4-2010

A social network based study of software team dynamics

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

Vikrant S. KAULGOUD

Vibhu Saujanya SHARMA

Nishant KUMAR

A Social Network Based Study of Software Team Dynamics

Subhajit Datta Vikrant Kaulgud Vibhu Saujanya Sharma

Nishant Kumar

Accenture Technology Labs
IBC Knowledge Park, 4/1 Bannerghatta Road, Bangalore 560 029, India
{subhajit.datta, vikrant.kaulgud, vibhu.sharma, nishant.x.kuma}@accenture.com

ABSTRACT

Members of software project teams have specific roles and responsibilities which are formally defined during project inception or at the start of a life cycle activity. Often, the team structure undergoes spontaneous changes as delivery deadlines draw near and critical tasks have to be completed. Some members – depending on their skill or seniority – need to take on more responsibilities, while others end up being peripheral to the project's execution. We posit that this kind of *ad hoc* reorganization of a team's structure can be discerned from the project's bug tracker. In this paper, we extract a social network from the bug log of a real life software system and apply ideas from social network analysis to understand how the positions of individual team members in the network relate to their organizational seniority, project roles, and geographic locations that define the formal team structure. In addition to providing insights on individual team members for the system studied, our approach can serve as a framework for analyzing team dynamics of software projects.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*life cycle, programming teams*; J.4 [**Social and Behavioral Sciences**]: Sociology

General Terms

Human Factors, Management

Keywords

software teams, social networks, bugs, centrality

1. INTRODUCTION

Professional software development is a team enterprise. Humphrey has pointed out that collaboration in a team succeeds when each team member has a specific role, the team collectively has a common goal and reaching the common goal needs some form of dependency among the team members [14]. Thus team dynamics play an important role in the success of any project.

Software teams consist of individuals at different levels of organizational seniority, varying skill-sets, and – in this age of global software development – often spread across geographic locations. When a project or a phase thereof begins, individuals are assigned specific roles depending on these factors. However as deadlines draw near, the team structure evolves spontaneously to respond to problems or contingencies. Some individuals become more central to the project's execution; they tend towards *omnipresence* – being everywhere in the project's execution map, and concomitantly, *omniscience* – concentration points of project knowledge. On the lines of Huxley's *alphas* [15], we will call these individuals the *omnis*. Omnis are critical to a project's success as they have as they strongly influence team congruence.

A congruent team is one in which people play roles, participate in delivery activities and collaborate as defined in the project delivery plan. Given a particular project plan, a congruent team can be thought to represent the most *harmonious* structure of the team. In a congruent team, individual team members fulfill responsibilities in perfect harmony with their skills, organizational seniority and other factors like location. *Degree of team congruence* can be thought to be the extent to which congruence as outlined above exists in a team. It can then be taken as metric by which to assess teams and possibly benchmark teams with each other and across established baselines.

Identification of the omnis with a view to better understanding of team congruence is the "team dynamics" we seek to study in this paper. Towards that end, an experiment is described to detect the omnis of a real life proprietary software system by using a social network extracted from the project's bug tracker. We recognize that bug tracker is not the only artifact that can help identify omnis; code repositories are among several others which may also be useful.

The following sections discuss motivation, research questions, outline of our approach, and related work. Subsequently, we present the assumptions and methodology of our study. The experimental results are analyzed next, and the paper ends with directions of future work and conclusions.

2. MOTIVATION

As discussed earlier, identifying the omnis has much importance in understanding and influencing team congruence. Identifying omnis is a non-trivial task; especially using a manual processes of sifting through project artifacts, or interviewing project stakeholders.

Everybody in a team likes to be seen as an omni, so asking around may not always help much in finding the real omnis. We could inquire some stakeholder of authority – say the project manager – but his/her opinion may be colored by preference or prejudice. Some artifact that is created and maintained in the process of project execution seems to be a more objective source of insights. Given that we have such an artifact, how can we automatically analyze it to find the omnis?

From Humphrey's observations mentioned earlier, we notice that inter-dependence between team members is an important aspect of successful collaboration [14]. In a team context, dependence is manifested in interaction. Team members build relationships amongst themselves around tasks and targets. Individuals may have specific attributes – depending on their skills, hierarchical positions etc – but when working together in a team, the relationships they form with other team members are of great consequence. Social network analysis perceives group dynamics through the lens of relationships, rather than attributes [13]. The thinking behind our study is that if social network(s) can be extracted from project artifact(s), the network characteristics can help us identify the omnis for that project. The particular kind of social networks of interest to us are the affiliation networks (discussed in detail in a later section).

