6-2018

How does developer interaction relate to software quality? an examination of product development data

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

How does developer interaction relate to software quality? An examination of product development data

Subhajit Datta

Singapore University of Technology and Design, 8 Somapah Road, Singapore, 487372, Singapore

Published in Empirical Software Engineering, 2018, 23, 1153-1187.
DOI: 10.1007/s10664-017-9534-0

Abstract

Industrial software systems are being increasingly developed by large and distributed teams. Tools like collaborative development environments (CDE) are used to facilitate interaction between members of such teams, with the expectation that social factors around the interaction would facilitate team functioning. In this paper, we first identify typically social characteristics of interaction in a software development team: reachability, connection, association, and clustering. We then examine how these factors relate to the quality of software produced by a team, in terms of the number of defects, through an empirical study of 70+ teams, involving 900+ developers in total, spread across 30+ locations and 19 time-zones, working on 40,000+ units of work in the multi-version development of a major industrial product, spreading across more than five years. After controlling for known factors affecting large scale distributed development such as dependency, system age, developer expertise and experience, geographic dispersion, socio-technical congruence, and the number of files changed, we find statistically significant effects of connection and clustering on software quality. Higher levels of intra-team connection are found to relate to higher defect count, whereas more clustering relates to fewer defects. We examine the implications of these results for individual developers, project managers, and organizations.

Keywords

Software teams, Interaction, Connection, Clustering, Software quality, Defect count, Jazz

1 Introduction

Computing and its effects dominate our lives today, and the technological civilization is deeply reliant on software (Stroustrup 2007). Many of the early software systems, especially the ones that are most visible and impactful, were developed by small teams of programmers (Cringely 1996). These systems were often conceived and developed by one or two individuals, who then gathered around them a small group of enthusiasts sharing their vision. But industrial software systems are built differently now. Many more developers than a select few are involved, and usually members of large development teams are separated by distance and time-zones. This has lead to the recent recognition of social aspects that influence software development by large teams of developers, beyond factors affecting individual productivity that had long been identified (Weinberg 1971).

Any examination of social factors influencing large scale software development needs to happen in the context of the myriad interactions between individual members of the development team. The social processes around these interactions are sought to be facilitated by collaborative development environments (CDE) such as Jazz.[1] Although the use of CDEs in software development is relatively recent, social phenomena surrounding interacting individuals in other fields has been studied for several decades (Coleman 1990). These studies have established key parameters that characterize interaction. However, the relationship between developer interaction and software quality remains far from a settled question (Brooks 2010).

Facilitating the sharing of information in a team is recognized to be of consequence to how the team performs (Wagstrom et al. 2010). But there are many nuances to the issue; how much and how easily team members can transfer and absorb information, and the effects of such interaction offers diverse evidence, often contradictory (Olson and Olson 2000; Wolf et al. 2008). As increasingly more complex software systems are being built by larger and more distributed teams, organizations require clearer directions on the characteristics of developer interaction that help better functioning of teams, as well as those that hinder.

In this paper we present results from an empirical study to discern how interaction between members of a distributed development team, relate to the quality of the software produced by the team. Our results can inform individual developers on ways of deriving more value from their peer interactions. To managers, our results can be useful for delegating and tracking responsibilities in a team. At the organizational level, conclusions from this study can help resource allocation and team assembly.

The **research contributions** from this study are:

- We identify a set of typically social characteristics of interactive human enterprise from existing literature, and examine how these characteristics relate to outcomes in large scale software development.
- We present results from a multi-project study analyzing how interaction between team members are associated with software quality.
- We identify statistically significant effects of information sharing and clustering between team members on the number of defects in the team's work products.
- On the basis of empirical evidence, we recommend a set of best practices for the governance of projects involving large distributed teams.

[1] https://jazz.net/

The paper is organized as follows: In the next few sections we present a background of our research, motivate our research question and develop a set of hypotheses. Subsequently, the study setting is described, statistical models developed and the results presented and discussed. We next identify threats to the validity of our results, and outline related work. The paper ends with a summary of our results, and conclusions.

2 Background

The premise behind facilitating interaction in large software development teams is that better interaction will relate to higher quality of output from the team (Humphrey 1999). Interaction can be viewed from several perspectives; we will now identify characteristics of interaction between individuals and their relevance to our context.

One of the earliest and most well known empirical studies on a large scale social phenomenon was conducted by Travers and Milgram (Travers and Milgram 1969), as they explored the "small-world problem" formulated by Milgram (Milgram 1967). The results reported by Travers and Milgram helped establish the widely recognized notion of *six degrees of separation*. In recent times, the idea that any two randomly chosen individuals in a large network are only few intermediaries apart has been further explored in studies such as (Watts and Strogatz 1998; Watts 1999), and an algorithmic perspective developed in (Kleinberg 2000). The *average separation* of a network indicates how many intermediaries separate two individual vertices (or nodes) of the network, on average. Average separation reflects on the **reachability** of individuals in a network. In a software development ecosystem, reachability points to how accessible developers are to one another.

While average separation indicates how far apart two randomly chosen individuals are in a network, the *average degree* is a measure of the network's interconnectedness. In a seminal paper, Feld explored how individuals in a network perceive their positions differently with respect to one another and how such perceptions influence their interactive responses (Feld 1991). In a software development scenario, average degree is the mean number of peers that developers in a team are connected to. By capturing the extent of **connection** between team members, the average degree points to the level of contact between developers in large and distributed teams.

Average degree and average separation capture important social characteristics of a group of interacting individuals. But how should we characterize "social" networks? Newman has established that social networks differ from other networks in two important ways – assortative mixing and non-trivial clustering (Newman and Park 2003).

Assortative mixing in networks indicates the tendency for vertices "to be connected to other vertices that are like (or unlike) them in some way" (Newman 2002). Our intuitive sense for *homophily* – the propensity of individuals to associate with *similar* others – is captured in the notion of assortative mixing. A large software team usually has developers with diverse skills, experience, and responsibilities. This diversity is also often reflected in the number of connections to other developers. Individuals with many connections are usually connected to individuals who have many connections themselves and vice-versa. Assortative mixing captures this phenomenon of how team members associate with one another, and is thus a measure of **association** levels in a team.

Clustering is the tendency for clusters or groups of closely connected individuals to emerge in networks; Newman has underscored how relatively high levels of clustering is an essential characteristic of networked individuals sharing similar interests and objectives (Newman 2002). The presence of notable clustering is a sign that social processes rather

than merely random linkages underpin a network (Albert and Barabasi 2002). These processes are often driven by the dynamics of clustering of individuals towards a common goal.

In light of the above discussion, we identify our research question as:

In distributed software development, how does interaction between team members relate to software quality?

In the next section, we derive a set of hypotheses from this research question, which will be empirically validated in our study setting.

3 Motivating the hypotheses

3.1 Context

The research question introduced in the previous section seeks to understand the relation between developer interaction and software quality. As we have already seen, interaction characteristics have various aspects; and each of these aspects needs to be taken into account to fully investigate the effects of interaction. To develop our hypotheses, we use the following social metrics, in the context they were introduced earlier:

- Average separation to reflect on reachability of individuals in a social network.
- Average degree to capture the levels of connectedness of individuals in a social network.
- Assortativity to point to the extent of association among individuals in a social network.
- Clustering coefficient to represent the scale of clustering among individuals in a social network.

Additionally, we take the number of defects in the work products of a team to reflect on the quality of the software produced by the team (Koru and Liu 2005; Bird et al. 2009a; Zimmermann and Nagappan 2009) (see Section 5 for more details).

The social networks studied in this paper are constructed from instances of developers co-commenting on units of work in a software development ecosystem. Vertices of the networks are developers, and two developers are joined by an undirected link if both the developers have commented on at least one common unit of work. Formal definitions of the network and the metrics, along with explanations of how they are calculated are presented in Sections 4 and 5. We now distill our research question into the following hypotheses, each of which addresses a specific aspect of interaction in our context.

3.2 Distilling the research question

- Many of today's software development teams have members who are at different locations, seldom meeting one another face to face. Functioning of such teams is largely reliant on how easily accessible developers are to one another for defining and fulfilling shared responsibilities. This reaching out is important for leveraging a project's ecosystem of implicit knowledge and expertise, something Booch has called "tribal memory" (Booch 2008). As team members separated by time-zones often share small overlap of working hours, the scope of synchronous communication – by phone calls or chat – is limited. Members of such teams need to utilize the team's network of interactions to reach out to one another. In general, developers need to be able to reach one another easily for discussion around project tasks. Lack of such access will impact knowledge sharing and is likely to be detrimental to the quality of the team's deliverable. For a

distributed development team, average separation reflects on the extent of reachability. Accordingly, our first hypothesis is framed as - **H1: Higher average separation of team members is related to lower number of defects.** The corresponding null hypothesis is that there is no relationship between average separation and number of defects.

– How well connected team members are to one another plays an important role in the percolation of information in a team. Developers having more information sharing channels can be expected to be better grounded in the current state of the project. In large software projects, needs of many stakeholders have to be recognized and addressed. So a team in which information flows more easily is likely to produce better quality software; we thus arrive at our second hypotheses - **H2: Higher average degree of team members is related to lower number of defects.** The corresponding null hypothesis is that there is no relationship between average degree and number of defects.
– In a truly "jelled" team, members together produce more effective results than the sum of individual contributions (DeMarco and Lister 1987). Association between members facilitate better team integration or jelling. Thus teams whose members are more closely associated with one another can be expected to produce better quality software, leading to our third hypothesis - **H3: Higher assortativity amongst team members is related to lower number of defects.** The corresponding null hypothesis is that there is no relationship between assortativity and number of defects.
– In large scale software development, collective tasks and responsibilities lead developers to cluster together. Such clustering indicates areas of closer interaction in a development team. As interaction is widely perceived to facilitate better team functioning (Herbsleb and Mockus 2003; Cataldo et al. 2006) we posit that software development teams with higher clustering can be expected to produce better quality software. Thus we frame our fourth hypotheses as - **H4: Higher average clustering coefficient of team members is related to lower number of defects.** The corresponding null hypothesis is that there is no relationship between average clustering coefficient and number of defects.

We next describe the study setting where these hypotheses will be examined.

4 Study setting

4.1 Context

To test the above hypotheses we need data from a cohort of software development teams. The Jazz platform mentioned earlier is listed as an "... initiative for improving collaboration across the software & systems lifecycle".[2] Jazz offers an integrated environment for defining, tracking, and evaluating development activities. All information that developers exchange are automatically recorded in the Jazz repository.

In this paper we study development data from three major *releases* of a major industrial product developed on the Jazz platform. A set of *iterations* leads to a release, and each release produces a *version* of the product. Within a release, developers are segregated into

[2] https://jazz.net/

team areas. A team area on the Jazz platform represents a group of developers focused on delivering a particular line of functionality in a particular release. Each member of a team area has a specific *role*, such as contributor, component lead, tester, build master, member of project management committee etc. Members of a team area can be based in different geographies. Each developer in a team area owns *work items* – uniquely identifiable units of work that are used to assign and track activities on the Jazz platform. Developers exchange *comments* around work items; commenting is the prescribed and predominant vehicle of work related developer communication on the Jazz platform.

