Studying Team Evolution during Software Testing

Vibhu Saujanya Sharma
Accenture Technology Labs
Accenture
Bangalore, India
vibhu.sharma@accenture.com

Vikrant Kaulgud
Accenture Technology Labs
Accenture
Bangalore, India
vikrant.kaulgud@accenture.com

ABSTRACT

Software development teams are one of the most dynamic entities of any software development project. While the individuals are assigned planned roles at the start of any project, during the course of the project, the team constitution, structure, relationships and roles change. Such changes are often spontaneous and constitute the evolution of the team along different phases of the software development lifecycle. As software development is a team effort, these dynamics may have a significant effect on the development lifecycle itself. This work is aimed at studying the evolution of project teams and gathering insights that can be correlated with project health and outcomes. In this study we apply social network analysis techniques to investigate team evolution in a project in its testing phase. While the questions and insights that we investigate in this paper are valid and useful for all phases of the software development lifecycle, we have focused on software testing phase as it one of the most critical phases in the lifecycle. Our work aims to provide insights in the changes in team interactions and individual roles as the testing process continues and can help find if the same is aligned to the planned and desired project behavior.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Programming Teams*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management; D.2.8 [**Software Engineering**]: Metrics

General Terms

Human Factors, Management

Keywords

Team Analysis, Team Evolution, Social Network Analysis, Software Testing

CHASE'11, May 21, 2011, Waikiki, Honolulu, HI, USA

1. INTRODUCTION

The software development lifecycle (SDLC) is inherently a people intensive process. Different phases of SDLC depend heavily on the individuals who contribute to various tasks as well as how they work together as a team. Software teams typically consist of individuals at varying levels of organizational seniority, different project roles and skills, and possibly located at diverse geographies. At the start of a software project, individuals are assigned specific responsibilities depending on the above factors. However as the project progresses, the team structure evolves spontaneously. A few individuals become more central to the team, while others' contributions might diminish. Further, individuals may switch from the planned roles and participate in project activities in an ad-hoc manner. Different clusters of individuals may form and dissipate as the project progresses and this may be totally unplanned too. As any software development project progresses, the evolution of the individuals and the team itself would have significant effect on how different phases of the SDLC fare and it becomes important to characterize and study this.

Note that there are many metrics traditionally associated with different phases. Take for example, software testing which is a crucial phase of a software delivery project. Ineffective testing leads to expensive rework effort and often times, to defective software being released to customers. There are several process metrics such as defect injection rate, defect closure time and defect reopen rate that comment on the outcome of the testing effort. Now consider a query such as - 'Why is the defect closure time increasing?'. Typically, one starts looking at process data and measurements to try to answer this. However it may so happen that many defect resolvers are taking up ad-hoc duties as a defect detector and thus reducing the available effort for defect closure, thereby increasing defect closure time. Traditional project metrics would not be able to capture this and need to be complemented with an in-process study of team dynamics to effectively answer such questions. In this ongoing work, we focus on characterizing team and individual evolution and gaining such insights in the process.

For this preliminary study, we focus on software testing phase as it is a significantly team effort intensive phase, requiring collaboration within the testing team as well as across the testing and development teams. Although task allocation is done at the start of the testing phase, with specific resources allocated to testing and defect resolution, often times due to project pressures and individual choices, ad-hoc task reallocations happen. Such reallocations, if not

monitored and appropriately corrected, could have a detrimental impact on the testing effectiveness.

Taking a different approach to monitoring and controlling testing process effectiveness, we focus on characterizing individual resources and team structure and correlating their evolution with project outcomes. The hypothesis is that there are 'clues' available through team analysis that a project / test manager could use to optimize the testing effort. We utilize the defect tracking data to extract the team interactions and apply social network analysis techniques to study the team behavior over time.

