Combined Methods, Thick Descriptions: Languages of Collaboration on Github

Nicholas M. Weber
Graduate School of Library and Information Science
University of Illinois, Urbana- Champaign
nmweber@illinois.edu

ABSTRACT
Like many professional work activities in this age of ubiquitous computing and high-speed internet connections, computer programming and software development are increasingly mediated by systems with ‘social media’ features like profiles, avatars, ‘liking’, and commenting capabilities. When working on shared tasks, programmers have effectively leveraged these capabilities to overcome differences in time and location while simultaneously using collaborative web applications, such as version control repositories like SCM or ‘git’ systems to work together more efficiently. Here we present preliminary findings from a project investigating patterns of collaboration on the social coding platform Github. We’ve used a research method that combines the use of statistical approaches from social network analysis (SNA) and traditional qualitative case study construction. Our results show that this method is useful in qualitatively explaining the topology of a collaborative network, especially the formation of cliques that have been identified using traditional SNA metrics.

Keywords
Collaborative Work, Social Network Analysis, Research Methods.

INTRODUCTION
The sharing, reuse and repurposing of computer code has been greatly aided by innovation in both infrastructure (e.g. high speed broad-band), and the capability of cloud services dedicated to software development (e.g. content management systems, source control management systems etc.) As these systems and infrastructures transform the way that computer programmers and software engineers are capable of working together, we as social scientists also have a greater ability to study, and understand performance in these networked arrangements by analyzing digital traces (Hine 2005) of collaboration through both direct observation of user activities, and analysis of audit trails of activity, such as a system’s data log.

Background
Git’s are quite simply a form of software control management (SCM). When implemented in a networked environment, a git schema allows users and teams of programmers to submit (commit), combine (branch), contribute (push) and obtain (fork) repositories of computer code that are generally hosted and managed by a third party. The git scheme for version control usefully provide a kind of backwards compatibility that allows different portions of code to be worked on simultaneously, while also guaranteeing the fidelity of an original code repository.

Github, is an online platform that offers users free repository hosting for their code (managed, as the name would imply with gits), as well as social networking features common across the web, like the ability to follow users through RSS, comment on changes or updates to a repository, and even solicit help by posting code snippets to a user forum. As a social network, Github is traceable via the log of activities that a user participates in, however most studies of collaboration on Github have either used qualitative methods (Dabbish et al 2012) or quantitative methods (Heller et al 2011) in isolation. In order to better understand how the ‘social’ functions of a system like Github affect collaboration, we’ve chosen to combine pieces of these two methodological research approaches to create what most ethnographers call a ‘thick description’ (Geertz, 1973) of these activities by first assembling, and then analyzing digital traces (Geiger and Ribes, 2011) of Github activity.

DATA
This dataset was originally gathered by Franck Cuny as part of the 'Stargit' project1. Using the Github developer API, profile data were gathered (n = ~120,000) for all newly registered users (2009-2011) of Github. User profiles with >= 1 repository that had been forked (indicating there is some other user interested in either improving or re-using the code) were kept in the

1http://lumberjaph.net/community/2011/06/20/stargit.html
dataset- and profiles capable of being geo-referenced were further sorted \((n = \sim 40,000)\) using the location referencing service GeoAPI.

Most user profiles on Github include some combination of the following information: Github handle, real name of user (redacted for this study), number of followers, number of fork requests, user’s location (nation), main programming language of the user, and number of repositories owned (and hence made publicly available). For the SNA portion of this study, nodes represented individual users (as opposed to repositories). Edges in this dataset are directed, and weighted. Directed edges represented a connection between two users by way of watching, forking or sending a pull request to another user. These Github activities are generally the way that the system records a trace of people working together, or expressing interest in one another’s repository. The weight of edges were determined by the number of repositories that have been forked or pull requests sent to that particular user.

We further sub-setted this data based on the user’s main programming language, as specified by their Github profile. (Note: a limitation of this method is that many programmers host repositories with code in languages other than their ‘native’ language designated in a profile). For this study we choose the programming languages PhP, Python, and Perl based on their popularity in overall Github repositories, and the manageability of effectively analyzing these networks compared with much larger Java and Ruby communities (see Figure 1).

**METHODS**

Our data was first loaded into the network visualization software Gephi (2009), using the forceatlas2 layout\(^2\) for the language networks we could then explore the formation of cliques (or sub-groups), and further statistically analyze both the network and the individual cliques separate from the network. Importantly, the forceatlas2 layout is a ‘linear-linear model’ where attraction and repulsion are proportional to distance between nodes. So this allows for what Hanneman and Riddle (2001) call a top down approach to qualitatively identifying cliques, or as they explain, “... differences in the ways that individuals are embedded in the structure of groups within in a network can have profound consequences for the ways that these actors see their ‘society,’ and the behaviors that they are likely to practice.”

---


---

![Figure 1 Breakdown of most popular languages (number of repositories hosted = ~3 million) on Github as of 06/2010.](image)

In the programming language graphs, we can observe cliques of connected nodes that are farther from the central network cluster. Isolating these cliques, we then measured the average weighted degree centrality, the density, the connected components (weak ties), the average path length and the number of shortest paths and compared these numbers against those of the programming language network as a whole. For individual nodes, within each clique, we also calculated a weighted degree centrality, closeness centrality and a betweenness centrality (individual node metrics are available as supplementary material.)