In the *Eclipse Way* of collaboration guiding development on the Jazz platform, spontaneous interactions are encouraged between developers in a team area (Frost 2007). The network of interactions within team areas are expected to facilitate the planning and execution of development activities (Herzig and Zeller 2009). Evidently, a Jazz team area represents a useful abstraction of a software development team. In the remainder of the paper, we use "team area" and "team" interchangeably.

In the preceding section we have identified parameters that reflect on the social traits of developer interaction. The computation of reachability, connection, association, and clustering are based on established network metrics, as described in detail in Section 5. The assumption underlying the use of these metrics is that the ecosystem of interacting individuals in a team can be represented by a network, as is widely done in similar studies (Newman 2003b). For our study a *team area network* or simply a *team network* for each team area is constructed in the following way from the data in the Jazz repository: Each work item in a release belongs to a particular team area. For each team area all the work items are collected; if two developers have commented on at least one such work item, they are joined by an edge (undirected link) in the corresponding team network. Thus the vertices (nodes) of the team network are developers, and the edges represent their instances of co-commenting on work item(s). In the unlikely instance of the owner of a work item not having commented on it, (s)he is also joined by edges to all other developers who have commented on that work item. A similar construct for abstracting communication between developers at the release level was used in our earlier work (Datta et al. 2012).

For our analysis, we only consider work items of the type *tasks*, *enhancements*, and *defects*. These are the only types of work items linked to source code changes, as described in *change-sets*. In this study, we are interested in discerning how interactions in a team relate to the quality of the software developed by the team. As discussed in detail in Section 5, we consider the number of defects in the software developed by each team as an indicator of quality, and take it as the dependent variable in our statistical models. Thus to ensure a separation between the influencing factors and the influenced parameter, only task and enhancement work items are used in constructing the team networks. In the following discussion, "defects" will denote defect work items.

We analyzed data across a total time span of five years and nine months of development. Our data set included 116 team areas, including 982 developers in total, working on 46,009 work items. The developers were spread across 35 locations and 19 time-zones, in total. We learned from discussion with team members that Jazz was the management mandated collaborative development environment for the teams from the inception of development of the product we have studied. Thus there was no development activity in our study setting prior to the adoption of Jazz. We removed team areas with minimal interactions; our final analysis was conducted using 71 team networks and 42,228 work items in total. The mean number of developers in team areas was 11.41, with a standard deviation of 8.55. The distribution of team sizes had a skewness of 1.6. Figure 1 shows the distribution of team sizes. Each team network corresponded to a team belonging to one out of the three releases of the

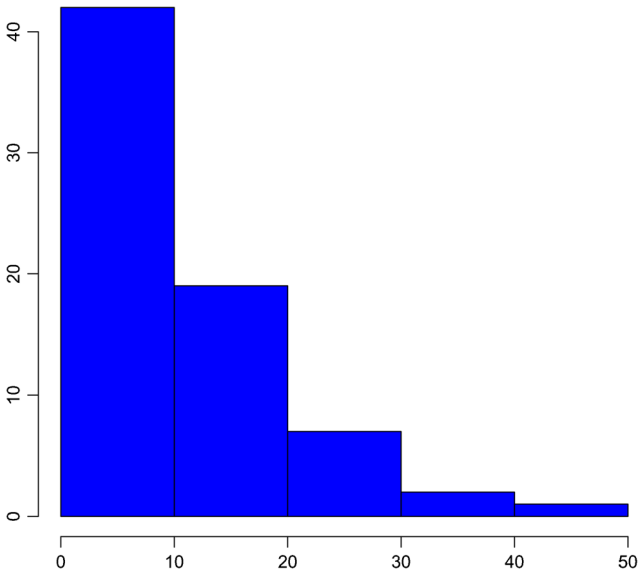

Fig. 1 Distribution of team sizes

product being studied; there was no team which belonged to more than one release. Less than 1% of the developers considered in this study worked on multiple teams, and the maximum number of such multiple teams was two. These developers are present in the team networks of both the teams they worked on. As our statistical models (see Section 5) are at the team level rather than individual developer level, these very few instances of overlapping team members do not influence the results. In a multi-team study as ours, teams need to share a common development context. The teams we are studying functioned in the same Jazz development ecosystem. Thus we believe our study setting is largely insulated from externalities that can introduce confounding variables such as different development methodologies, technologies and communication practices.

In Fig. 2 we outline the major components of our study setting. The project repositories for the teams studied are accessed from the Jazz platform. Relevant project information is extracted from these repositories, and stored in a MySQL[3] database. The team networks as defined earlier are extracted from the data. On the basis of literature survey and contextual information, statistical models are developed to test our hypotheses. Our results are then examined and insights derived in the context of the state of art and practice.

4.2 Team profiles

To help us better interpret the statistical models developed in Section 5, let us try to derive an intuitive understanding of how the teams functioned, by studying the team networks. Degree distribution is a key parameter that characterizes a network (Jackson 2010). To get a sense of how the team networks are oriented, we give the degree distributions of a sample of 18 networks (approximately 25% of the total 71 teams we are studying) in Figs. 3 and 4. We observe that most of the distributions are positively skewed, with a longer right

[3] https://www.mysql.com/

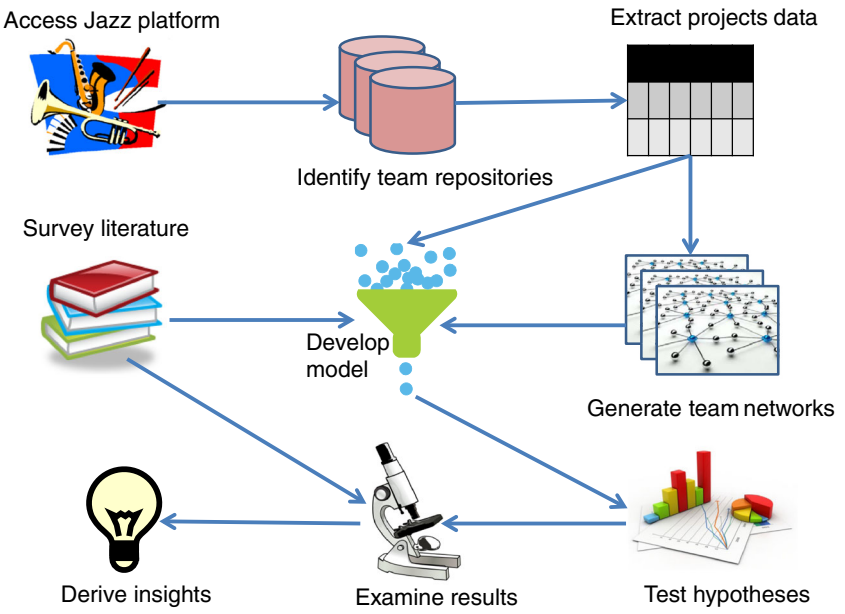

Fig. 2 An outline of the study setting

tail. This is a characteristic common to many social networks (Albert and Barabasi 2002) (Newman and Park 2003). In our context, this implies that *there are relatively few developers in the team who are connected to many others, whereas many developers are connected to few*. For a deeper understanding of the team dynamics, the distributions of the clustering coefficients of these networks are also given in Figs. 5 and 6. We note that the clustering coefficients are mainly skewed to the left. This is an interesting trend in light of the various network models (Dorogovtsev et al. 2002; Ravasz and Barabási 2003). In our study setting, the negatively skewed distributions for the clustering coefficients lead to an interesting observation. It appears that *many developers are engaged in closer clustering, whereas few are relatively more isolated.* These characteristics of developer interaction will be revisited when we discuss our results later in Section 7.

5 Model development

In developing statistical models, our objective is to find out how interaction characteristics relate to software quality. We consider software quality as the *dependent* or *outcome* variable while interaction characteristics as expressed by reachability, connection, association, and clustering are the *independent* or *predictor* variables. To isolate the effects of the independent variables on the dependent variable, we need to account for other relevant effects on the outcome as identified in existing literature – these are the *control* variables. Intuitively, control variables represent the background effects in our study, in light of which we examine how the independent variables relate to the dependent variable. In Table 1 we highlight the relevance of some of the major factors influencing the outcome of large scale software development and then identify our model variables and discuss how they are calculated in the following subsections.

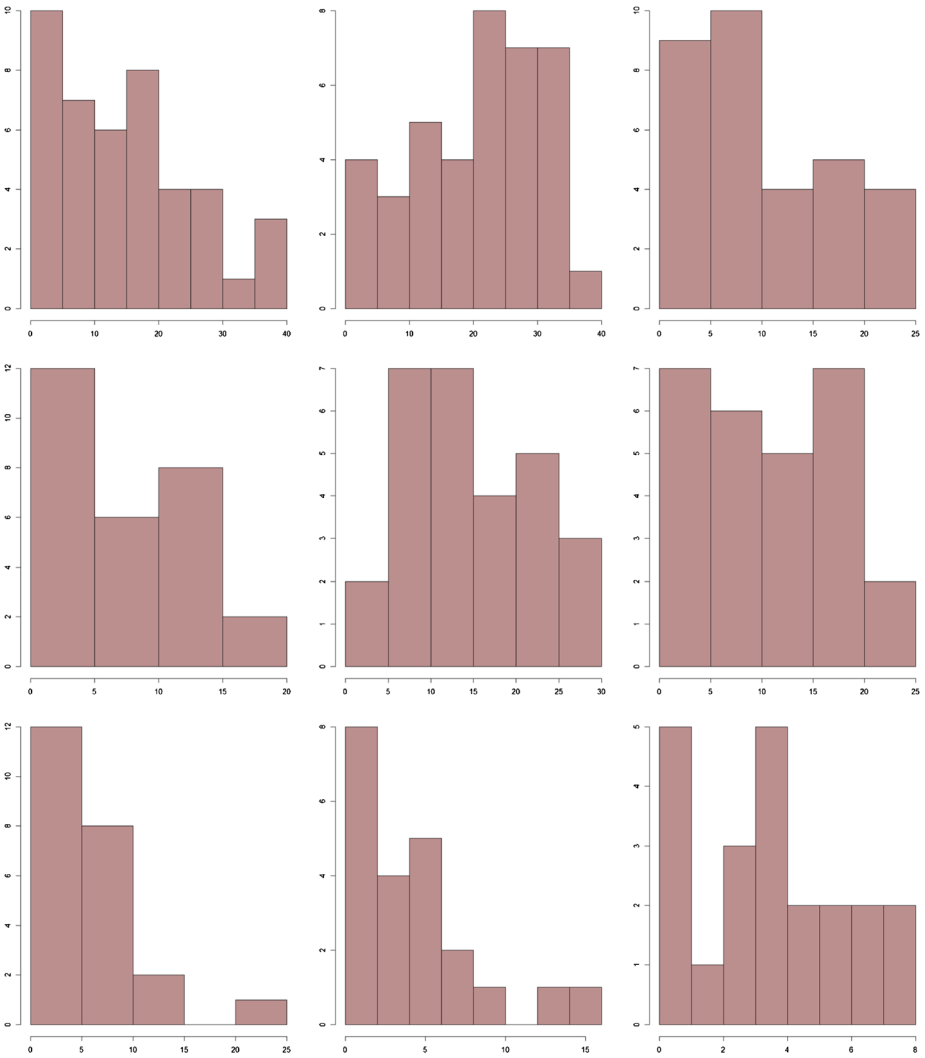

Fig. 3 Degree distributions of teams

5.1 Dependent variable

Software quality can be perceived and measured in a number of ways, as the notion of "quality" has many connotations in industrial products (Datta 1998). While we recognize that counting defects is not the only way to measure quality (Wolf et al. 2009), in many studies the *number of defects* is taken as indication of the quality of a software system (Koru and Liu 2005; Bird et al. 2009a; Zimmermann and Nagappan 2009). Additionally, the number of defects owned by a team is an important parameter for project governance, as it relates to key issues such as customer satisfaction. On these considerations, we take *DefectCountforTeam* as the dependent variable for our models. On the Jazz platform, each