Note that there is a body of related work studying people aspects of software development. A study of the life cycle of bugs from a social and organizational perspective has been presented in [2]. In [9], the authors explore the use of developer-module networks to investigate the relationship between the fragmentation of developer contributions and the number of post-release failures. Knowledge sharing within the software engineering community based on social networking has been explored in [7]. An interesting work on extraction of social networks from emails has been presented in [4]. More recently, in [8, 5] the authors have studied the effect of organizational structure and team distribution respectively, on software quality. Our approach is different in that unlike these approaches we do not limit the study to the planned or static team structure but rather extract the *dynamic* structure of the team based on individual interactions and investigate its evolution over time. Moreover rather than a point study, we study the dynamic nature of the network of individuals in a software project and aim to correlate the network evolution with project outcomes. This study builds upon our earlier work [6] and extends the scope and set of insights that we wish to explore along with a different experimental setup.

The paper is organized as follows: The next section presents our approach to characterize software testing teams for analysis. Section 3 elaborates on the current experiment and some early results that we have obtained. We conclude the paper in Section 4.

2. CHARACTERIZING TESTING TEAMS

Testing teams are characterized at an individual level and at the team level. This two-level characterization allows rich comprehension of the social dynamics of a testing team and finer grained correlations with outcome metrics.

2.1 Individual Resource Characterization

Individual resources are characterized by static and dynamic attributes. Static attributes are typically organization and project specific. Key static attributes that we use to characterize individuals in a testing project are geographic location of the resource, seniority or organizational level to which resource is assigned and the 'Assigned Role' (e.g. tester, test architect, developer)

Dynamic attributes are those whose values change as the testing phase progresses. A key dynamic attribute is 'Played Role'. E.g. in the testing phase, the 'Played Role' could be a *defect detector* or *defect resolver*. This attribute's value changes as per the activities performed by an individual during the testing phase. We use the defect tracker logs for labeling an individual as a detector or resolver at any given instant based on the frequency of her participation in logging new defects or being assigned defects to resolve.

2.2 Team Level Characterization

Team level characterization leverages the notions of explicit and implicit interactions. Explicit interaction happens when two or more individual collaborate using emails, instant messaging etc. Implicit interaction occurs by virtue of two or individual working on same artifacts - e.g. test case, defects etc. Based on these interactions a social network emerges within the testing team. Team level characterization is done using established social network analysis metrics. The key attributes are the eigenvector centrality of individuals and cluster membership of individuals.

3. CURRENT EXPERIMENT AND EARLY RESULTS

We analyzed 2 months of defect data from a real-life testing project. The defect data has details on defect detection - detected date, who detected the defect, severity and priority of defect and defect closure - to whom was the defect assigned and when was it closed. Based on this data we first identified all individuals participating in the testing phase and their characterization. Also, using standard social network analysis tools [3, 1], we generated the key team level characteristics. The individual and team level characteristics were generated for each day of the 2 month period to study the 'evolution' of these characteristics.

3.1 Key Questions and Insights

Based on this characterization and our interactions with different project managers and team members, we elicited a set of pertinent questions and associated insights that we want to investigate. These are:

- Is there a temporal pattern in cluster formation in typical testing projects?

  ⇒ This is an important question to study how individuals align themselves as the activities progress. This can help understand if those clusters are forming due to interpersonal issues, geographic spread, or more importantly do the clusters reflect the structure of the activity/artifact under purview. Evolution of a cluster might also be closely tied to set of project activities being performed and metrics which characterize that activity might be highly correlated with this evolution.

- Can we identify high performing clusters and use that knowledge to restructure testing teams for effective testing?

  ⇒ While interacting with different project groups, we found that in almost every case, a few informal team clusters consistently would perform better than the others. However how and why exactly that happens is not immediately clear. The aim thus is to be able to characterize the evolution of high performing clusters and then utilize this for re-structuring teams.

- How do individuals evolve over period of time? Do they settle into specific roles? Does evolution impact an individual's efficiency?