**PRELIMINARY RESULTS**

*For the sake of this abstract, we’ll discuss the results of one clique in one language (Python). We however have provided the full visualization and metrics for the analysis of three additional cliques from that graph (figure 2).*

**Python**

Python’s network is quite large, consisting of 3862 nodes, with over 8000 connections. The density of the graph however is quite sparse at .0001. Python has an ever-increasing user community, and is best known as the workhorse of the object oriented programming paradigm. In this network we see distinct communities emerging from the larger network (55), which is a much higher number than our other languages on Github (PhP = 11; Perl = 4) The connected components in the Python network is very large at 544, with the network diameter (average shortest distance between pairs of nodes) being 20. This implies that overall many python users are loosely connected to one another (meaning that users could more easily exploit weak-ties), but even more important for our work here, there exist many communities of densely connected users with strong connections (implying frequent interaction and collaboration amongst those closest in network distance).

Clique one (Figure 3) is a small cluster, where users plaes, nekahayo and certik have the highest betweenness scores-meaning that they hold important roles as connecting or
bridging between members of this clique. Interestingly, when investigating this clique’s users on Github profiles and their linked webpages, we discovered that plaes and certik are physicists and of Estonian nationality – they’re also an uncle / nephew in real life. Additionally, of the nodes with the highest three closeness measures (implying an inflated ability for actions to see and be seen within the network), one is a developer of a multiplatform desktop environment GNOME\(^3\) used often by physicists and, the other two are also physicists collocated with user certik – in Reno, Nevada. Of the 29 nodes, 20 are of Estonian nationality. Regmi, the user with the highest closeness centrality in this clique (2.8125) is also exceptionally well connected (490) in the larger python network (implying he plays a bridging role to the external network) and interestingly he is also a physicist.

DISCUSSION
From the metrics used to isolate this clique, and our analysis of users within the clique we were able to offer intuitive explanations as to why certain nodes held prominence (head of a scientific lab, developer of a platform etc.) as well as why the density of the clique overall might be so high (common nationality, co-location and professional affiliation). Assembling these traces gives us a ‘thicker’ description for explaining collaboration, but we want to stress the ethnographic ethos that these descriptions are not causal explanations; the contextual information we were able to gather from personal traces left within this and other systems does not cause the clique to form, but it does imply that these factors, in addition to a shared programming language, are important in how and why people work together in an open, and social platform like Github.

FUTURE WORK
In future work, we will complete the clique isolation and metric analyses for all three programming languages. Additionally, we plan to use this strategy in future Github studies to supplement traditional qualitative sampling methods used in survey and interview recruitment. One important aspect of collaboration that has not been considered here are Github activities that have a temporal dimension, or a time stamp. The success, frequency of activity, impact, or popularity of a repository / project may be signaled by how often, at what times or even how regular the patterns of committing and branching code are within a network of programmers- all of which are potential avenues for future studies of collaboration between computer programmers.

ACKNOWLEDGMENTS
The author would like to thank Franck Cuny for sharing his data, Jana Diesner for her advice and guidance. All data associated with this project are available through a CC0 license at wiki.nicwe.be/r/

REFERENCES


\(^3\) http://library.gnome.org/misc/release-notes/2.26/

Figure 3. Python clique of Estonian physics platform developers (note numerous triadic closures)
Table 1. SNA Metrics for the Complete Network Graph and Sub-graphs

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>Python_1</th>
<th>Python_2</th>
<th>Python_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>3862</td>
<td>29</td>
<td>11</td>
<td>30</td>
</tr>
<tr>
<td>Edges</td>
<td>8099</td>
<td>79</td>
<td>56</td>
<td>97</td>
</tr>
<tr>
<td>Avg. Weighted Degree</td>
<td>5.708</td>
<td>2.69</td>
<td>5.5</td>
<td>3.233</td>
</tr>
<tr>
<td>Graph Density</td>
<td>0.001</td>
<td>0.096</td>
<td>0.611</td>
<td>0.111</td>
</tr>
<tr>
<td>Modularity</td>
<td>0</td>
<td>0.248</td>
<td>0.079</td>
<td>0.194</td>
</tr>
<tr>
<td>Connected Components</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Weak)</td>
<td>544</td>
<td>6</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Avg. Clustering Coefficient</td>
<td>0.086</td>
<td>0.119</td>
<td>0.74</td>
<td>0</td>
</tr>
<tr>
<td>Number of communities</td>
<td>55</td>
<td>23</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>Average Embeddedness</td>
<td>1.615</td>
<td>5.577</td>
<td>5.655</td>
<td>5.691</td>
</tr>
<tr>
<td>Average Path length</td>
<td>7.605</td>
<td>1.702</td>
<td>1.358</td>
<td>1.9775</td>
</tr>
<tr>
<td>Number of shortest paths</td>
<td>1492947</td>
<td>148</td>
<td>81</td>
<td>244</td>
</tr>
</tbody>
</table>

Figure 2. Full Python network visualization, with three cliques isolated. The statistical analysis of these sub-graphs are provided above.