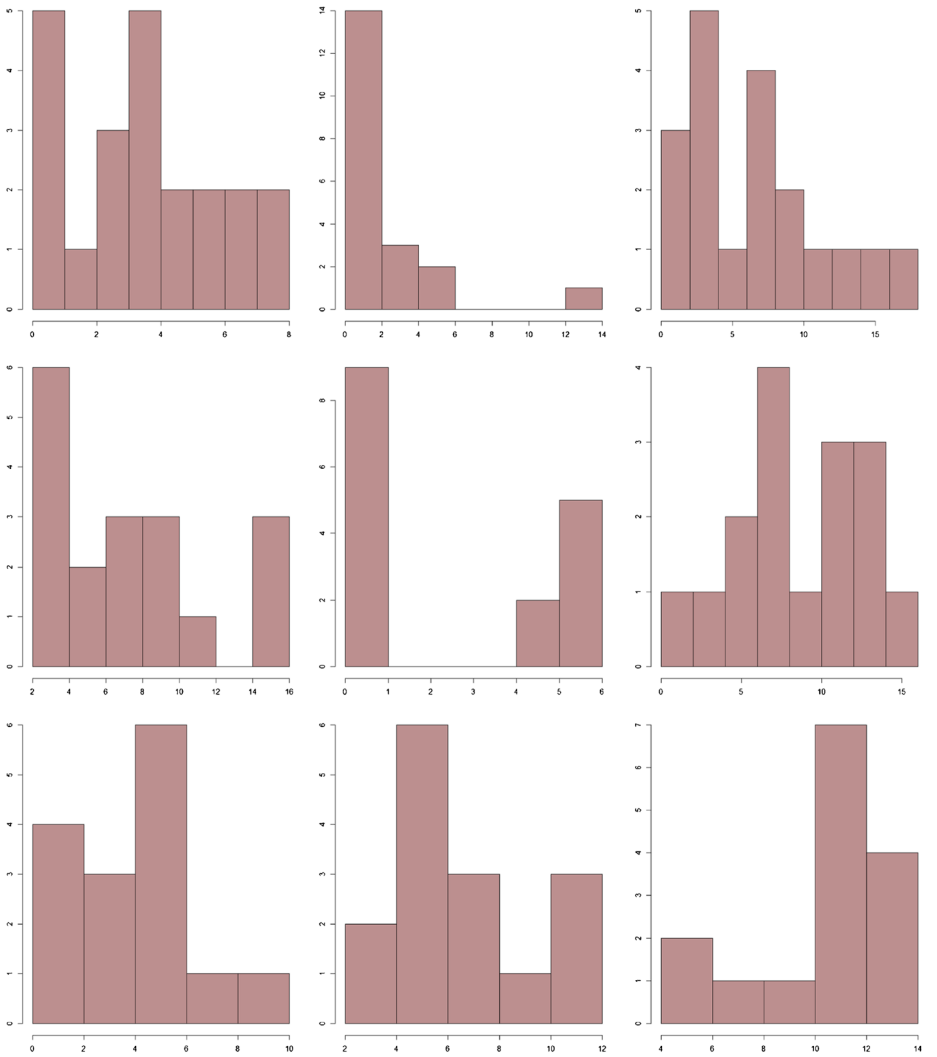

Fig. 4 Degree distributions of teams contd.

defect work item belongs to a team area. Recalling that a team area is specific to a release, we calculate *DefectCountforTeam* as the total number of defect work items for a team area.

5.2 Independent variables

On the basis of the notions developed in Section 2, we denote the independent variables as *Reachability*, *Connection*, *Association*, and *Clustering*. The independent variables are calculated in the following ways:

– *Reachability*: In a network, the ability of two vertices i and j to communicate with one another is influenced by the length of the *shortest* path l_{ij} between them. The average

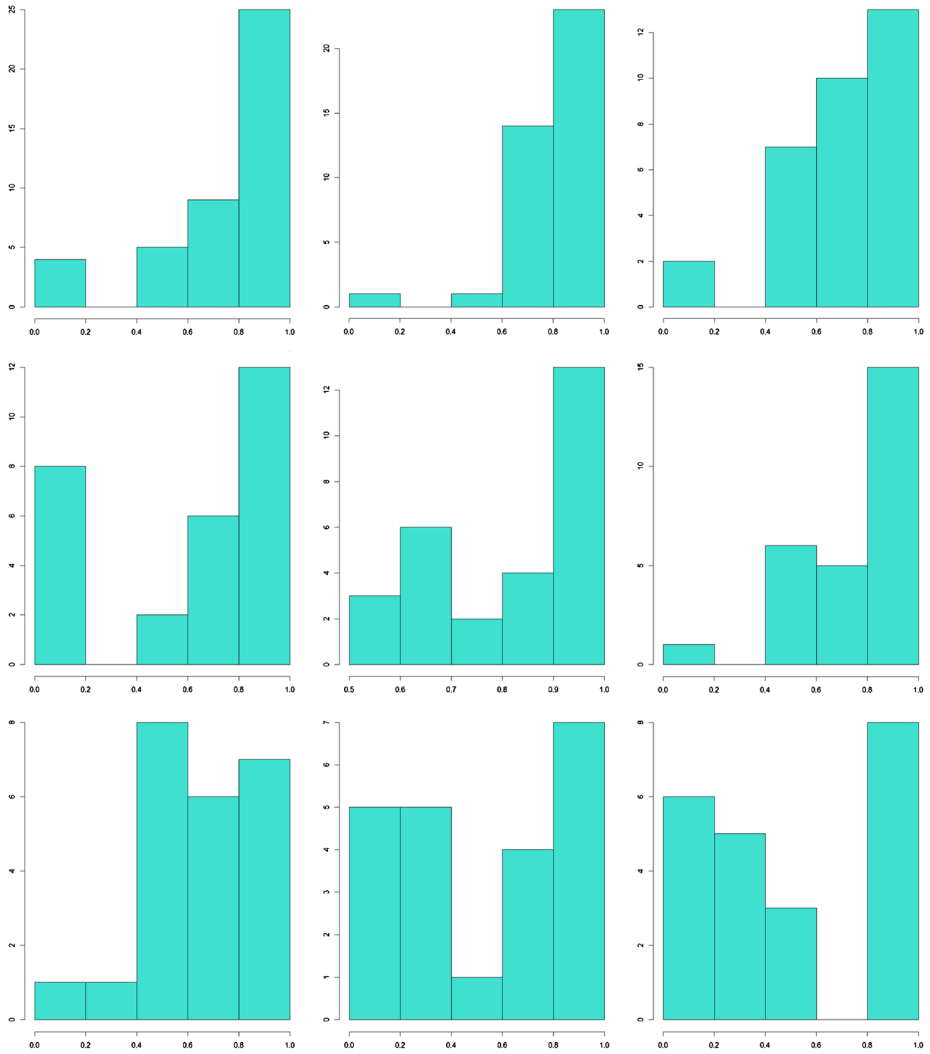

Fig. 5 Clustering coefficient distributions of teams

of l_{ij} over all pairs of vertices is the average separation of the network (Barabasi et al. 2002). We use average separation to measure the *Reachability* of developers in each team.

– *Connection*: The average number of edges per vertex, that is, the average degree is a measure of a network's interconnectedness (Barabasi et al. 2002). For a network with V vertices and E edges, average degree can be calculated as $2 * E / V$. We take the average degree to represent the *Connection* between developers in each team.
– *Association*: According to Newman, "a network is said to show assortative mixing if the nodes in the network that have many connections tend to be connected to other nodes with many connections" (Newman 2002). The author establishes that in a social network, the probability of a target vertex being connected to, depends on the degree

Fig. 6 Clustering coefficient distributions of teams contd.

of the source vertex; and shows how assortative mixing can have notable influence on the behavior of social networks (Newman 2002). As discussed in (Newman 2002) (Newman 2003a), assortative mixing in a network can be calculated by measuring the degree correlation of vertices. Using the method outlined in (Newman 2003b) we calculated the assortativity as the Pearson correlation coefficient between the degrees of pairs of vertices at the ends of the edges of the team network for each team area. This is taken to capture the level of *Association* in a team.

– *Clustering*: In a network, the clustering coefficient (C_v) for a vertex v is defined as follows: If v has a degree of k_v, that is there are k_v vertices directly linked to v, the *maximum* number of edges between these k_v vertices is k_v *choose* 2 or $k_v * (k_v - 1)/2$. If the *actual* number of such edges existing is N_v, then $C_v = 2 * N_v / k_v * (k_v - 1)$.

Table 1 Factors impacting outcomes in software development

	Factors	References	Remarks
1	**Administrative coordination**: Aspects related to project governance and organizational best practices	(Faraj and Sproull 2000)	Since the teams included in this study function on the same Jazz platform, this factor is uniform across all teams and thus has not been considered as a control variable.
2	**Code change**: Modification of code base due to changing requirements and defect resolution	(Cataldo and Nambiar 2009)	This is considered as the *FilesChanged* control variable in our study.
3	**Congruence**: Alignment between communication structure and work structure in a team	(Cataldo et al. 2008; Kwan et al. 2011; Wagstrom et al. 2010)	This factor is considered as the *SocioTechnicalCongruence* control variable in our study.
4	**Dependency**: Inter-connections and/or reliance between different parts of a system	(Cataldo et al. 2006, 2008)	This is considered as the *Dependency* control variable in our study.
5	**Developer experience**: Know-how gained from past instances of developers working on software systems	(Espinosa et al. 2007; Faraj and Sproull 2000; Boh et al. 2007; Cataldo et al. 2006, 2008; Wagstromet al. 2010)	From our discussion with members of the teams we studied, we gathered that more experienced developers usually own more work items, and this practice was widely prevalent in the teams. Accordingly, *Experience* is considered as a control variable in our study.

