	2.095*	1.956*
Density (coordination)–log					2.014+	2.132+	1.876
Num. components (coordination)–log					2.761*	2.468*	2.304*
Density (technical-logical)–log						3.704*	
Num. components (technical-logical)–log						1.921**	
Density (technical-syntactic)–log							459.4**
Num. components (technical-syntactic)–log							1.761*
Log Likelihood	-1214.7	-904.1	-869.2	-850.7	-845.8	-792.4	-814.8
Deviance Explained		25.6%	28.4%	29.9%	30.4%	34.8%	32.9%
($+p < 0.1$, $*p < 0.05$, $**p < 0.01$)							

The Role of Communication Structures

Model 3 in Table 2 examines our first research question. The model introduces measures that capture the properties of communication network for the various projects. We observe that higher levels of density and a higher number of components in the communication network reduce the likelihood of project success (odd ratios lower than 1). These results suggest that successful projects benefit from interaction patterns that are able to disseminate information to most of the project participants such as developers and users while minimizing redundant interconnections. On the other hand, highly clustered interaction patterns among project participants are more likely to be associated with unsuccessful projects. These results differ from past work suggesting that core-periphery (e.g., [16]) or hierarchical communication structures (e.g., [23]) tend to benefit distributed workgroups the most. One possible explanation for this difference is the evolving nature of open source development where developers and users interact and share information about a wide range of topics from what are the current tasks and problems, what future features should be included in the system under development, how to install and use the product in machines that where not tested before. Such patterns might be indicators that a wider range of users and developers are participating in communication and information sharing activities rather than those actions being concentrated around a few individuals.

The Interplay between Project and Ecosystem

Research question 2 refers to the relationship between communication patterns on the project-level and the online-production-system-level. In model 4, we observe that the higher the number of project members that are part of the core of the system-wide communication network, the lower the likelihood of a project being included in the GNOME distribution. One possible explanation for this result is that community members who are part of the core are more likely to be active in more than one project in the GNOME community, consequently, those individuals need to divide their attention and effort among more projects than those individuals that are position away from the community-wide network core. In addition, higher levels of dispersion in the core-periphery scores among project members are associated with a higher likelihood of project success. These results suggest that project success depends on its members occupying different structural positions within the network as a mechanism to balance the benefits and limitations of belonging solely to the core or the periphery. Based on the web-based open source code repository Sourceforge, Xu and colleagues revealed that small projects mainly consist of core developers and projects leaders (core) whereas large projects had many co-developers and active users (periphery) [45]. In our study, larger projects are mainly those that has been released; therefore, based on [45] it can be assumed that larger projects exhibits more diverse structural positions and this leads to their final success.

The Role of Coordination Needs Structures

Model 5 in Table 2 reports the results of the analysis that assessed the impact of the structural properties of coordination needs networks on the likelihood of a project being included in an official GNOME release (RQ3). We observe that the higher the density and the higher the number of components in the coordination needs network, the higher the likelihood of project success. These results suggest that when tasks dependencies are partitioned among separate clusters of highly interdependent sets of individuals, projects are more likely to succeed by becoming part of an official GNOME distribution. Our findings are consistent with the long-standing argument in the modular design literature (e.g., [1], [36], [31]) that modular organizational designs are beneficial.

The Role of the Technical Structure

Finally, Models 6 and 7 in Table 2 report the results of the analysis that assessed the impact of the structural properties of technical dependency networks on the likelihood of a project being included in the GNOME distribution (RQ 4). In both models, we observe that higher levels of density as well as higher numbers of components in the structure of technical dependencies of a software system increase the likelihood of project success. Combined, these results suggest that modular technical structures (those with independent clusters of highly interdependent parts) are an important success driver for online production systems. Our analyses show that model 6 which considers logical technical dependencies fits the data better (34.8% of explained deviance) than model 7, which includes the syntactic technical dependencies (32.9% of explained deviance). These results add to past results suggesting that the logical structure of software systems is not just a major driver of coordination needs [8] or quality outcomes [7] in corporate environments but also an important driver of overall project success in online production systems as well.

Additional Analyses

The various projects included in our dataset, as previously discussed, have different patterns of inclusion in the GNOME distribution. For example, there were persistent inclusion (projects are included in all nine releases), temporary inclusion (projects are included in at least one release but not the subsequent), and late but continued inclusion (projects are included in a release and in all subsequent). Then, one potential concern is that the results reported are driven primarily by systematic characteristics of a subset of the projects. In order to examine the robustness of our results, we perform a number of additional analyses. We first computed a set of dummy variables that categorized the projects into for groups: (a) projects that were never included in the GNOME distribution, (b) those projects that were always part of the GNOME releases, (c) those projects that were included and remained part of the GNOME distribution until the end of the period covered by our data and finally (d) those projects that were included at some point but later were removed from the GNOME release. We then included the dummy variables as control factors in our regression models and the results were consistent with those reported in Table 2.