3. RESEARCH QUESTIONS

Based on the preceding discussion, we state our hypothesis as: *Given the bug tracker of a real life software system, it is possible to identify the omnis – individuals who are most central to the bug resolution activity – using social network analysis, thereby providing insights on team congruence.*

Taking validation of the hypothesis as the goal, the following questions need to be addressed:

- How do we recognize relationships – manifested in common affiliations – between team members from the information available in the bug tracker?
- From the social network generated from the bug tracker, how do we objectively identify the omnis?
- How do the omnis identified from the social network relate to the actual position of the individuals in the team structure in terms of their organizational seniority, official role, and geographic location?

We next outline our approach towards answering these questions.

4. OUTLINE OF THE APPROACH

The basic idea behind affiliation networks is that two types of entities may be perceived in a social context: groups and members, and the two are related by affiliations. These relationships can be described through bipartite graphs, which are called *affiliation networks* [19]. (The vertices of a bipartite graph are divided into two *disjoint* sets U and V, such that every edge connects a vertex in U to a vertex in V.) From the bipartite graph, a social network can be derived by substituting paths of length two among vertices in either one of the sets – U or V – by an edge.

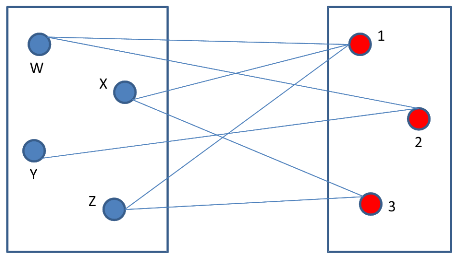

Figure 1: The Bipartite Graph of an Affiliation Network

Figure 1 depicts the affiliation network in the form of a bipartite graph; where vertices on the left (W, X, Y, Z) are individuals and vertices on the right (1, 2, 3) are affiliations (clubs, places of work etc. – anything individuals may be attached to), and the links signify memberships. W is a member of (say) club 1 and 2, X is a member of 1 and 3, and so on. Figure 2 gives the social network arising from the affiliation network of Figure 1. There is an edge between W and X since *both* are members of 1. The weights of the links in Figure 1 signify the number of co-memberships; edge XZ has weight of 2 since both X and Z are members of the same *two* clubs, namely 1 and 3.

It may be noted that given an affiliation network in the form of a bipartite graph, we can generate two social networks from it: one each for the vertices of U and V. From Figure 1 we have extracted the network where the vertices are the individuals (W, X, Y, Z), as shown in Figure 2. We could have also extracted a network where the vertices are the clubs (1, 2, 3). In the context of this paper, a network of individuals is of primary interest to us.

As apparent from the preceding discussion, generating a social network from a bipartite "affiliation" graph is based on the observation that relationships between people are often developed due to some common affiliation – going to the same club, taking the same train to work, writing a paper together, and in our case, working on the same bug. The study of affiliation networks goes back several decades; Breiger had explored the duality of persons and groups in his eponymous paper in 1974 [7].

By parsing the bug tracker, our objective is to detect the social network that comes out of the affiliations between individual team members working together on a bug. In this context an *affiliation* between two individuals is recognized if *both* of them have worked on *at least one* particular bug. Thus in the bipartite graph of the affiliation network, the vertices in set U are all the individual team members engaged in the bug resolution activity, and the vertices in set V represent the bugs. Given this bipartite graph, we generate the corresponding social network between individual team

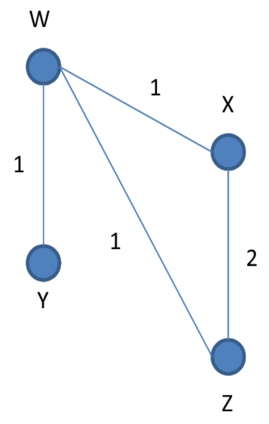

Figure 2: The Social Network Extracted from Figure 1

members based on the bugs they have worked on. A social network between the bugs could also have been generated from the same affiliation network, recognizing a different affiliation – two bugs being related if at least one individual worked on both of them. But this network would not have been useful in identifying the omnis.

The vertices of our social network thus represent individuals, and the undirected links – edges – represent the affiliations. Each edge has a *weight*, which indicates the number of affiliations shared by the vertices connected by that edge. For example, if A and B have both worked on one bug, the edge between A and B will have the weight of 1, whereas if there are six bugs which A and B have both worked on, the edge between A and B will have the weight of 6.

Our objective is to find the omnis - individuals representing significant concentration of project knowledge. It is reasonable to assume such individuals will also be in position of considerable power, as measured by their ability to influence the team's success in the activity it is engaged in (bug resolution, in our case). In social network analysis, it is an established viewpoint that "central positions tend to be more powerful positions" [13]. Thus, in our hunt for the omnis, we first measure the centrality of each vertex of the network.