Table 1 (continued)

	Factors	References	Remarks
6	**Expertise at team level**: Specific set of skills possessed by members of a team	(Faraj and Sproull 2000)	Familiarity of a team's developers with their tasks is taken as a proxy for team expertise, and accordingly *TaskFamiliarity* is considered as a control variable in our study.
7	**Geographic distribution**: Members of a team being distributed across varied geographic locations	(Boh et al. 2007; Cataldo and Nambar 2009; Herbsleb and Mockus 2003; Cataldo et al. 2008, 2009)	This factor is considered in the control variables *SpatialDistribution* and *TemporalDistribution*.
8	**Number of comments**: Number of comments made by developers on units of work they are working on	(Wagstrom et al. 2010)	This factor is subsumed in the construction of the team networks based on co-commenting on work items.
9	**Release**: A unit of software system made available to end users at the end of a development cycle	(Cataldo et al. 2006, 2008)	The dummy variables V_x and V_y control for the release related effects.
10	**Size of tasks**: Metric(s) that reflect on the scope and extent of development activities	(Espinosa et al. 2007; Boh et al. 2007; Herbsleb and Mockus 2003; Cataldo et al. 2006, 2008, 2009)	We assumed that work items as defined on the Jazz platform are atomic units of work of similar scope and granularity, on the basis of our discussions with members of the teams we studied. We have recognized this assumption as a threat to validity in Section 9.

Table 1 (continued)

	Factors	References	Remarks
			Additionally, among the control variables considered in this study, *ActivitySpan* accounts for varying workloads across teams and *FilesChanged* accounts for modifications to code base. These variables can also serve as proxies for size of tasks.
11	**Size of teams**: Number of personnel engaged in units of development	(Espinosa et al. 2007; Boh et al. 2007; Herbsleb and Mockus 2003; Wagstrom et al. 2010; Cataldo et al. 2009; Bird et al. 2009a)	This factor has been considered as the control variable *TeamSize*.
12	**Software methods**: Specific methodologies followed in the development and management of software systems	(Faraj and Sproull 2000; Cataldo et al. 2009)	As all the teams functioned on the Jazz platform following the same development methodology, this factor is uniform across all teams and thus has not been considered as a control variable.
13	**System age**: Elapsed time from the inception of a system to the current time.	(Espinosa et al. 2007; Herbsleb and Mockus 2003; Wagstrom et al. 2010; Grewal et al. 2006)	This has been considered as the control variable *Age* in our study.
14	**Task familiarity**: Familiarity of developers to particular tasks that are being worked on	(Espinosa et al. 2007)	This has been considered as the *TaskFamiliarity* control variable.

Table 1 (continued)

	Factors	References	Remarks
15	**Task priority and severity**: Level of importance/criticality of particular tasks	(Espinosa et al. 2007; Herbsleb and Mockus 2003; Cataldo et al. 2006, 2008; Wagstrom et al. 2010)	All work items were given equal importance in the Jazz development practices followed by the teams in the study. Thus, this factor is not relevant for this particular study.
16	**Task reassignment**: Extent to which development activities are reassigned amongst members of a team	(Cataldo et al. 2006)	Each work item has a single owner and changing work item ownership was not allowed in the Jazz development practices followed by the teams in the study. Thus, this factor is not relevant for our study.
17	**Team structure**: Organization of team and reporting hierarchy	(Curtis et al. 1988; Wolf et al. 2009; Meneely et al. 2011; Grewal et al. 2006)	Team structure can be seen at different levels. In the context of our study, we are interested in the structure that emerges out of the interaction between team members. As our independent variables capture some of the key aspects of team interaction structure, this factor has not been further considered as a control variable.
18	**Team workload** Level of work collectively assigned to a team	(Cataldo et al. 2006, 2008; Wagstrom et al. 2010)	This factor is considered as the *ActivitySpan* control variable.

Thus, the clustering coefficient of a vertex is the ratio of the actual number of edges existing between its neighbors and maximum number of such edges that can exist (Albert and Barabasi 2002). For an entire network, the clustering coefficient is the average clustering coefficients across all of its vertices. We recognize there are multiple ways of measuring the clustering coefficient (Albert and Barabasi 2002; Newman 2003b). We have chosen the above definition due to its intuitive appeal in our context. It has been verified that the general direction of our results based on the statistical models remain unchanged irrespective of how the clustering coefficient is measured.

5.3 Control variables

On the basis of existing literature and our study setting, the following categories of control variables are considered.

- **Amount of work**: It has been observed that the number of defects owned by a team is influenced by the amount of work the team is engaged in (Cataldo et al. 2006; Cataldo et al. 2008; Wagstrom et al. 2010). As a proxy for the amount of work, we consider the total number of task and enhancement work items for a team area – ***ActivitySpan*** – as a control variable. The notion of work items in Jazz development ensures the division of work into atomic units of relatively similar scope and intensity (Frost 2007). Thus we assume that controlling for the number of work items by the *ActivitySpan* variable also includes a reasonably accurate control for the complexity of development work handled by each team. We calculate *ActivitySpan* as the total number of task and enhancement work items for a team area.
- **Number of developers**: Whether and how the number of developers working together influences the number of defects in the software they produce has been explored at length (Brooks 1995; Raymond 2001). Accordingly, we consider the effects of ***TeamSize***, consistent with similar studies (Espinosa et al. 2007; Boh et al. 2007; Herbsleb and Mockus 2003; Wagstrom et al. 2010; Cataldo et al. 2009; Bird et al. 2009a). The total number of developers in a team area is taken as the *TeamSize*.
- **Familiarity with tasks**: The number of defects raised in a team's work products is expected to be related to the level of expertise of team members (Wagstrom et al. 2010). Expertise comes out of familiarity with the development tasks being performed (Espinosa et al. 2007). The level of *TaskFamiliarity* in the members of a team is thus an important factor that needs to be controlled for. For each developer, we count the number of releases for whose work items (s)he has posted at least one comment. The count for all developers in a team area is averaged to give the *TaskFamiliarity* for the team area. We assume that the act of commenting on work items reflects a developer's level of familiarity with units of work.
- **Distributed development**: As discussed in Section 4, the teams we are studying are distributed across a number of locations. Whether distance does or does not matter in large scale distributed development is a topic of much analysis and little agreement (Olson and Olson 2000; Wolf et al. 2008). The notions of spatial distribution and temporal distribution are well established in the literature on distributed development (Boh et al. 2007; Cataldo and Nambiar 2009; Herbsleb and Mockus 2003; Cataldo et al. 2008, 2009). Any analysis of the number of defects in the software produced by distributed teams must take into account these factors. Thus we consider *SpatialDistribution* and *TemporalDistribution* as control variables in our model.

– *SpatialDistribution*: The *Spatial Distribution* (SD) metric defined in (Cataldo and Nambiar 2009) can be customized in our context for team areas as:

$$SD = \frac{\sum_{p=1}^{p=\binom{k}{2}} (KM_{p_{ij}} * N_i * N_j)}{(N^2 - N)/2} \quad (1)$$

The notations denote:

- $KM_{p_{ij}}$ = Distance between the pair of locations i and j in kilometres.
- N_i, N_j = Number of developers in locations i and j respectively.
- k = Number of different locations the team members work from.
- N = Total number of developers in the team across all locations.
- $\binom{k}{2} = k(k-1)/2$ (i.e. k choose 2)

– *TemporalDistribution*: The *Temporal Distribution* (TD) metric defined in (Cataldo and Nambiar 2009) can be customized for team areas as:

$$TD = \frac{\sum_{p=1}^{p=\binom{k}{2}} (TZ_{p_{ij}} * N_i * N_j)}{(N^2 - N)/2} \quad (2)$$

The notations denote:

- $TZ_{p_{ij}}$ = Number of time-zones between the pair of locations i and j in kilometres.
- N_i, N_j = Number of developers in locations i and j respectively.
- k = Number of different locations the team members work from.
- N = Total number of developers in the team across all locations.
- $\binom{k}{2} = k(k-1)/2$ (i.e. k choose 2)

– **Socio-technical congruence**: The alignment between the communication structure of a team and the structure of dependencies in its work products has been studied for long (Conway 1968; Colfer and Baldwin 2010). This perspective has recently been developed further in the software development context as socio-technical congruence (STC) (Cataldo et al. 2008). STC has been found to be related to how a team performs (Cataldo et al. 2008; Kwan et al. 2011; Wagstrom et al. 2010). Thus we consider *SocioTechnicalCongruence* as a control variable. Based on the definition of socio-technical congruence in (Cataldo et al. 2008), in the context of this study, we compute the congruence (C) for a team area (X), C_X in the following way:

We generate two n by n square matrices M_t and M_w to capture the communication and work related connections between developers respectively. If developers d_i and d_j have co-commented on at least one work item, the value of "1" is entered in the cell ij of M_t (that is, the cell at the intersection of $i'th$ column and $j'th$ row) – as well as cell ji. If developers d_i and d_j own work item(s) that are dependent on one another, the value of "1" is entered in the cell ij of M_w as well as cell ji. (The information whether work items share dependencies between them is available from the Jazz repository.) This is repeated for all ij pairs of developers. All cells of M_t and M_w which are not marked by "1" are marked with "0". Let c_w be the number of cells

in M_w that have the the value of 1. Let us initialize another count variable $c_{both} = 0$. For each cell pq (that is intersection of $p'th$ column and $q'th$ row) that has a value of 1 in M_w, if the corresponding pq cell in M_t *also* has a value of 1, we set $c_{both} = c_{both} + 1$. What we are essentially calculating in c_{both} is the number of developers who are connected *both* through communication and work dependencies. We calculate $C_X = c_{both} / c_w$.

- **Different releases**: In an earlier study, we found that a development team's familiarity and utilization of Jazz's features evolve with progressive releases of the product being developed (Datta et al. 2012). As our data set spreads across three versions of the product, we introduce V_x and V_y as two explanatory "dummy" variables for the varying team dynamics between releases. The effects of different releases has been observed in similar other studies (Cataldo et al. 2006, 2008). Consistent with similar studies, with appropriate coding two variables can account for the three versions (Ehrlich and Cataldo 2012).
- **System age**: System age is known to influence team output (Espinosa et al. 2007; Herbsleb and Mockus 2003; Wagstrom et al. 2010; Grewal et al. 2006), and (Crowston and Scozzi 2002). In this study, for each team area we compute the elapsed time in days between the earliest and latest creation dates of all work items of types task or enhancement. The mean elapsed time for a team area is taken as the *Age* control variable.
- **Code change**: Given the importance of code change (Cataldo and Nambiar 2009) we include it as a control variable in our models. As mentioned earlier, Jazz work items have *change-sets* associated with them, which are collections of code files relating to various modifications undergone by the work items. For each work item of the type task or enhancement in each team area, we counted the total number of code files across all change-sets. The mean of this count for a team area is considered as the *FilesChanged* control variable.
- **Dependency**: Recognizing the role of dependency in team outcomes (Cataldo et al. 2006, 2008) we compute the *Dependency* control variable in our models in the following way: for a particular work item, the Jazz platform allows the recording of other work items this work item depends on. For each work item of the task or enhancement type in each team area in our study, we calculated the number of other work items it depends on. The mean number of other work items depended on by the work items in a team area is considered as the *Dependency* control variable.
- **Developer experience**: As developer experience has been found to influence team outcome (Espinosa et al. 2007; Faraj and Sproull 2000; Boh et al. 2007; Cataldo et al. 2006, 2008; Wagstrom et al. 2010), we include it as a control variable in our models. Assuming more experienced developers own more units of work, we consider the mean number of work items (of the task or enhancement types) owned by developers in a team area as the *Experience* control variable.