We also examine the differences in the independent variables across those for groups using Kruskal Wallis tests. The results show no significant differences across groups with the exemption of the variables committers and number of changed lines of code (X^2 =518.4, $p<0.001$ and X^2 =265.9, $p<0.001$, respectively). Projects that were never part of a GNOME release had significantly less number of committers and lower levels of change in the source code than the other types of projects. Despite these specific differences across groups, we think that these additional analyses provide evidence of the robustness of the results reported in this paper.

DISCUSSION

An online production system comprises a social dimension of an open collaborative community where geographically distributed individuals collaborate to produce information goods and the technical dimension of such product. In this study, we investigated how three aspects of those dimensions (communication networks, coordination needs networks and product structure) impact success of a project. We found that each aspect has an independent role as success driver. Those results represent an important contribution to the online community and online production systems literature because they further our understanding of existing modes of actions and their influence on the success of those environments. Our analyses of the role of the communication structure within projects of an ecosystem revealed that communication patterns that interconnect most projects members (e.g., few components in the network) and are not densely clustered increase the likelihood of a project being successful. A possible explanation for such findings is that successful projects might exhibit a continuously active core group that is able to integrate all members of the project or the developed software. On the other hand, the likelihood of project success is increased when both the coordination needs among project members and the technical structure of the product consists of collections of independent and highly interconnected clusters of actors and product elements, respectively. Finally, our results showed that successful projects tend to have communication patterns within the context of the entire online production system that involve interactions with core as well as peripheral members of the community.

Limitations

Our study suffers from several limitations worth mentioning. First, our analyses considered only a subset of available projects in GNOME. However, the included projects do represent completely available data in terms of communication in the community, development activity and software source code. Second, we assumed that each change to the source code identified in the version control system was associated with the developer who committed such change. However, GNOME follows code ownership processes where a core developer might commit changes made by a third person. We think that the number of those instances is rather small number although we do not have the necessary data to confirm it. Third, the consideration of a single online production system raises concerns for external validity. Our analyses and results could certainly be generalized to other open source software development communities. In order to generalize our results to other forms of online production systems such as Wikipedia might require considering, for example, a more adequate definition of what product structure constitutes in an encyclopedia.

Future Research Directions

Our research benefited from existing data in the GNOME mailing list and source code repositories. Examining additional, potentially different patterns of communication, information sharing and coordination represent a valuable future research endeavor. For example, in Crowston and Howison [14], the structure of teams and existing coordination mechanisms are studied based on data from task and defect tracking systems. Using those data might provide valuable insight on coordinative actions and processes that are used in open source communities. Considering a broader range of data sources would allow us to investigate the distribution of user participation across different collaborative technologies might represent additional success drivers in online production systems. Furthering our understanding of existing success factors in online production systems, would also lead to future research on a maturity model for online production systems that might help potential participants to estimate the reliability of a certain online production system. Similarly, such a model would allow “community owner” to actively influence processes in their community in order to ensure or obtain success. Additionally, further investigations by using online production systems with different product structures would show existing requirements for coordination mechanisms in online production systems. Successful patterns of the interrelatedness of the technical structure of the product and the social structure of the community could fertilize traditional models for producing information goods. Such an application can even transcend corporate boundaries and new questions emerge about the consistency of our results in corporate settings and possible changes in our results because of the dominance of corporate participation in online production systems (e.g., Eclipse community).

REFERENCES

[1] C. Y. Baldwin and K. B. Clark. Design Rules: The Power of Modularity Volume 1. MIT Press, 1999.

[2] M. Bass, M. Bass, V. Mikulovic, L. Bass, J. Herbsleb and M. Cataldo. Architectural misalignment: An experience report. Conf. on Softw. Archit., pages 17–26, 2007.

[3] Y. Benkler. Coase’s penguin, or linux and the nature of the firm. The Yale Law J., 112(3):369–446, 2002.

[4] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. Social Networks, 21(4):375–395, 2000.

[5] 5. R. S. Burt. Structural holes: The social structure of competition. Harvard University Press, 1992.

[6] M. Cataldo and J. D. Herbsleb. Coordination breakdowns and their impact on development productivity and software failures. Technical Report ISR-10-104, CMU, 2010.

[7] M. Cataldo and S. Nambiar. The impact of geographic distribution and the nature of technical coupling on the quality of global software development projects. J. of Softw. Maint. and Evol., 2010.

[8] M. Cataldo, P. A., Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In Proc. of CSCW, pages 353–362, 2006.

[9] M. Cataldo, J. D. Herbsleb, and K. M. Carley. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In Proc. of ESEM, pages 2–11, 2008.

[10] M. Cataldo, A. Mockus, J. A. Roberts. and J. D. Herbsleb. Software dependencies, work dependencies, and their impact on failures. IEEE Trans on Softw Engin, 99:864–878, 2009.

[11] G. Cattani and S. Ferriani. A Core/Periphery Perspective on Individual Creative Performance. Org. Sci., 19(6):824–844, 2008.

