Individualized Socio-Technical Congruence

Patrick Wagstrom
Engineering and Public Policy
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA, 15213 USA
pwagstro@andrew.cmu.edu

James Herbsleb
Institute for Software Research
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA, 15213 USA
jdh@cs.cmu.edu

ABSTRACT

At a macro level, congruence between a team's actual communication and coordination dependencies, a metric called socio-technical congruence (STC), is related to higher team performance. However, at an individual level, the calculation of STC may seem opaque, leaving individuals confused about how their communication patterns affect the STC metric and impact overall team performance. We propose two methods of calculating individualized STC based on previous work in the field. Examining this relationship further, we find that when the individualized STC metrics are broken into their constituent parts, individuals with high amounts of coordination requirements tend to have higher performance, while individuals with higher levels of communication have lower performance.

1. INTRODUCTION

Complex non-routine intellectual work, such as software engineering, is comprised of many tasks with large amounts of explicit and implicit dependencies. Accomplishing these complex tasks successfully requires individuals to collaborate not only with those on their assigned tasks, but also with individuals working on related tasks. By examining the network of task dependencies and individuals who worked on each task, a network of required communication is created. Work comparing networks of required communication to the actual communication in organizations, a metric called socio-technical congruence (STC), found that teams with high congruence had increased performance, as measured by a decrease in software defect resolution time[2].

This original formulation of STC treated all links as equal – meaning that the frequency of communication and amount of required communication between individuals was not taken into account when calculating a score. Later, a reformulation of STC in a graph theoretic framework provided a method to address valued edges in a network and also laid the foundation for providing meaning to values in the network[5]. From a functional perspective, when a network consists of binary only values, these two methods are identical and make similar claims about the impact of STC on overall team performance.

More recently, there has been increased interest in providing tools for end users to gain awareness of the overall communication and coordination network structure and guide users toward improving overall communication and congruence[1]. For a user to take the time to utilize such a tool, there must be sufficient individual benefit to overcome the cost of learning and utilizing the tool. In the case of STC, the output metric is provided only at the team level, leaving little context for an individual to understand their role in improving coordination within the team. In this work, we propose two new metrics to help individuals better understand congruence: Unweighted Individual Congruence (UIC) and Weighted Individual Congruence (WIC). We then provide analysis of the metrics and address possible issues with tools that attempt to increase overall congruence and the implications for individualized congruence.

2. MATHEMATICAL FOUNDATION

For the purposes of this work, we utilize the original notation and formulation of the STC metric as provided by Cataldo et. al. although the method could be easily translated to a graph theoretic framework[2]. Briefly, given a team with a network of task dependencies, $\mathbf{D}$, such that $D_{i,j} = 1$ if there is a dependency between task i and j, or 0 otherwise, and a network of task assignments, $\mathbf{A}$, such that $A_{i,j} = 1$ if developer i was assigned to task j we can calculate the coordination requirements $\mathbf{C_R} = \mathbf{A} \times \mathbf{D} \times \mathbf{A^T}$. The overall congruence of the network is then calculated as the proportion of edges that exist in $\mathbf{C_R}$ that are also observed in an actual communication network, $\mathbf{C_A}$.

Unweighted Individual Congruence, UIC is calculated in a very similar method to overall STC. A coordination requirements matrix $\mathbf{C_R}$ is generated for the entire network as is the actual coordination matrix $\mathbf{C_A}$. Unweighted individual congruence for an individual is then calculated by observing the congruence between $\mathbf{C_R}$ and $\mathbf{C_A}$ only for those edges in $\mathbf{C_R}$ that are incident upon the person in question. Formally, for an ego i, we have:

$$UIC_i = \frac{\sum (\mathbf{C_R}[i,] \times \mathbf{C_A}[i,]) + \sum (\mathbf{C_R}[,i] \times \mathbf{C_A}[,i])}{\sum \mathbf{C_R}[i,] + \sum \mathbf{C_R}[,i]} \quad (1)$$

We assume for this equation that all cells in the matrices have been dichotomized to 0,1 and that diagonals are zero. All multiplications of vectors are pairwise multiplications and $\mathbf{C_R}[i,]$ and $\mathbf{C_A}[i,]$ represent row i of their respective

STC 2008 Leipzig, Germany

networks in matrix form.