Table 2 presents the descriptive statistics of the variables introduced above. On the basis on these statistics we develop models to understand how the independent variables relate to the dependent variable, as explained in the next section.

6 Results

We now describe the development of the statistical models.

Table 2 Descriptive statistics of variables considered for modeling

	Mean	Median	Standard Deviation
DefectCountforTeam	408.93	90.00	814.00
Reachability	1.37	1.33	0.26
Connection	4.77	3.60	3.77
Association	−0.33	−0.36	0.30
Clustering	0.60	0.67	0.28
ActivitySpan	185.83	83.00	244.95
TeamSize	11.41	9.00	8.55
TaskFamiliarity	2.42	2.46	0.64
SpatialDistribution	1682.33	331.37	2445.47
TemporalDistribution	1.55	0.00	2.59
SocioTechnicalCongruence	0.64	0.80	0.39
FilesChanged	9.94	6.51	12.96
Dependency	0.93	0.84	0.54
Age	402.06	355.12	273.28
Experience	72.83	89.94	45.41

6.1 Choosing a modeling paradigm

In this study, the aim of developing statistical models is to understand the relationship between the dependent variables and the independent variables, in the presence of control variables. With that view, we outline the steps and considerations towards selecting the most appropriate modeling paradigm.

Our dependent variable, *DefectCountforTeam* is a *count* variable. To choose a modeling paradigm for examining the effects of the independent variables on the dependent variable, we first considered Poisson regression. Poisson distribution is a single parameter distribution with the mean equal to variance. Overdispersion – violation of this stringent underlying assumption of the equality of variance and mean – represents a major threat to the validity of Poisson regression (Barron 1992). Evidently, this threat is notable in our study. Using Poisson regression for the interaction model (as explained below) gives a residual deviance of 9117.3 on 57 degrees of freedom and Akaike Information Criterion (AIC) = 9574 (Akaike 1974). We next tried out negative binomial regression, which is also sometimes considered for count variables. The residual deviance for the interaction model using negative binomial regression is 81.79 on 57 degrees of freedom and the AIC is 863.7. To make a final decision on the choice of a modeling paradigm we also calculated the AIC of the interaction model using a multiple linear regression (with some of the variables transformed, as explained in Section 6.3), which came out to be 496.1, and compared it with the AIC for Poisson regression (9574) and the AIC for negative binomial regression (863.7). As the AIC for the multiple linear regression model is the lowest, it indicates a higher appropriateness of this modeling paradigm in our context than the negative binomial or Poisson models (Akaike 1974). Thus we chose multiple linear regression as our modeling paradigm.

With reference to Table 4, on the left we give parameters of the *base model* which only considers the effects of the control variables on the dependent variable, while on the right parameters of the *interaction model* - that additionally includes the independent variables - are presented. In the presentation of the model parameters in Table 4, superscripts of the

coefficients denote ranges of their respective p values, as we specify in the table caption. The p value for each coefficient is calculated using the t-statistic – the ratio of each coefficient to its standard error – and the Student's t-distribution. In the table's lower section, overview of the models are given: N denotes the number of data points used in building each model – in our case the number of team areas considered. R^2 is the coefficient of determination – the ratio of the regression sum of squares to the total sum of squares; it indicates the goodness-of-fit of a regression model in terms of the proportion of variability in the data set that is accounted for by the model. df denotes the degrees of freedom. F is the Fisher F-statistic – the ratio of the variance in the data explained by the linear model divided by the variance unexplained by the model. The p value is calculated using the F-statistic and the F-distribution, and it indicates the overall statistical significance of the model. For the coefficients as well as the overall regression, if $p \leq$ *level of significance*, we conclude that the corresponding result is statistically significant, based on null hypothesis significance testing.

6.2 Model assumptions

Multiple linear regression rests on the assumptions of linearity, normality, and homoscedasticity of the residuals, and absence of multicollinearity between the independent variables. The residual properties were verified using histograms, Q-Q plots and scatter plots of the standardized residuals.

Among the control variables, *SpatialDistribution* and *TemporalDistribution* had a high correlation (Pearson correlation coefficient of 0.9) which is understandable from their definitions (see Section 5). Between the two, *TemporalDistribution* is considered to be a more discerning factor in distributed development (Brooks 2010). Additionally, *ActivitySpan* and *TeamSize* had Pearson correlation coefficients of 0.65 and 0.57 respectively with the control variable *Age*; and Pearson correlation coefficients of 0.77 and 0.83 respectively with the independent variable *Connection*. Thus *SpatialDistribution*, *ActivitySpan* and *TeamSize* were removed from our models to satisfy the multicollinearity assumption, and ensure parsimonious models. To further check whether multicollinearity was posing problems for our models, we computed the Variance Inflation Factors (VIF) of all model variables, and they were found to be below the upper limit of 10 (Tabachnick and Fidell 2007). The correlations among the independent variables were low, the highest being between *Connection* and *Clustering* (Pearson correlation coefficient of 0.43). Table 3 presents the Pearson correlation coefficients between all pairs of model variables.

We may also take this occasion to comment on some of the moderate correlations (greater than 0.3 or less than -0.3) involving the control variables, in Table 3. At this time, these comments are conjectural, and each relationship needs to be studied separately to conclude why the variables are related the way they are. Such studies are beyond the scope of the current work, which focuses on the relationship between the dependent and the independent variables. However, we include these comments as they can illuminate interesting aspects of the project ecosystem. As we notice, *TaskFamiliarity* and *Experience* have a Pearson correlation coefficient of -0.34. The way these variables are measured (Section 5) can explain the strength of relationship. However the directionality is interesting; developers owning more work items (hence being better informed) seem to be less prone to post comments. Discussions with team members offered a possible explanation: developers often use the commenting mechanism in Jazz to pose questions to peers. Experienced developers are less likely to pose such questions, and hence comment less in general. *SocioTechnicalCongruence* and *Connection* have a correlation of 0.35: given the definition of the former, teams

Table 3 Correlation matrix for model variables: *TaskFamiliarity* (TF), *TemporalDistribution* (TD), *SocioTechnicalCongruence* (ST), *FilesChanged* (CH), *Dependency* (DP), *Age* (AG), *Experience* (EX), *Reachability* (RC), *Connection* (CN), *Association* (AS), *Clustering* (CL)

	TF	TD	ST	CH	DP	AG	EX	RC	CN	AS	CL
TF	1	−0.14	−0.18	−0.13	0.03	−0.08	−0.34	−0.08	−0.02	0.11	−0.07
TD		1	0.1	0.04	−0.15	−0.03	−0.05	−0.02	−0.07	0.02	−0.08
ST			1	0.11	0.05	0.27	−0.01	0.05	0.35	0.24	0.41
CH				1	−0.02	0.08	0.13	0.23	0.06	−0.05	0.09
DP					1	0.17	0.00	−0.04	0.1	−0.01	0.03
AG						1	−0.26	0.15	0.36	−0.01	0.40
EX							1	−0.04	−0.25	−0.02	−0.29
RC								1	0.17	−0.22	0.01
CN									1	−0.08	0.43
AS										1	−0.11
CL											1

with higher congruence are more likely to have higher average degree. A similar observation may also explain the correlation of 0.41 between *SocioTechnicalCongruence* and *Clustering*. *Age* has correlations of 0.36 with *Connection* and 0.40 with *Clustering*; long running projects are more likely to have higher average degrees and clustering coefficients among team members, as they have been interacting over extended periods of time.

6.3 Variable transformations

Although a skewness of around 3 for a variable is considered acceptable for including it in a linear regression model, we considered various established transformations for variables with the absolute value of skewness greater than 1.5, for making their distributions closer to normal (Tabachnick and Fidell 2007). We tried the following transformations separately on the candidate variables: taking the natural logarithm, taking the square root, and taking the fourth root. Out of these candidate transformations, taking the square root of the variables *DefectCountforTeam*, *TemporalDistribution*, *FilesChanged*, *Dependency*, *Experience* and *Connection* yielded the best combination of the following criteria: nearest approximation to normality of distribution of the transformed variable, closest model fit to the data, and most favourable cross-validation results (as reported in the next sub-section). Hence these variables were transformed by taking square root and the transformed variables were included in the models.

Based on the above discussion around verifying model assumptions and transforming variables, we concluded that the assumptions of linear multiple regression hold within permissible limits in our study (Tabachnick and Fidell 2007).

6.4 Model description and validation

From the parameters describing the base and interaction models respectively in Table 4, we note that both the models are statistically significant overall ($p < 0.001$), with the base model accounting for 62% of the variability of the data ($R^2 = 0.62$) and the interaction model accounting for 80% of the variability ($R^2 = 0.8$). Thus, the goodness-of-fit increases

Table 4 Modelling interaction characteristics. (Superscripts '*', '†' & ‡ denote $p \leq 0.01$, $p \leq 0.05$ & $p \leq 0.1$ respectively)

	I: Base Model		II: Interaction Model	
	Coefficient	Std error	Coefficient	Std error
Intercept	12.46	9.74	5.46	9.01
Control variables				
TaskFimiliarity	−3.08	2.46	−2.07	1.88
TemporalDistribution	−2.39	1.69	−1.49	1.30
SocioTechnicalCongruence	−0.22	3.44	−0.67	2.81
FilesChanged	1.09	0.80	0.58	0.62
Dependency	−1.86	4.57	−2.32	3.45
Age	0.04*	0.01	0.02*	0.01
Experience	−0.73‡	0.39	−0.4	0.3
V_x	4.59	5.2	0.46	4.13
V_y	−1.5	3.98	−3.07	3.13
Independent variables				
Reachability			−6.26	3.87
Connection			13.37*	1.89
Association			−0.74	3.22
Clustering			−10.18†	4.56
	Model parameters		Model parameters	
N	71		71	
R^2	0.62		0.8	
df	61		57	
F	10.9		17.5	
p	< 0.001		< 0.001	

from 62% to 80% with the introduction of the independent variable in the model, over and above the control variables. The standard technique of 10-fold cross validation was applied by randomly partitioning the data into 10 sub-samples, training the interaction model with 9 sub-samples and validating the model on the 10th sub-sample, and repeating this procedure 10 times. The Pearson correlation coefficient between the actual number of defects for each team and that predicted by cross validating the interaction model was found to be 0.89 ($p < 0.001$ for this correlation) with a mean absolute error of 195.2. Given the fact that the dependent variable has a range of 4198, the above mean absolute error - along with Pearson correlation coefficient and R^2 value - indicates a good fit of the regression model with the data.