Centrality may be measured in several ways:

- Degree: By far the simplest measure of centrality, degree is the number of links a particular vertex has to other vertices in the network. If a vertex has a high degree, it implies it is well connected to other vertices, and vice versa. For our study we will consider the *Relative Degree Centrality* $C_D(x)$ of the vertex x, as given by [11]:

$$C_D(x) = \frac{c_D(x)}{n-1} \qquad (1)$$

where, $c_D(x)$ is the degree centrality of x, that is the degree of x, and $n-1$ is the highest degree of the network of n vertices. In this case, the degree of a vertex is the number of edges incident on the vertex.

- Closeness: A common criticism of degree centrality is that it accounts for only the immediate links of a vertex. A vertex may be closely connected to its neighbors – thus having a high degree – but the neighbors themselves may be isolated from the network as a whole. Thus considering degree centrality, our vertex of interest will be central, but in a very local sense. The idea of closeness centrality tries to address this inadequacy by considering the distance from the vertex of interest to all other vertices in the network, instead of just its neighbors [13]. Closeness centrality thus considers not only direct connections between vertices, but also indirect connections. In our context we will consider the *Relative Closeness Centrality* $C_C(x)$ of the vertex x, as given by [25], [11]:

$$C_C(x) = (n-1) * c_C(x) \qquad (2)$$

where, $c_C(x)$ is the closeness centrality of x. $c_C(x) = \frac{1}{\sum_{y \in U} d(x,y)}$; $d(x,y)$ being the graph theoretic distance – the length of the shortest path – between vertices x and y, and U is the set of all vertices.

- Betweenness: In addition to degree and closeness, another dimension of centrality may have to be considered. Power often comes from being in a position to serve as an intermediary or a connector between two entities who can not otherwise be in contact [13]. If the vertex of interest lies on the shortest path between pair(s) of other vertices, it will be in a position to influence the information flow in the network. The notion of betweenness recognizes a vertex to be central if it lies on several shortest paths between pairs of vertices in the network. We will use the metric for *Relative Betweenness Centrality* $C_B(x)$ for undirected networks, given by [12], [11]:

$$C_B(x) = \frac{c_B(x)}{(n-1) * (n-2)} \qquad (3)$$

where, $c_B(x)$ is the betweenness centrality of vertex x, defined as $c_B(x) = \sum \frac{f(x,y,z)}{g(y,z)}$ where, $f(x,y,z)$ gives the number of shortest paths between the vertices y and z that go through the vertex x and $g(y,z)$ gives the number of shortest paths between y and z $(y < z)$.

Additionally, from the bug tracker we can extract the information on how many times a particular individual has created (that is detected or raised) a bug, been assigned a bug, and have contributed to the bug's resolution without having created or assigned it. We will then try to correlate this information with the measures of centrality discussed earlier; how do the omnis relate to those who raise, get assigned, or work on a bug?

As mentioned earlier, the team members of the system we studied have different levels of organizational seniority, official roles in the project, and geographic locations (see

Table 1). While trying to find the omnis amongst them, it is also interesting to know whether and how individuals of the same seniority level, same role, or same location relate to each other within the team. Do they form *cores*, or teams within teams? (A k-core is a maximal group of vertices, all of which are connected to some number – k – of other members of the group [11], [13]) As in the social network we derive from the bug tracker edges also have weights, it implies that two team members who are joined by a link of higher weight are closer than two who are connected by a lower weighted link. To understand the implications, we also calculate *valued cores*, which take into account the values of edges between vertices instead of just counting the links between them [11], [13].

The values of the above metrics and their interpretation will set us up to confront the question: Are the omnis those individuals who are *expected* to have central position in the bug resolution activity in terms of their organizational seniority, official role and geographic location?

In the next section, we give an overview of related work.

5. RELATED WORK

The theory and characteristics of social and affiliation networks have been explored by a number of researchers. Watts and Strogatz investigate the collective dynamics of small world networks [26]. Kleinberg has studied navigation in a small world [18], and algorithmically analyzed the long recognized small world phenomenon [17]. Lattanzi and Sivakumar discuss affiliation networks and their features at great depth [19]. The evolving characteristics of social network graphs over time have been studied by Leskovec et al. [20].