  ⇒ While a project might start with a well defined set of roles and responsibilities for each individual, over time, team members may act as expertise mavens, knowledge brokers, or may be omnipresent in many

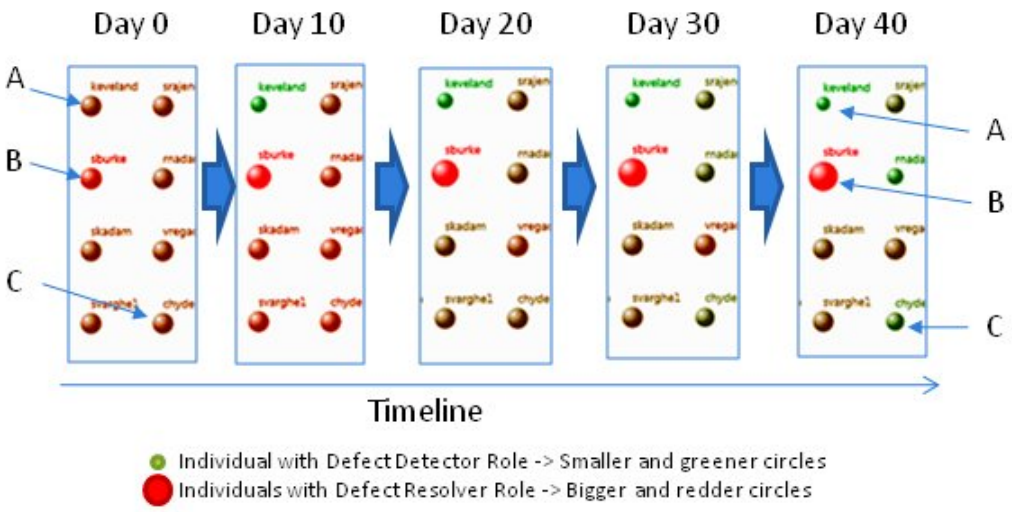

Figure 1: Change in Individual Roles over 2 months with a 10 day increment

activities and interactions and thus become central to different activities - both literally and in the graph-theoretic way. It is key to identify this evolution as this would heavily influence individual efficiency and effectiveness and thus, the success and direction of the project.

- Does the deviation of 'Played Role' from the 'Assigned Role' have an impact of outcome metrics?

  ⇒ As individuals continue to deviate from the planned set of roles, project activities and their outcomes may be affected. Studying what is the deviation and how does that relate to the outcome metrics thus becomes very pertinent.

- Are there individuals who are 'overly' important and connected while others in the team are 'disconnected'?

  ⇒ Certain individuals may become prominent and more important than others intermittently while others may consistently play a very central part throughout (and vice-versa). Disconnected team members may need special attention through focused interaction or assignments, or need to be better equipped through training if need be. One can plan and schedule better if one understands who are these key individuals and can organize efforts accordingly.

Note that these insights are valid and useful through the different phases of the software development lifecycle. Till now though, we have aligned and applied these specifically to study software testing.

3.2 Early Results

Although this work is still under progress, we have already got some useful results. We have studied aspects such as the evolution of the testing team itself in terms of the attribute associated with the defects such as *severity* and *business priority*. We also studied how people evolved in their roles as defect detector or resolver and how their centrality to the testing effort changed.

In Figure 1, we show the evolution of the role of an individual. The grid vertices represent different personnel and the size and color of each vertex represents the role based on the type of jobs performed till a particular day. Figure 1 shows the change in the roles over a period of time with each snapshot taken after roughly 10 days. In particular, notice the three individuals labeled **A**,**B** and **C**. Both **A** and **B** settle down quickly in their roles as defect detector and defect resolver. However, **C** initially seems to become a resolver, but gradually gravitates towards becoming a detector. Though this is only a small subset of all the individuals, we saw a similar patterns in others too - some quickly settle in a particular role, while some others take a long time, or keep on shifting from one role to the other. The latter behavior most probably is not as planned, and it might be problematic for an 'average' project. There are however, exceptions and this might be perfectly acceptable behavior. The important point is that such patterns can be surfaced from the data and exposed to the project manager who can then apply the right project context to take appropriate decisions.

We used the eigenvector centrality measure to determine an individual's "importance" in the team. Since the eigenvector centrality assigns higher score to a node connected with other high score nodes, the net relative score suggests an implicit hierarchy in the team based on the parameters chosen to build the team's social network. Thus, eigenvector centrality comments on the 'global importance' of an individual in the team. We further used NodeXL to color and size the vertices based on their eigenvector centrality. As shown in Figure 2 green colored vertex (biggest circle) represents the most important resource based on the eigenvector centrality. Notice that this resource remains important throughout a 30-day period. It is important for the project manager to ensure that such individuals remain with the testing team for the entire duration and are shadowed for knowledge dispersal.