In Table 4, the coefficient of each model variable is mentioned and the corresponding standard error is given beside it under the columns titled "I: Base Model", "II: Interaction Model". We note that the signs of the coefficients of the control variables remain unchanged between the base and interaction models, indicating overall stability of the models. Among the control variables, higher levels of *TaskFimiliarity*, *TemporalDistribution*, *SocioTechnicalCongruence*, *Dependency*, and *Experience* are seen to relate to lower number of defects whereas the effect is inverse for *FilesChanged* and *Age*.

As presented in Table 1, control variables considered for our study were drawn from the literature which identified their relevance to team outcomes in software development. Among the control variables, *Age* and *Experience* are the ones which have statistically significant effects in the models. The absence of statistical significance for the effects of other control variables can be due to various reasons.

Our statistical models are based on data from 71 teams. A larger cohort size is likely to enhance the effects of the control variables, leading to more instances of statistical significance. Within this limitation of data-set size, let us examine the control variables which do not have statistically significant effect in our models, in the light of how these are calculated (Section 5) and further analysis of the data. As *TaskFimiliarity* is measured in terms of the number of releases in which developers in a team area post comments, and there are only three releases considered in this study, *TaskFimiliarity* is unlikely to be a discerning factor in explaining the variability of the dependent variable. We found that many developers of large teams worked out of various locations of a country with a *single* time zone. Thus, in spite of notable spatial separation, these developers had no time difference; this may indicate why *TemporalDistribution* did not have statistically significant effect. A collaborative development environment such as Jazz strongly facilitates the alignment of social and technical interactions in the development ecosystem, unlike other systems for which socio-technical congruence has been studied (Cataldo et al. 2008). This alignment facilitated by Jazz may have contributed to the lack of statistical significance for *SocioTechnicalCongruence*. By studying the data, we could not identify probable reasons why the effects of *FilesChanged* and *Dependency* are not statistically significant. Further investigations - beyond the scope of the current study - are required to understand the lack of these effects. The explanatory "dummy" variables V_x and V_y included in the models to reflect on the varying team dynamics between releases did not reveal statistically significant effects, most likely because all three releases had a closely similar development context, without radically different environmental factors.

From the regression results, we observe the following:

- The null hypothesis corresponding to hypothesis H1, that is, there is no relationship between average separation and number of defects *is not rejected* with statistical significance of $p = 0.05$.
- The null hypothesis corresponding to hypothesis H2, that is, there is no relationship between average degree and number of defects *is rejected* with statistical significance of $p = 0.05$.
- The null hypothesis corresponding to hypothesis H3, that is, there is no relationship between assortativity and number of defects *is not rejected* with statistical significance of $p = 0.05$.
- The null hypothesis corresponding to hypothesis H4, that is, there is no relationship between average clustering coefficient and number of defects *is rejected* with statistical significance of $p = 0.05$.

Thus, we see that average degree (*Connection*) and average clustering coefficient (*Clustering*) have *statistically significant* relations with the defect count (at $p \leq 0.05$). Based on the sign of the coefficients, we notice that higher *Connection* relates to more defects, whereas higher *Clustering* is related to fewer defects.

7 Discussion

At the inception of this study, we sought to derive insights from relationships between social network metrics such as average separation, average degree, assortativity, and average clustering coefficient and defect count of software product development teams. The usefulness of these metrics depends on how closely the team networks reflect the interaction between team members. Such interaction has many facets: face-to-face meetings, conference calls, email communication, and use of a collaborative development environment such as Jazz. With reference to Sections 4 and 9, we recognize that not every detail of developer interaction may have been captured in the data we analysed. Such gaps may have led to the failures to reject the null hypotheses for H1 and H3; and these prevent us from deriving all the insights we expected from this study. These limitations of what we learned from this study come from the non-exhaustive nature of the data. With this background, we discuss the implications of our results.

Before discussing the statistically significant effects of the independent variables, let us consider the control variables. As we observe from Table 4, the influence of *Age* on the dependent variable is statistically significant. As explained in Section 6, *ActivitySpan* and *TeamSize* were found to be strongly correlated with *Age* and hence were removed from the final model. *TeamSize* was also strongly correlated with the independent variable *Connection*. In the light of these relationships, H2 can be alternatively taken to imply that larger teams or teams which have been in operation over longer periods of time, tend to generate more defects in their work products. Larger teams handle a bigger portfolio of activities, and it is not unexpected that these would lead to more defects. Older teams, on the other hand, tend to be in the maintenance mode, where activities around bug fixing are prevalent.

In the preceding sections, we have examined how interaction among team members using Jazz as their development platform relates to the number of defects in the teams' work products. Jazz has a mission of improving collaboration in the software development ecosystem, "Inspired by the artists who transformed musical expression, Jazz is an initiative to transform software and systems delivery by making it more collaborative, productive and transparent, through *integration of information and tasks* across the phases of the lifecycle"[4] (italics added). While integration of information and tasks remains a guiding principle, it is recognized that developers communicate in different ways to coordinate their work (Frost 2007). In the agile way of development, developers are encouraged to interact freely with one another to leverage the "tribal memory" of the community, as they go about their individual tasks (Booch 2008).

To better understand these dynamics, we spoke to a random sample of developers from the teams we studied with questions on how they interacted with their peers. We found that developers seemed committed to remain in close contact with their peers, often discussing issues not directly related to their immediate deliverables; and there was wide variability in the number of peers each developer could easily connect to. This variability mainly came out of developers' position and experience; those who have been working longer on a project had a larger peer group they closely interacted with. Understanding this context helps put our results in perspective.

[4] https://jazz.net/story/about/

As we see from Figs. 3, 4 , 5 and 6 the degree distributions of teams are generally skewed to the right, whereas the distribution of the clustering coefficients are generally skewed to the left. The right skewed distributions imply that few developers are connected to many others, while many developers are connected to few. On the other hand, the left skewed distributions indicate relatively many developers collaborate more, and few collaborate little. So it is evident that irrespective of how many peers they are connected to, there is a general tendency among developers to actively collaborate with their peers, which supports our understanding from talking to developers. As we observe from Table 2, the distribution of defects across teams is right skewed to some extent, implying that few teams have have many defects raised on their work products, whereas many teams have few defects. It is expected that larger teams, as well as teams that develop and maintain more complex parts of the system, will be handling more defects. The control variables in our regression models have accounted for such factors.

We find statistically significant evidence that average degree relates to more defects, while average clustering coefficient relates to fewer defects. Within the limitations of this study as identified earlier, as well as in Section 9, these results indicate that higher levels of connections between developers may not necessarily lead to higher quality software, unless there are concomitantly higher levels of clustering.

Being able to predict the number of defects in the system being developed by a team can aid project governance in meeting quality control guidelines. The results from cross validation as presented in Section 6 demonstrate that the *interaction model* can serve as a useful mechanism for defect prediction. There is potential for using this for quality assurance and integrating it as a feature in future collaborative development environments.

8 Related work

While developing our models in Section 5 we had discussed existing results that informed our choice of model variables. We now identify some general areas in literature which relate to our work.

8.1 Brief overview of related works

- **Organizational behavior and theory:** In the context of distributed software development teams, Perlow and Weeks have analyzed how connecting to and helping peers is perceived differently in different geographies due to the influences of national, organizational, and occupational cultures (Perlow and Weeks 2002). Monge et al. examine the evolution of communication networks in organizational communities, and illustrate how evolutionary principles such as variation, retention, and selection influence the life cycle of communication networks (Monge et al. 2008). Contractor et al., discus how technology can be moved inside a social network for an automobile design firm; this perspective has implications in the study of socio-technical networks of software development (Contractor et al. 2011).

  In the context of these studies, our results illuminate how intra-team information flow can influence outcomes in software development organizations, leading to a deeper understanding of factors relating to the quality of information technology work products.
- **Product development:** In a detailed literature survey, Brown and Eisenhardt synthesize different streams of product development literature into a model of factors that

influence the success of product development (Brown and Eisenhardt 1995). The model establishes the difference between process performance and product effectiveness, and emphasizes the impact of different stakeholders on the outcome of product development. McDonough, Kahn, and Barczak investigate how global, virtual, and collocated teams for new product development function differently, based on a survey across 103 firms (McDonough et al. 2001). Their results indicate that global teams face bigger project management and behavioral challenges than collocated teams, and have poorer performance levels.

The dilemma between collocating teams, versus distributing them across the globe was deeply prevalent in the system we studied. On one hand the very nature of global software development, nurtured by global organizations that produce and consume software products, makes it imperative that software teams become distributed. While on the other hand, there are conflicting conclusions about the effects of distance (Olson and Olson 2000; Wolf et al. 2008; Wagstrom and Datta 2014).

- **Social network analysis:** Krackhhardt and Brass outline how traditional micro organizational behaviour questions such as those around leadership, job design, turnover/absenteeism, and work attitudes can be examined by social network analysis (Krackhardt and Brass 1992). Borgatti and Foster analyse the increasing application of the network paradigm in organizational research (Borgatti and Foster 2003). They identify the dimensions along which network studies vary: causality, levels of analysis, explanatory goals, and explanatory mechanisms. Kilduff, Tsai, and Hanke emphasize the recognition of a set of core concepts such as primacy of relations, ubiquity of embeddedness, social utility of connections, and structural patterning of social life to "signal bold ideas" in the social network research as applied to organizational research (Kilduff et al. 2006).