Ideas from social network analysis have been applied in diverse contexts. Pinzger et al. use developer-module networks to investigate the relationship between the fragmentation of developer contributions and the number of post-release failures [23]. CVS repository information related to source code has been studied using social network analysis in [21]. A case study on the penetration of social network analysis within the enterprise has been conducted by the Burton Group [1]. Bird et al. study the extraction of social networks from emails in [6]. Social networks of Java classes have been shown to obey the power-law distribution in [24]. The topology of the so-called "dark networks" – terrorist groups and individuals connecting with one another across the world – has been analyzed using social networking ideas in [27]. Dietrich et al. explore knowledge sharing within the software engineering community based on social networking in [10]. The idea of social capital as a motivating factor in open source projects has been studied in [22].

Social networking has been found to be useful for analyzing the occurrence, location, as well as handling of bugs. Chen et al. propose a social network based model for predicting and tracking the location of faults [9]. In [16], bug triage "with bug-tossing" graphs is explored. Arand et al., track the life cycle of bugs from a social and organizational perspective [3]. A seminal paper on the social perspective of software development suggests a set of organizational patterns reflecting on the productivity of organizations [8]. The authors take the position that software development is predominantly a social activity and hence a sociological view of the software process is required [8]. The use of affiliation networks to understand aspects of software development has been posited in [2].

Our study is inspired by many of these important contributions. Using the framework of social network analysis, we seek to understand one aspect of the functioning of teams, vis-a-vis their formal structure.

There are several tools, both proprietary and open source, for analyzing social networks. In this paper, we have used *Pajek*: Analysis and Visualization of Large Networks [4], [5] for creating the network and calculating the metrics.

6. ASSUMPTIONS AND METHODOLOGY

For conducting the experiment, we selected the bug tracker of a proprietary software system. The tracker was used for recording information related to bugs identified through internal testing by the development team.

Our experiment was based on the following assumptions:

- Each bug and each individual team member is *uniquely* identifiable in the bug tracker.
- The entry for a particular bug in the bug tracker serves as the single point of reference for all the information of interest relating to that bug; that is, no other project artifact needs to be referred to for deriving the social network.
- The entry for a particular bug in the bug tracker captures the names of *all* the individuals who have worked on the bug; from identification to resolution and post-resolution verification.
- Necessary information related to organizational seniority, official roles, and geographic locations of individual team members can be reliably obtained from other project artifacts.

The following methodology was followed in conducting the experiment:

From the bug tracker, the affiliations between each pair of individuals was extracted using a set of Java programs. The affiliation information was converted into an *adjacency matrix* [13] describing the corresponding social network in a Pajek compatible input format. Pajek was used to generate the social network from the this input file, and subsequently calculate the measures of centrality and the memberships of the k-core and valued cores mentioned earlier.

Additionally, the number of times an individual has raised a bug, been assigned a bug, or has worked on a bug without either having raised it or been assigned to it was also calculated. Further information was gathered specifying the organizational seniority of each individual on a scale of 1 to 3 (1 being the highest), the formal role of the individual in the project (developer, architect etc.), and the geographic location of the individual.

The above information was interpreted in the light of our quest for the omnis.

In the next section, we discuss the experimental results in detail.

7. EXPERIMENTAL RESULTS

7.1 Description of the System Studied

We studied a proprietary software system whose functionality centered around automatically parsing requirement descriptions for detecting omissions and inconsistencies. The

Table 1: Team Member Data

Member	Role	Seniority	Location
A	Quality Assurance	3	X
B	Architect	2	X
C	Quality Assurance	3	X
D	Developer	3	X
E	Developer	3	X
F	Developer	3	X
G	Developer	3	Y
H	Developer	2	X
I	Developer	3	X
J	Developer	3	X
K	Project Manager	1	X
L	Researcher	1	Y

system matured over six versions and the project team consisted of 12 members across two geographic locations, five different roles, and three levels of seniority. The role, seniority, and geographic locations of the team members, whom we will call A through L are presented in the Table 1. (Note: 1 denotes highest organizational seniority; and X and Y are locations in two different continents, in India and the United States of America respectively).

Some of the relevant columns of the bug tracker were: *Issue ID, Assigned To, Issue Status, Priority, Due Date, System Version No., Creation Time, Created By, Modification Time, Modified By, Category, Comments, Content Type, Description, Related Issues, Title, Item Type, Path.* There were 323 rows of data.

Out of these, for a particular bug (*Issue ID*), information in the fields *Assigned To, Created By, Modified By* and *Comments* uniquely identified all the individuals who relate to a particular bug across the entire system. As mentioned earlier, this affiliation information was parsed out of the bug tracker using a set of Java programs and presented in a Pajek compatible input format (*.net file). The resulting network generated by Pajek has 12 vertices and 47 edges; it is depicted in Figure 3.