Note that we have not considered information regarding the geographic location and demographic data of the team members in this study unlike our recent study [6] where it

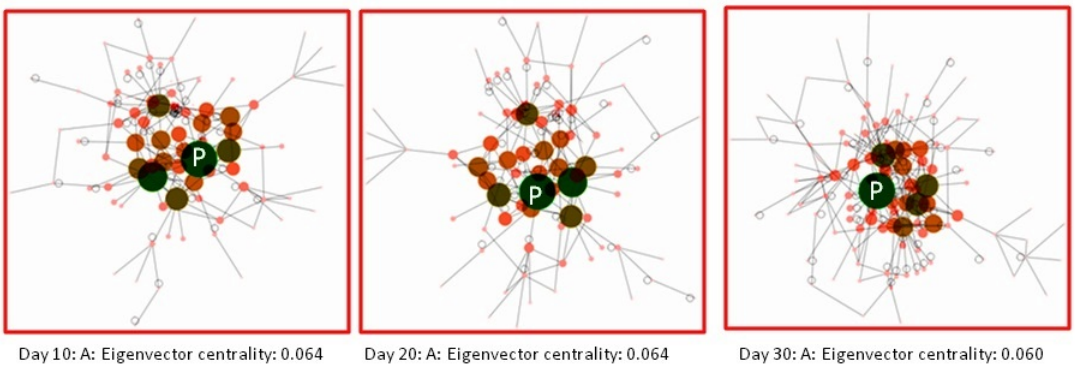

Figure 2: Change in Eigenvector Centrality of Individuals over time

allows us to gain further related insights. We plan to extend this study to include such data. Furthermore, this study is only based on data collected from the project's defect tracker, and we have not included any auxiliary information regarding the team and project events happening as this data was collected. Interviewing team members to understand their perceptions and the project's ground realities, and trying to correlate the analysis results with these will also be an avenue for extending this work.

4. CONCLUSION AND ONGOING WORK

Software development activities can be heavily affected by team dynamics. Specifically outcomes of critical phases like software testing and defect resolution can be significantly influenced by how the team forms, collaborates, and evolves. While the scope of research is across the entire development lifecycle, this specific work is aimed at studying the evolution of a testing team over the course of time and how that affects the outcomes of the testing and defect resolution process as a whole. For this purpose we first characterize individuals and the team in terms of measurable attributes and interactions. We then utilize social network analysis and study the evolution of the individuals and the team by asking pertinent questions regarding them.

The surfacing of the *seemingly chaotic* nature of roles played by individuals is interesting. While development phases like testing needs individuals to be agile in terms of tasks performed, too much agility can be detrimental, especially in crunch situations. The other (intuitive) fact that emerges is the presence of "important" individuals. One interesting question is 'Are these important individuals playing a stable role or do they change roles frequently?' Further analysis regarding important individuals based on nature of defects handled, defect location, role played etc. will help in identifying team structure patterns, their impact on project outcomes and knowledge sharing opportunities. Furthermore, it will be interesting to observe the role dynamics in other development phases, especially in 'coding' and also observe roles played by individual across phases. Does continuity in terms of working in same modules across design, code, and test help improve individual performance and project outcomes? Accumulating such insights will help teams to plan their team structure and individual responsibilities better.

We are continuing on the research to provide answers to the aforementioned questions to discover typical patterns of team dynamics in the testing and defect resolution phase.

5. REFERENCES

[1] Nodexl: Network overview, discovery and exploration for excel. 2010.

[2] J. Aranda and G. Venolia. 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*, pages 298–308. IEEE Computer Society, 2009.

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

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

[5] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does distributed development affect software quality?: an empirical case study of windows vista. *Commun. ACM*, 52:85–93, August 2009.

[6] S. Datta, V. Kaulgud, V. S. Sharma, and N. Kumar. A social network based study of software team dynamics. In *Proceedings of the 3rd India software engineering conference*, ISEC '10, pages 33–42, New York, NY, USA, 2010. ACM.

[7] J. Dietrich and N. Jones. 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*, pages 129–136, 2007.

[8] N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality: an empirical case study. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 521–530, New York, NY, USA, 2008. ACM.

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