  Interestingly, Kilduff et al. identify ubiquity of embeddedness as a core concept. Embeddedness brings with it ubiquity of connections, and our results show how connection between team members needs to be carefully moderated. In collective enterprises, "bold ideas" arise out of a subtle interplay of cooperation and conflict. In this context we note the contrasting effects of connection and clustering that we see in our study.
- **Socio-technical factors and the role of communication:** Given the wide proliferation of studies that use social network analysis (SNA) in the study of collaborative software development, Meneely and Williams examine whether SNA metrics actually measure what they are purported to measure (Meneely and Williams 2011). Their results support the perception that these metrics reflect socio-technical relationships, and offer guidance on their interpretation. In a related work, Meneely et al. also explore whether adding manpower affects quality, through a longitudinal analysis (Meneely et al. 2011), finding increased team size and linear growth to be correlated with subsequently better product quality. Crowston and Scozzi analyse 7,477 open source projects as virtual organizations, by validating a set of hypotheses around project participants developing necessary competencies (Crowston and Scozzi 2002). The relationship between distributed development and software quality has been studied by Bird et al., through an empirical case study of Windows Vista; the authors find a negligible difference in the post-release failure of components between distributed vis-a-vis collocated teams (Bird et al. 2009a). However, there is evidence that distributed work items take more times to be completed than co-located ones in a study by Herbsleb and Mockus (Herbsleb and Mockus 2003). As the teams we studied were predominantly distributed, some of this evidence may have been reflected in the detrimental effects of connection that we found in our study. Sacchi identifies the reliance of free and open source projects on

electronic communication media and virtual project management to coordinate globally dispersed development (Scacchi 2004). Grewal et al. have used data from a consortium of open source projects to show that network embeddedness significantly influences project success (Grewal et al. 2006). As higher embeddedness essentially leads to more connections, Grewal et al.'s conclusions point to the relevance of our results. Using ethnographic data, de Souza and Redmiles identify various strategies used by developers to address the effects of dependencies and changes in their work (de Souza and Redmiles 2008). Cataldo et al., introduce *socio-technical congruence* (STC) as a framework to analyze the effects of dependencies in software development (Cataldo et al. 2008). The authors present evidence that congruence of developers' coordination patterns with their coordination needs leads to notable reduction in the resolution time of modification requests. The relationship between successful coordination outcome and communication structures has been studied using Jazz data by Wolf et al. (Wolf et al. 2009). The authors build a predictive model using several communication structure measures. Wagstrom et al. extend the STC notion to differentiate between general communication and communication aligned to task dependencies (Wagstrom et al. 2010). Their key result highlights that the former does not have notable benefit, while the latter reduces bug resolution time.

These studies around socio-technical factors and its effects on development outcomes indicate that: *Connecting with peer developers is most beneficial when peers can offer assistance on specific tasks and challenges with these tasks often arise out of dependencies.*

– **Computer supported cooperative work:** Crowston demonstrates how the application of coordination theory can aid in organizational process design, such as the software change process (Crowston and Osborn 1998). Olson and Olson review over 10+ years of data from collocated and non-collocated synchronous group collaborations with a focus on socio-technical conditions that facilitate effective "distance work"; they conclude that "distance still matters" (Olson and Olson 2000). The same authors examine how the effects of distance on intellectual work can be mitigated, in a subsequent paper (Olson and Olson 2002). They conclude that while collaborative work at a distance will always have its challenges, the wide range of available collaborative tools will notably facilitate such work. Faraj and Sproull investigates whether and how expertise coordination is important in a study of 69 software development teams, and their analysis shows its significant influence on team performance (Faraj and Sproull 2000). Agerfalk and Fitzgerald recognize the challenges innate in global software development and present a balanced view of how agile methods seek to address these challenges (Agerfalk et al. 2006). Espinosa et al., examine whether there are gradual differences across time zones that influence team performance through a study of 42 teams (Espinosa et al. 2007). Their results show the nuanced influence of time zones across accuracy and production speed. Cataldo et al., study the relative performance of various dependency measures – syntactic, logical, and workflow – as they relate to customer-reported defects (Cataldo et al. 2009). They find that logical dependencies explained most of the variance, followed by workflow and syntactic dependencies. Cataldo has also studied the sources of errors in distributed projects and how that impacts the design of collaborative tools, through an empirical analysis of 209 projects (Cataldo 2010). He finds that experience, geographic distribution, technical properties of the product, and the project's schedule pressures have implications for collaborative tools. Huckman et al. analyse data from an Indian software services organization and find that team familiarity has a significant positive influence on performance, and the role experience of individuals in a team

leads to better team performance (Huckman et al. 2009). Sawyer et al. study 60 software development teams across 22 locations of 15 organizations to understand how social interactions influence team performance, using five patterns of team-level social interactions identified through cluster analysis (Sawyer et al. 2010). They find that no one pattern maximizes all performance measures. This is reflective of one of the arguments that has been used to motivate our study: consideration of a variety of typically social traits is necessary for a deeper understanding of developer interactions on software quality. Sosa proposes an *affiliation matrix* to relate the product architecture with the organizational structures, and uses this to address questions like who should talk to whom in a product development enterprise (Sosa 2008). Smite et al. conduct a systematic review of empirical studies in global software engineering (Smite et al. 2010) and conclude that the number of such studies is relatively small, with most studies at the superficial level, instead of engaging in deeper analysis. Cataldo and Herbsleb study the relationship between socio-technical congruence and software quality and development productivity (Cataldo and Herbsleb 2013), by analyzing data from two large projects from two companies with different characteristics. They found that gaps between coordination requirements and actual coordination activities of developers notably increased failures.

With this background of how our work relates to existing literature, let us now put our results in perspective.

8.2 Positioning our study

In the above discussion, we have identified how our results relate to specific conclusions available from existing studies. We now discern the following themes in literature that underpin our study.

- Mores of intra-organizational interaction between individuals can be effectively studied using the social network paradigm.
- Success of industrial product development is associated with several factors related to the location and interaction of team members.
- Socio-technical factors relate to the outcomes of large scale software development.
- Dynamics of communication between team members influence the quality of work products.

Our work complements existing studies by examining many teams working on the same organizational and development platform on a set of projects in a large product ecosystem. We extract typically social characteristics of developer interaction using the network paradigm, and analyze how they relate to the quality of team output. As tools and processes are being increasingly tuned for closer developer interaction in the hope that social factors will benefit team outcome, our results help in arriving at clear conclusions on which of such factors can help, and which may hinder.

9 Threats to validity

In this paper we report results from an observational study rather than a controlled experiment. Thus correlation does not necessarily imply causation in our statistical models. With this background, let us identify the limitations of our results.

9.1 Construct validity

Threats around construct validity relate to whether the variables are measured correctly.

As highlighted in earlier discussion, the network paradigm has been widely used for studying team dynamics. Our independent variables are measured using established network metrics. The measurement of control variables are also grounded in standard software engineering practices and existing literature.

The way we have constructed the team network is one among several approaches to building communication networks of software developers (Bird et al. 2009b; Ehrlich and Cataldo 2012). Although, our approach is consistent with similar studies (Wolf et al. 2009; Kwan et al. 2011), building the networks in a different way is likely to influence the results. In our networks, the links between developers are non-directional and un-weighted, as many of the enduring results of network theory pertain to this class networks (Newman and Park 2003). Directionality and weights of links may offer a wider context for a similar study and we plan to explore this in our future work.

On the basis of our discussion with team members, we understood work items to be similar in scope and granularity, leading to the assumption that work items tend to be of equal size. We recognize this assumption as a threat and have sought to mitigate its effects by some related variables in our models as mentioned in Table 1.

Usually, churn is measured as the number of lines of code added or removed in a given change-set. In our study, the *FilesChanged* control variable (as defined in Section 5) reflects the extent of change in terms of the number of files. Thus, even as we do not measure churn in the conventional sense, *FilesChanged* captures some aspect of how much the code base is changing.

In any project, teams are subject to attrition as existing members leave and new members join. Effects of attrition are significant over longer periods of time. In this study, each team network is constructed for a particular release, as specified in Section 4. As a release is a time-boxed set of iterations of limited duration, we assumed the rate of attrition to be minimal. However, as the identity of an erstwhile member of a team and his/her replacement can not be reconciled, such individuals exist as separate vertices in our team networks. We recognize the effects of attrition as a threat to construct validity.

Depending on their size, some teams may have been internally organized into sub-teams, with each sub-team working relatively independently on a particular piece of functionality. In such situations, interactions across sub-teams may have been limited. As sub-team information was not available in the repository, we recognize this as another threat to construct validity.

9.2 Internal validity

Internal validity ensures that a study is free from systematic errors and biases.

As the Jazz repository is our only source of data, common issues affecting internal validity such as mortality and maturation do not arise in this case. A key influence on internal validity is the extent to which the development team used the Jazz platform. Even as Jazz was the mandated collaborative development environment for the system's development, developer comments on work items (as captured in Jazz) were sometimes clarified by telephonic or face-to-face conversations. From our discussion with developers we understood that summaries of such conversations were recorded as comments on the Jazz platform as a standard practice.

We also learned from these discussions that allocation of units of functionality to work items were done during the design workflow, and the reasons behind allocating a particular task to a work item were not documented on the Jazz platform. We gathered from the developers that dependencies between work items were recorded in the Jazz platform on the basis of discussions between owners of the work items, and they were verified during internal reviews. As this is an essentially manual process for recording work item dependencies, few dependencies may have been inadvertently missed. However, since we found very few work items without dependencies recorded in Jazz, we have reason to believe that manual recording of work item dependencies was largely consistent and adequate.

We also found that priority was not mentioned for a large number of work items across different teams. We understood from our discussions with developers that open work items at any point in the development life cycle were addressed with similar levels of attention. Thus, even as our assumptions around the work items as outlined in Section 5 are based on discussions with developers, we recognize them as threats to internal validity.

As explained in Section 5, control variables considered in this study such as *ActivitySpan* reflect on the level of complexity of the work products of each team. In Table 2, we observe that the number of defects across teams' had a high variance. This is expected, given the wide variety of team sizes. However, as all teams were developing parts of the same product, we assume their development contexts were comparable, with similar goals and expectations. We recognize this assumption as a threat to internal validity.

9.3 External validity

External validity is concerned with the generalizability of results.

Our study ranges across many teams working on multiple versions of a single product developed on the Jazz platform. As demonstrated in existing literature, useful insights can be drawn from observational studies on single subjects (Wolf et al. 2008, 2009; Bird et al. 2009b). Thus even as we do not claim our results to be generalizable as yet, we believe our results present insights that can inform similar studies.

9.4 Reliability

A study's reliability is established when the results are reproducible.

From the discussion in Section 4, evidently there is no human intervention in the extraction and processing of data. Once the data is extracted, we use standard statistical techniques for analyzing the data and interpreting the results. Given access to the Jazz repository, our results can be reproduced by following the steps outlined in this study.

10 Future work

The evidence we found about more connections between team members relating to more defects, warrants further investigation. In our future work, we plan to conduct observational studies on larger data-sets as well as controlled experiments to establish whether unbridled connection between team members indeed *causes* lower quality of work products. We also find evidence that while too much connection is detrimental, enhanced intra-team clustering relates to higher quality of the team's work products. This leads to an intriguing question whether splitting a large team into smaller and more cohesive units can help in some

situations. We plan to explore this in our future work, by developing a set of guidelines for such splitting, and testing whether the teams after splitting indeed function better.

11 Summary and conclusions

In this paper, we examined how various aspects of developer interaction in distributed software development teams relate to the quality of software produced by the teams. Based on an empirical study of many teams working across multiple versions of a major industrial product, we found statistically significant evidence that a higher level of connection between team members relates to more defects, while higher clustering is associated with fewer defects. These results can guide individual developers, project managers, as well as organizations in the planning and positioning of resources in large scale distributed software development.