7.2 Presentation of the Measurements

Table 2 presents the measures of centrality for each vertex (x) in the metrics discussed earlier – relative degree centrality, $C_D(x)$; relative closeness centrality, $C_C(x)$; and relative betweenness centrality, $C_B(x)$. Figure 4 gives the graphical representation of these measures.

Table 3 shows for each team member, the number of bugs created by (RB), assigned to (AT), and contributed-to by (NB).

Table 4 presents the measures of sub-groupings in the network, core (C) and valued core (VC), the latter taking into account the weights of the lines. While calculating the valued core, the threshold of 25 and steps of 10 were considered for the weights of the lines.

Out of the total 323 instances of bugs being created (RB) and assigned to (AT), and 738 instances of team members contributing to bug resolution (NB), Figures 5, 6, and 7 respectively give the percentages in each category for each team member based on the information of Table 3.

Table 2: Measures of Centrality

Member	$C_D(x)$	$C_C(x)$	$C_B(x)$
A	0.55	0.69	0.00
B	1.00	1.00	0.09
C	0.64	0.73	0.01
D	1.00	1.00	0.09
E	0.55	0.69	0.01
F	0.91	0.92	0.05
G	0.55	0.69	0.00
H	0.64	0.73	0.00
I	0.55	0.69	0.00
J	0.82	0.85	0.02
K	1.00	1.00	0.09
L	0.36	0.61	0.00

Table 3: Involvement of Team Members in Bug Resolution Activities

Member	RB	AT	NB
A	163	21	171
B	8	69	162
C	80	0	88
D	2	56	72
E	0	11	10
F	0	15	16
G	27	2	27
H	6	34	52
I	7	1	7
J	2	102	74
K	22	12	53
L	6	0	6

Table 4: Measures of Sub-Groupings: Core (C) and Valued Core (VC)

Member	C	VC
A	6	7
B	6	7
C	6	6
D	6	5
E	5	0
F	6	1
G	6	1
H	6	3
I	6	0
J	6	7
K	6	3
L	4	0