Weighted individual congruence is slightly more complicated, a weighted coordination requirements matrix $\mathbf{C_R}$ is generated by not dichotomizing the result of $\mathbf{A} \times \mathbf{D} \times \mathbf{A}$ matrix. Formally the individual weighted congruence, WIC for ego i can be expressed as:

$$WIC_i = \frac{\sum (\mathbf{C_R}[i,] \times d(\mathbf{C_A}[i,])) + \sum (\mathbf{C_R}[,i] \times d(\mathbf{C_A}[,i]))}{\sum d(\mathbf{C_R}[i,]) + \sum d(\mathbf{C_R}[,i])} \quad (2)$$

Where $d(\mathbf{x})$ represents dichotomizing matrix $\mathbf{x}$ to 0,1 such that all cells > 0 are set to 1, the multiplications are element wise multiplications, and we assume zero along the diagonals for both $\mathbf{C_R}$ and $\mathbf{C_A}$.

There are some clear similarities between the two metrics, in particular, the denominator, which represents the number of instances of required communication, is the same in UIC and WIC. The difference between these two metrics is subtle, but very important. UIC is functionally a subset of the original STC metric where examination is done only for a single person and produces a value between 0 and 1, inclusive. WIC results in higher values if communication was observed between ego i and those individuals with whom more coordination requirements exist. In this way, WIC captures not only whether or not communication was present, but also whether or not the communication addresses coordination needs.

3. PRELIMINARY RESULTS

For this work, we examined the interactions of a large Open Source software community, the GNOME desktop environment. Although there are number of commercial firms participating in the community, most project teams are highly distributed, having no more than a handful of developers present at each location. The community has established norms that prefer communication to go across mailing lists – however, not to the degree of the Apache project which requires all decisions to be made of the mailing lists. For more background, we direct the reader to German's description of this community[4].

We collected data from 10 different projects in the community. Projects were selected based on their age, number of developers contributing code, and overall community size. All projects were between five and ten years old, had at least 25 developers, and had archived mailing lists and a publicly accessible bug tracker. The projects varied widely in their goals – an email client, web browser, music player, and several system libraries were included in the sample.

For the purposes of generating data, we broke each of the projects into six month time periods that correspond to the release cycle of the community. Tasks within each project were mapped to source code files, filtering out automatically generated files, documentation, graphics, and other files that are not generally considered to be code. The task assignment matrix, $\mathbf{A}$, was generated through an examination of version control system logs for each project at each time period. An edge was present between a developer and task if the developer had modified that file at a time prior to that time period. The task dependency matrix $\mathbf{D}$ was calculated by looking at files that were committed together at any time prior to the time period – filtering out bulk operations affecting more than 20 files[1].

[1]It was found that operations on large numbers of files were almost always maintenance tasks, such as updating copyrights, or moving files around, and had little to do with technical activity in the project.

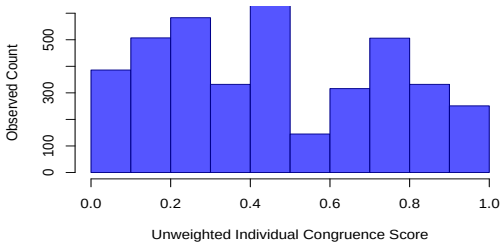

Figure 1: Histogram of Average Unweighted Individual Congruence (UIC) by Bug Report

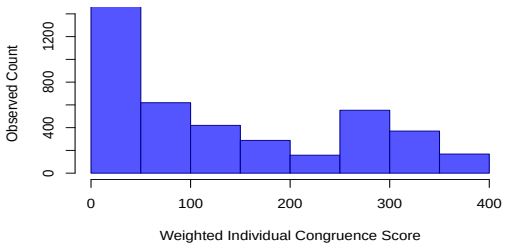

Figure 2: Histogram of Average Weighted Individual Congruence (WIC) by Bug Report

The actual communication network, $\mathbf{C_A}$ was calculated via an examination of archived bug reports and mailing lists. An edge was placed in $\mathbf{C_A}$ if two developers had directly communicated through an archived email message or by posting comments in the bug tracking system. This network was treated as a dichotomous network. Such a method of building networks misses large scale face to face planning events that occur at project conferences. However, the impact of long range planning and discussion on fixing individual bugs later down the line is likely small. These networks also miss interactions which may take place over non-logged chat, while such data is very valuable, the norms of the community request that communication be preserved through archived communication, such as mailing lists.