References

Agerfalk J, Fitzgerald B, In OP (2006) Flexible and distributed software processes: old petunias in new bowls. Commun ACM 49:27–34

Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723

Albert R, Barabasi A (2002) Statistical mechanics of complex networks. Rev Mod Phys 74:47. [Online]. Available: arXiv:0106096, Jun. 2001

Barabasi AL, Jeong H, Neda Z, Ravasz E, Schubert A, Vicsek T (2002) Evolution of the social network of scientific collaborations. Physica A 311(3-4):590–614. [Online]. Available: arXiv:0104162, Apr. 2001

Barron D (1992) The analysis of count data: overdispersion and autocorrelation. Sociol Methodol 22:179–220

Bird C, Nagappan N, Devanbu P, Gall H, Murphy B (2009) Does distributed development affect software quality? an empirical case study of windows vista. In: Proceedings of the 31st international conference on software engineering, ser. ICSE '09. IEEE, Computer Society, Washington, DC, pp 518–528

Bird C, Nagappan N, Devanbu P, Gall H, Murphy B (2009) Putting it All together: using socio-technical networks to predict failures. In: Proceedings of the 17th international symposium on software reliability engineering IEEE computer society

Boh WF, Slaughter SA, Espinosa JA (2007) Learning from experience in software development: a multilevel analysis. Manag Sci 53(8):1315–1331

Booch G (2008) Tribal memory. IEEE Softw 25(2):16–17

Borgatti SP, Foster PC (2003) The network paradigm in organizational research: a review and typology. J Manag 29(6):991–1013

Brooks FP (1995) The mythical man-month: essays on software engineering, 20th anniversary edition. Addison-Wesley

Brooks FP (2010) The design of design: essays from a computer scientist. Addison-Wesley, Upper Saddle River, NJ

Brown SL, Eisenhardt KM (1995) Product development: past research, present findings, and future directions. Acad Manag Rev 20(2):343

Cataldo M (2010) Sources of errors in distributed development projects: implications for collaborative tools. In: Proceedings of the 2010 ACM conference on Computer supported cooperative work, ser. CSCW '10. ACM, New York, pp 281–290

Cataldo M, Herbsleb J (2013) Coordination breakdowns and their impact on development productivity and software failures. IEEE Trans Softw Eng 39(3):343–360

Cataldo M, Herbsleb JD, Carley KM (2008) Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In: Proceedings of the Second ACM-IEEE international symposium on empirical software engineering and measurement, ser. ESEM '08. ACM, New York, pp 2–11

Cataldo M, Mockus A, Roberts JA, Herbsleb JD (2009) Software dependencies, work dependencies, and their impact on failures. IEEE Trans Softw Eng 35(6):864–878

Cataldo M, Nambiar S (2009) Quality in global software development projects: a closer look at the role of distribution. In: Fourth IEEE international conference on global software engineering, 2009. ICGSE 2009, pp 163–172

Cataldo M, Wagstrom PA, Herbsleb JD, Carley KM (2006) dentification of coordination requirements: implications for the design of collaboration and awareness tools. In: Proceedings of the 2006 20th anniversary conference on computer supported cooperative work, ser. CSCW '06. ACM, New York, pp 353–362

Coleman J (1990) Foundations of social theory. Harvard University Press

Colfer L, Baldwin CY (2010) The mirroring hypothesis: theory, evidence and exceptions, social science research network, Rochester, NY SSRN Scholarly Paper ID 1539592

Contractor NS, Monge P, Leonardi P (2011) Multidimensional networks and the dynamics of sociomateriality: bringing technology inside the network. Int J Commun 5:682720

Conway M (1968) How do committees invent?. Datamation J, pp 28–31

Cringely RX (1996) Accidental empires: how the boys of silicon valley make their millions, battle foreign competition, and still can't get a date. Collins

Crowston K, Osborn CS (1998) A Coordination theory approach to process description and redesign

Crowston K, Scozzi B (2002) Open source software projects as virtual organisations: competency rallying for software development. Softw IEE Proc 149(1):3–17

Curtis B, Krasner H, Iscoe N (1988) A field study of the software design process for large systems. Commun ACM 31:1268–1287

Datta JP (1998) Iso 9000: A roadmap for design, installation and implementation of quality management systems. Manuscript

Datta S, Sindhgatta R, Sengupta B (2012) Talk versus work: characteristics of developer collaboration on the jazz platform. In: Proceedings of the ACM international conference on object oriented programming systems languages and applications, ser. OOPSLA '12. ACM, New York, pp 655–668

de Souza CRB, Redmiles DF (2008) An empirical study of software developers' management of dependencies and changes. In: Proceedings of the 30th international conference on software engineering, ser. ICSE '08. ACM, New York, pp 241–250

DeMarco T, Lister T (1987) Peopleware: Productive projects and teams. Dorset House Pub Co

Dorogovtsev SN, Goltsev AV, Mendes JFF (2002) Pseudofractal scale-free web. Phys Rev E, 65

Ehrlich K, Cataldo M (2012) All-for-one and one-for-all?: a multi-level analysis of communication patterns and individual performance in geographically distributed software development. In: Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, ser. CSCW '12. ACM, New York, pp 945–954

Espinosa JA, Slaughter SA, Kraut RE, Herbsleb JD (2007) Familiarity, complexity, and team performance in geographically distributed software development. Organ Sci 18(4):613–630

Faraj S, Sproull L (2000) Coordinating expertise in software development teams. Manag Sci 46(12):1554–1568

Feld S (1991) Why your friends have more friends than you do. Am J Soc 96(6):1464–1477

Frost R (2007) Jazz and the eclipse way of collaboration. IEEE, Softw 24(6):114–117

Grewal R, Lilien GL, Mallapragada G (2006) Location, location, location: how network embeddedness affects project success in open source systems. Manage Sci 52(7):1043–1056

Herbsleb J, Mockus A (2003) An empirical study of speed and communication in globally distributed software development. IEEE Trans Softw Eng 29(6):481–494

Herzig K, Zeller A (2009) Mining the jazz repository: challenges and opportunities. In: 6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR '09, pp 159–162

Huckman RS, Staats BR, Upton DM (2009) Team familiarity, role experience, and performance: evidence from Indian software services. Manage Sci 55(1):85–100

Humphrey WS (1999) Introduction to the team software process. SEI series in software engineering

Jackson MO (2010) Social and economic networks. Princeton University Press, Princeton

Kilduff M, Tsai W, Hanke R (2006) A paradigm too far? a dynamic stability reconsideration of the social network research program. Acad Manag Rev 31(4):1031–1048

Kleinberg J (2000) The Small-World phenomenon: an algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing, pp 163–170

Koru AG, Liu H (2005) Building defect prediction models in practice. IEEE Softw 22(6):23–29

Krackhardt D, Brass DJ (1992) Intraorganizational networks: the micro side. In: advances in social network analysis: research in the social and behavioral sciences. SAGE Publications, Inc., Thousand Oaks, pp 207–229

Kwan I, Schroter A, Damian D (2011) Does socio-technical congruence have an effect on software build success? a study of coordination in a software project. IEEE Trans Softw Eng 37(3):307–324

McDonough IIIEF, Kahn KB, Barczak G (2001) An investigation of the use of global, virtual, and colocated new product development teams. J Prod Innov Manag 18(2):110–120
Meneely A, Rotella P, Williams L (2011) Does adding manpower also affect quality?: an empirical, longitudinal analysis. In: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ser. ESEC/FSE '11. ACM, New York, pp 81–90
Meneely A, Williams L (2011) Socio-technical developer networks: should we trust our measurements? In: Proceedings of the 33rd International Conference on Software Engineering, ser. ICSE '11. ACM, New York, pp 281–290
Milgram S (1967) The small-world problem. Psychology Today 1:61–67
Monge P, Heise B, Margolin D (2008) Communication network evolution in organizational communities. Commun Theory 18:449–477
Newman MEJ (2002) Assortative mixing in networks. Phys Rev Lett 208701:89. arXiv:0205405
Newman MEJ (2003) Mixing patterns in networks. Phys Rev E 67:026126. arXiv:0209450, Sep. 2002
Newman MEJ (2003) The structure and function of complex networks. SIAM Rev 45:167–256. arXiv:0303516
Newman MEJ, Park J (2003) Why social networks are different from other types of networks. Phys Rev E 68:036122. [Online]. Available: arXiv:0305612
Olson GM, Olson JS (2000) Distance matters. Hum Comput Interact 15(2):139–178
Olson GM, Olson JS (2002) Mitigating the effects of distance on collaborative intellectual work. Econ Innov New Technol, pp 27–42
Perlow L, Weeks J (2002) Who's helping whom: A comparison of helping behavior among american and Indian software engineers. J Organ Behav 23(4):345361
Ravasz E, Barabási A-L (2003) Hierarchical organization in complex networks. Phys Rev E 67:026112. [Online]. Available: doi:10.1103/PhysRevE.67.026112
Raymond ES (2001) The cathedral and the bazaar: musings on linux and open source by an accidental revolutionary. O'Reilly
Sawyer S, Guinan PJ, Cooprider J (2010) Social interactions of information systems development teams: a performance perspective. Inf Syst J 20(1):81–107
Scacchi W (2004) Free and open source development practices in the game community. IEEE Softw 21(1):59–66
Smite D, Wohlin C, Gorschek T, Feldt R (2010) Empirical evidence in global software engineering: a systematic review. Empir Softw Eng 15(1):91–118
Sosa ME (2008) A structured approach to predicting and managing technical interactions in software development. Res Eng Des 19(1):47–70
Stroustrup B (2007) The problem with programming. http://www.technologyreview.com/InfoTech/17987/?a=f. Last accessed: December 12, 2013
Tabachnick B, Fidell L (2007) Using multivariate statistics. Pearson Education, Boston
Travers J, Milgram S (1969) An experimental study of the small world problem. Sociometry 32(4):425
Wagstrom P, Datta S (2014) Does latitude hurt while longitude kills? geographical and temporal separation in a large scale software development project. In: Proceedings of the 36th International Conference on Software Engineering, ser. ICSE 2014. ACM, New York, pp 199–210. [Online]. Available: doi:10.1145/2568225.2568279
Wagstrom P, Herbsleb JD, Carley KM (2010) Communication, team performance, and the individual: Bridging technical dependencies. Acad Manag Proc 2010(1):1–7
Watts D (1999) Networks, dynamics, and the Small-World phenomenon. Am J Soc 493(2):527
Watts DJ, Strogatz SH (1998) Collective dynamics of /'small-world/' networks. Nature 393(6684):440–442
Weinberg GM (1971) The psychology of computer programming. Van Nostrand Reinhold
Wolf T, Nguyen T, Damian D (2008) Does distance still matter? Softw Process 13(6):493–510
Wolf T, Schroter A, Damian D, Nguyen T (2009) Predicting build failures using social network analysis on developer communication. In: Proceedings of the 31st International Conference on Software Engineering, ser. ICSE '09. IEEE Computer Society, Washington, DC, pp 1–11
Zimmermann T, Nagappan N (2009) Predicting defects with program dependencies. In: 3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009, pp 435–438