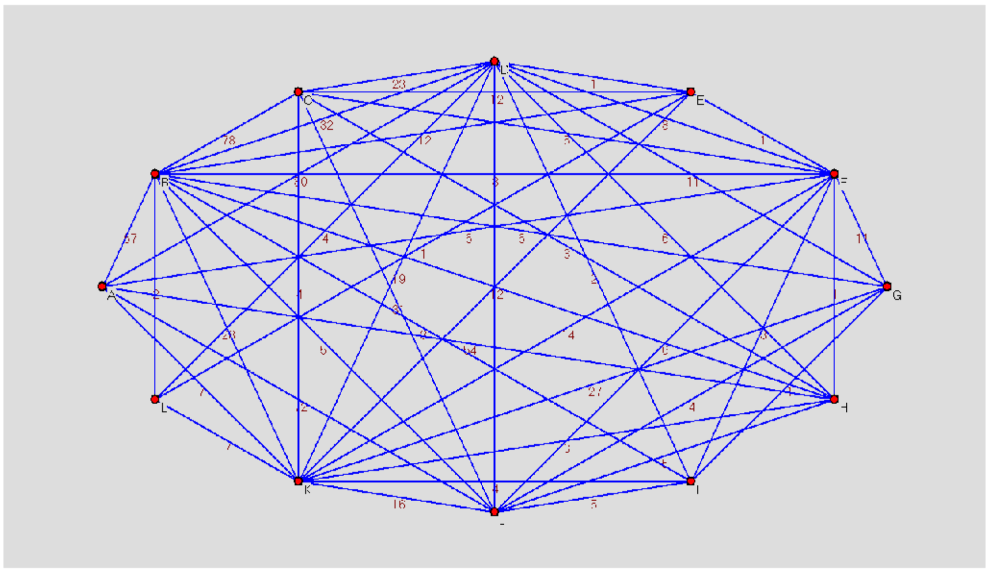

Figure 3: The Social Network Generated from the Bug Tracker

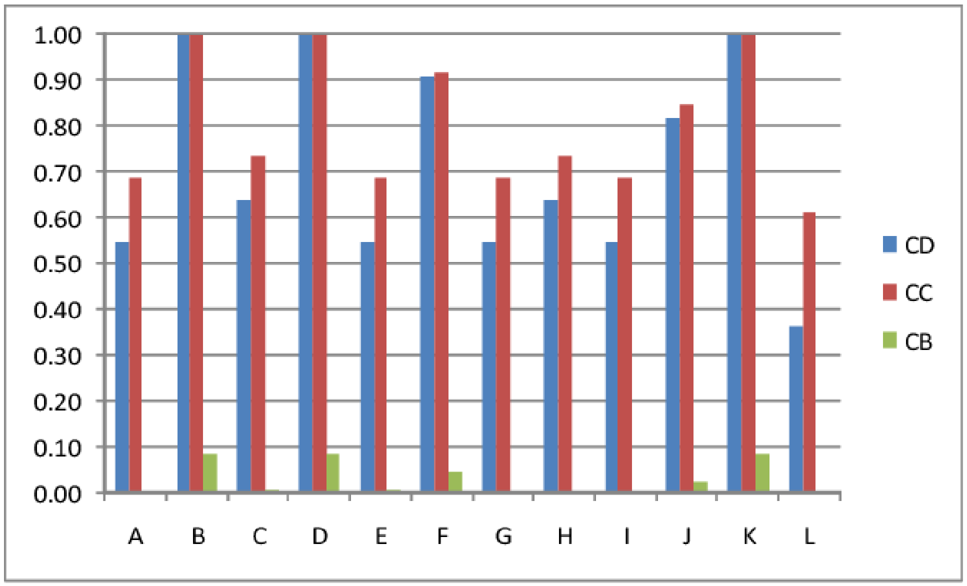

Figure 4: Centrality Measures

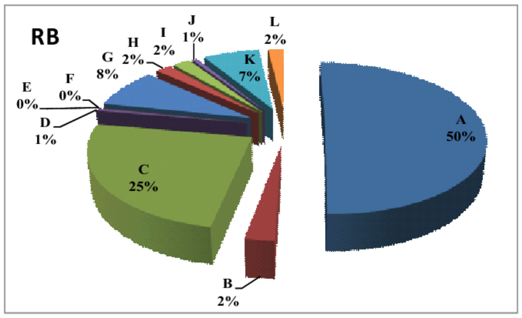

Figure 5: Percentage of Bugs Created By Team Members

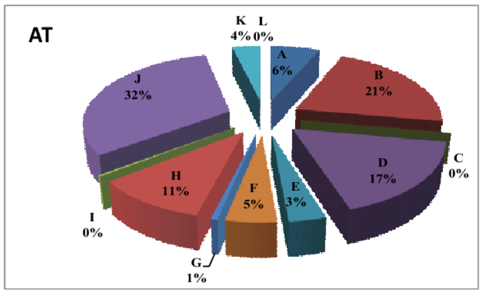

Figure 6: Percentage of Bugs Assigned To Team Members

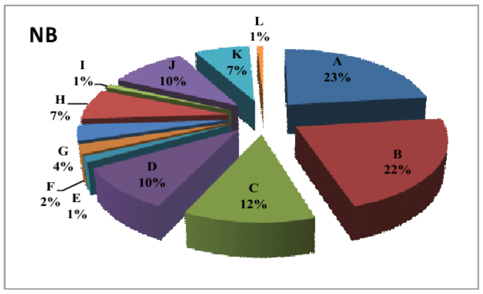

Figure 7: Percentage of Bugs Contributed-to by Team Members

7.3 Discussion

Interpretation of the above metrics values in light of team member data presented in Table 1 leads to the following observations:

- By all the measures of centrality, B, D, K – the tallest bars in Figure 4 – are the most central nodes. These are our omnis!

  B has the official role of architect and is at organizational seniority of level 2; D is a developer at level 3; and K is the project manager at level 1. All of them are at the same location. B and K expectedly play a central role. But D is an unexpected omni; officially (s)he is just a junior developer, yet the individual has a level of centrality similar to the project manager and the architect in the bug resolution activity.

- A, C, G have raised the most bug reports – 50%, 25%, and 8% respectively (Figure 5) – indicating they were most deeply involved in testing the system.

  A, C are officially in charge of quality assurance (both of them are at level 3 and at the same location) and their enhanced involvement in testing is expected. However, G is a developer at level 3 at a different location. His/her relatively high involvement in testing may indicate a deliberate decision to have the system tested by someone not co-located with the bulk of the team's developers.

- J, B, D have been assigned the most bugs – 32%, 21%, and 17% respectively (Figure 6).

  J and D are at the same level (3); both developers, and both at the same location(X). B being the architect was in overall charge of the design and it is expected that bugs will be assigned to them when initially raised. But it is surprising J and D has been assigned a considerable number of bugs, given their junior role. It is likely each of them was owning the development of a major part of functionality. As noted earlier, D also appeared as an unexpectedly central team member.

- A, B, C have participated most in bug resolution activities other than raising a bug or being assigned a bug – 23%, 22%, and 12% respectively (Figure 7).