[12] K. Clark and T. Fujimoto. Product Development Performance. Harvard Business School Press, 1991.

[13] J. C. Coleman. Social capital in the creation of human capital. Americ. J. of Socio., 94:95–120, 1988.

[14] K. Crowston and J. Howison. The social structure of open source software development teams. In Proc. Of ICIS, 2003.

[15] K. Crowston, H. Annabi, J. Howison and C. Masango. Towards a portfolio of FLOSS project success measures. In Workshop on Open Source Software Engineering (ICSE 2004), pages 29–33, 2004.

[16] J. N. Cummings and R. Cross. Structural properties of work groups and their consequences for performance. Social Networks, 25(3):197 – 210, 2003.

[17] K. Eisenhardt and B. Tabrizi. Accelerating adaptive processes: Product innovation in the global industry. Administrative Science Quarterly, 40(1):84–110, 1995.

[18] GNOME. How to propose modules for inclusion. http://live.gnome.org/ReleasePlanning/ModuleProposing, 2009.

[19] GNOME. GIT source code repository. http://git.gnome.org/browse, 2010.

[20] B. Gokpinar, W. J. Hopp, and S. M. R. Iravani. The Impact of Misalignment of Organizational Structure and Product Architecture on Quality in Complex Product Development. Manage. Sci., 56(3):468–484, 2010.

[21] J. Hahn, Moon, Y. Jae and C. Zhang. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. Inform. Sys. Res., 19(3):369–391, 2008.

[22] D. Hinds and R. M. Lee. Social network structure as a critical success condition for virtual communities. Proc. of HICSS, page 323, 2008.

[23] P. Hinds and C. McGrath. Structures that work: social structure, work structure and coordination ease in geographically distributed teams. In Proc. of CSCW, pages 343–352, 2006.

[24] A. Iriberri and G. Leroy. A life-cycle perspective on online community success. ACM Comput. Surv., 41(2):1–29, 2009.

[25] A. Kittur and R. E. Kraut. Harnessing the wisdom of crowds in wikipedia: quality through coordination. In Proc. of CSCW, pages 37–46, 2008.

[26] A. Kittur and R. E. Kraut. Beyond Wikipedia: coordination and conflict in online production groups. In Proc. of CSCW, pages 215–224, 2010.

[27] A. Kittur, B. Lee, R. E. Kraut. Coordination in collective intelligence: the role of team structure and task interdependence. In Proc. of CHI, pages 1495–1504, 2009.

[28] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. Science, 311(88), 2006.

[29] M. S. Krishnan, C. H. Kriebel, S. Kekre and T. Mukhopadhyay. An Empirical Analysis of Productivity and Quality in Software Products. Manage. Sci., 46(6):745–759, 2000.

[30] M. Kutner, C. Nachtsheim, J. Neter, and W. Li. Applied Linear Regression Models. Oxford, 2004.

[31] R. N. Langlois. Modularity in technology and organization. Journal of Economic Behavior and Organization, 49:19–37, 2002.

[32] J. Lazar and J. Preece. Social considerations in online communities: Usability, sociability, and success factors. In H. van Oostendorp, editor, Cognition in the Digital World. Lawrence Erlbaum, 2002.

[33] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. ACM Comput. Surv., 26(1):87–119, 1994.

[34] J. Preece. Online Communities: Designing Usability, Supporting Sociability. John Wiley & Son, 2000.

[35] J. Preece. Sociability and usability in online communities: determining and measuring success. Behav. & Inform. Techn., 20(5):347–356, 2001.

[36] M. A. Schilling and H. K. Steensma. The use of modular organizational forms: an industry-level analysis. Aca. of Manage., 44(8):1149–1168, 2001.

[37] R. Selby and V. Basili. Analyzing error-prone system structure. Trans. on Softw. Engin., 17(2):141 –152, 1991.

[38] R. Sethi. New product quality and product development teams. Journal of Marketing, 64:1–14, 2000.

[39] J. D. Singer and J. B. Willett. Applied longitudinal data analysis : modeling change and event occurrence. Oxford Press, 2003.

[40] M. E. Sosa, S. D. Eppinger, and C. M Rowles. The misalignment of product architecture and organizational structure in complex product development. Manage. Sci., 50(12):1674–1689, 2004.

[41] P. A. Wagstrom. Vertical Communication in Open Software Engineering Communities. PhD thesis, CMU, SCS, Pittsburgh, 2009.

[42] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. Nature, 393:440–442, 1998.

[43] B. Wellman, J. Salaff , D. Dimitrova, L. Garton, M. Gulia and C. Haythornthwaite. Computer networks as social networks: Collaborative work, telework, and virtual community. Annu. Rev. Sociol., 22:213–238, 1996.

[44] D. M. Wilkinson and B. A. Huberman. Assessing the value of cooperation in Wikipedia. CoRR, 2007.

[45] J. Xu, Y. Gao, S. Christley and G. Madey. A topological analysis of the open source software development community. In Proceedings of HICSS, page 198.1, 2005.