  It is usual for initial assignees of bugs to reassign them – usually to developers – for the actual implementation of the "fixes" [16], [3]. A and C were in charge quality assurance, and B was the architect. So it is expected they will reassign the bugs to those more closely associated with development activities. But instead they seem to have worked on the bug resolution most themselves. This is an anomaly which can not be explained with the available data.

- Without considering weights of edges, everyone other than E and L are part of the largest 6-core; this implies these individuals were most loosely connected with the rest of the network.

  E is developer at level 3 and L is a researcher at level 1; the two are at different locations. E belongs to a 5-core, whereas L belongs to the smallest 4-core. This seems quite plausible as L was most likely not closely involved with the core activities of the bug resolution.

- Considering weights of edges, A, B, J form the largest valued core.

  This seems to indicate these individuals form the most close knit group within the team. All three are at the same location, two of them are at the same level (A, J at 3 and B at 2) and all three have different roles (quality assurance, architect, and developer respectively). While it is not unexpected that architect and quality assurance will be in close contact during the bug resolution activity, the presence of J in this group is surprising.

In summary, considering the entire network, we find that centrality measures reflect B and K to be omnis which positively correlate to their formal project roles and organizational seniority. Centrality measures as well as percentage of bug assignments indicate D is also an omni, which is surprising, given D's role and seniority. J's role and seniority also does not suggest (s)he is likely to be deeply involved in the bug resolution activity; yet J is, as indicated by his/her membership in the most important valued core as well as well as significant percentage of bugs being assigned to him/her.

So there is surely one team member (D) and probably another (J) whose level of involvement within the team as indicated by the social network analysis of the bug tracker information can not be reconciled with their official positions suggested by the project team member data. This is an instance of the lack of team congruence that has been revealed through automated social network analysis. This points to several open questions that need to be addressed in future work.

8. OPEN QUESTIONS & FUTURE WORK

We will first recognize the open questions and then discuss planned future work.

What if, there is a subject matter expert who has facilitated several bug resolutions by informally advising developers (may be going physically from desk to desk, as it often happens amongst co-located project team members)? Such an individual's references in the bug tracker data will not fully reflect his/her actual contribution. Our current analysis does not take into account such a commonly occurring situation.

Moreover, it is well established that larger the network, the more insightful is the social network analysis [13]. How well will our method and observations scale when we take a bug tracker with many more rows of data than the one we have studied?

Currently we have only considered edges (undirected links). In a real life project, the flow of information (as well as authority) is often directed. So, how does our approach need to be adjusted when the edges are replaced by arcs (directed links)? In such case do we also need to consider measures of prestige [13] in addition to measures of power for identifying the omnis?

In our future work we plan to address these questions by studying large scale open source systems and other project artifacts in addition to bug trackers. We also intend to explore whether and how team dynamics in general vary between open source systems and proprietary ones.

Additionally, correlation of the degree of team congruence to project outcome metrics (e.g. delivered defect rate, cost and schedule variance, cost of quality activities etc.) are planned. Within a single project, we would like to create team congruence scores from different dimensions – such as groups, events, artifacts – to get deeper insights on how some individuals may be consistently aligned differently than other team members. We also plan to expand the scope of this study to explore whether similar trends manifest across similar groups and events in dissimilar project situations, and whether correlations between identification of the omnis, degree of team congruence, and project outcomes hold. This will facilitate better planning of future releases and/or in-process optimization of the team.

9. CONCLUSIONS

In this paper, we have studied the bug tracker of a real life proprietary software system to understand its team dynamics. We extracted a individual-versus-bug affiliation network from the bug tracker data, converted it into a social network of the team's members, and used measures of centrality, subgrouping, and other topical information to identify the omnis – individuals who tended to be omnipresent and omniscient in the bug resolution initiative, and thus emerge as concentration points of the project's execution know-how. Our interest in the omnis was inspired by a quest to better understand team congruence – whether each team member's contribution is commensurate with his/her role, organizational seniority, and location, as recognized in the formal project plan.

By analyzing the social network derived from the bug tracker, we found that while some individuals played expectedly central roles – such as the project manager, and the architect – there were other team members (junior developers) whose involvement was significantly larger than what their formal roles and organizational seniority seemed to suggest. Thus in the system studied, automated social network analysis pointed to some aspects of the lack of team congruence. Our approach also provides a general mechanism to identify individuals who serve as knowledge centers in a project's execution map; this can be useful for planning knowledge transition activities and knowledge management tasks.

In our future work, we plan to address some of the open questions surrounding the current work and correlate the degree of team congruence with project outcome metrics (such as defect rate) to gauge the effectiveness of a team's current structuring and guide future reorganization.

10. REFERENCES

[1] Burton group field research study: Social networking within the enterprise, 2009.

[2] Amrit, C., Hillegersberg, J., and Kumar, K. A social network perspective of conway's law. In Proceedings of the CSCW Workshop on Social Networks, Chicago, IL, USA, November 2004, 2004.

[3] Aranda, J., and Venolia, G. The secret life of bugs: Going past the errors and omissions in software repositories. In *Proceedings of the 2009 IEEE 31st International Conference on Software Engineering* (2009), IEEE Computer Society, pp. 298–308.

[4] Batagelj, V., and Mrvar, A. Pajek: Analysis and visualization of large networks. In *Graph Drawing*. 2002, pp. 8–11.

[5] Batagelj, V., and Mrvar, A. Introduction to social network methods. `http://vlado.fmf.uni-lj.si/pub/networks/pajek/`, 2009.

[6] Bird, C., Gourley, A., Devanbu, P., Gertz, M., and Swaminathan, A. Mining email social networks. In *Proceedings of the 2006 international workshop on Mining software repositories* (Shanghai, China, 2006), ACM, pp. 137–143.

[7] Breiger, R. The duality of persons and groups. *Social Forces 53*, 2 (1974), 190, 181.

[8] Cain, B., Coplien, J., and Harrison, N. Social patterns in productive software development organizations. *Annals of Software Engineering 2*, 1 (Dec. 1996), 286, 259.

[9] Chen, I., Yang, C., Lu, T., and Jaygarl, H. Implicit social network model for predicting and tracking the location of faults. In *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference* (2008), IEEE Computer Society, pp. 136–143.

[10] Dietrich, J., and Jones, N. Using social networking and semantic web technology in software Engineering–Use cases, patterns, and a case study. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian* (2007), pp. 129–136.

[11] Ferligoj, A., and Mrvar, A. Analysis and visualization of social networks. `http://mrvar.fdv.unilj.si/sola/info4/programe.htm`, 2005.

[12] Freeman, L. A set of measures of centrality based on betweenness. *Sociometry 40*, 1 (Mar. 1977), 41, 35.

[13] Hanneman, R. A., and Riddle, M. Introduction to social network methods. `http://www.faculty.ucr.edu/~hanneman/nettext/`, 2005.

[14] Humphrey, W. S. *TSP: Leading a Development Team.* Addison-Wesley, 2006.

[15] Huxley, A. *Brave New World*, reprint ed. Harper Perennial Modern Classics, Sept. 1998.

[16] Jeong, G., Kim, S., and Zimmermann, T. Improving bug triage with bug tossing graphs. In *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering on European software engineering conference and foundations of software engineering symposium* (Amsterdam, The Netherlands, 2009), ACM, pp. 111–120.

[17] Kleinberg, J. The Small-World phenomenon: An algorithmic perspective. *IN PROCEEDINGS OF THE 32ND ACM SYMPOSIUM ON THEORY OF COMPUTING* (2000), 163—170.

[18] Kleinberg, J. M. Navigation in a small world. *Nature 406*, 6798 (2000), 845.

[19] Lattanzi, S., and Sivakumar, D. Affiliation networks. In *Proceedings of the 41st annual ACM symposium on Theory of computing* (Bethesda, MD, USA, 2009), ACM, pp. 427–434.

[20] Leskovec, J., Kleinberg, J., and Faloutsos, C. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (Chicago, Illinois, USA, 2005), ACM, pp. 177–187.

[21] Lopez-Fernandez, L., Robles, G., Gonzalez-Barahona, J. M., and Carlos, J. Applying social network analysis to the information in cvs repositories.

[22] Okoli, C., and Oh, W. Investigating recognition-based performance in an open content community: A social capital perspective. *Inf. Manage. 44*, 3 (2007), 240–252.

[23] Pinzger, M., Nagappan, N., and Murphy, B. Can developer-module networks predict failures? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering* (Atlanta, Georgia, 2008), ACM, pp. 2–12.

[24] Puppin, D., and Silvestri, F. The social network of java classes. In *Proceedings of the 2006 ACM symposium on Applied computing* (Dijon, France, 2006), ACM, pp. 1409–1413.

[25] Sabidussi, G. The centrality index of a graph. *Psychometrika 31*, 4 (Dec. 1966), 581–603.

[26] Watts, D. J., and Strogatz, S. H. Collective dynamics of /‘small-world/’ networks. *Nature 393*, 6684 (June 1998), 440–442.

[27] Xu, J., and Chen, H. The topology of dark networks. *Commun. ACM 51*, 10 (2008), 58–65.