After calculating UIC and WIC for each project at each time period, We note that there is high correlation between UIC and WIC, approximately 0.9, which is to be expected, given their very similar derivations. We also compared these values to the number of instances of required communication, $ReqComm$, for each developer as measured in the denominator from equation 1 and 2. These correlations were 0.37 and 0.47 respectively – indicating a loose relationship between high levels of coordination requirements and actual communication. The distribution of each of the metrics can be seen in figures 1–3.

The unit of inquiry in the model is bug reports that were

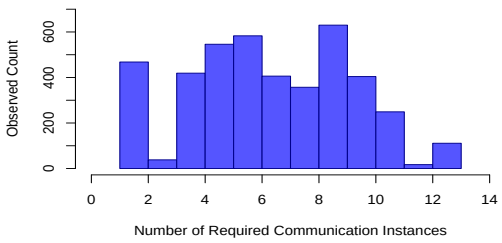

Figure 3: Histogram of Average Number of Communication Requirements, ($ReqComm$) by Bug Report

Table 1: Analyzing the Impact of Unweighted Individual Congruence (equation 3)

Variable	Estimate	Std Err	P
Intercept	1.484	0.145	<.001
$NumDevs$	1.301	0.088	<.001
UIC	-0.583	0.167	<.001

resolved as "Fixed" and had patches associated. The dependent variable in the model is the $\log_2$ of the time to resolve bugs as measured in days. To help control for complicated bugs, we also note the number of developers active on each bug. For each bug, we calculated the values of UIC and WIC for each developer active on the bug, and average these values by bug to determine the overall UIC and WIC for each bug. In most cases (80%) there was only one developer assigned to the bug, but frequently more developers participated in discussion. There were 5800 bugs for which we had complete data in the data set. The regression equations can be seen in equation 3 and 4 and the results can be seen in table 1 and 2

$$log_2(ResolutionTime) = \beta_0 + \beta_1 NumDevs + \beta_2 UIC + \epsilon \quad (3)$$

$$log_2(ResolutionTime) = \beta_0 + \beta_1 NumDevs + \beta_2 WIC + \epsilon \quad (4)$$

The results indicate that in both cases, the higher the individual congruence of the developers working on a bug, the lower the time to resolution. As expected, the more developers involved with the bug, the greater the time to resolve the bug. We find that UIC has a more pronounced effect, while WIC has much less of an effect at least by looking at the coefficient, but given the unbounded maximum of WIC, high weighted individual congruence may have a very profound effect on overall time to resolve defects. Both regressions explained only very a small amount of variance in the time to resolve bugs – 5.0% and 6.6% respectively for UIC and WIC.

Table 2: Analyzing the Impact of Weighted Individual Congruence (equation 4)

Variable	Estimate	Std Err	P
Intercept	1.542	0.127	<.001
$NumDevs$	1.382	0.088	<.001
WIC	-0.003	0.000	<.001

Table 3: Analyzing the Impact of Unweighted Individual Congruence and Communication Opportunities

Variable	Estimate	Std Err	P
Intercept	2.761	0.148	<.001
$NumDevs$	1.509	0.086	<.001
$ActualCommU$	0.061	0.024	0.009
$ReqComm$	-0.302	0.022	<.001

Table 4: Analyzing the Impact of Weighted Individual Congruence and Communication Opportunities

Variable	Estimate	Std Err	P
Intercept	2.700	0.147	<.001
$NumDevs$	1.501	0.086	<.001
$ActualCommW$	-0.001	0.000	0.403
$ReqComm$	-0.255	0.017	<.001

To better understand the phenomena, we broke UIC and WIC into their constituent parts, terming the numerator, which represents the actual communication $ActualCommU$ and $ActualCommW$ respectively. The process of dichotimization in equation 2 ensures that the denominator, $ReqComm$, is identical in both metrics. The results of the regression on these models is shown in tables 3 and 4.

Under the new regression models, we find that the amount of required communication is significant and that bugs that were worked on by individuals with higher levels of required communication show a lower time to resolution. We note that while $ReqComm$ is unbounded, in our data it has a maximum value of 13 – indicating a possible reduction of 93% and 90% in the time to resolve the bug in each model. We also note that $ActualCommU$ increases the time to resolve a bug, while $ActualCommW$ has no statistically significant relationship with the time to resolve bugs. The fit of the new models has increased significantly over the original model, with each model explaining 11% of the variance.

Finally, we explored the possibility of communication overload being a main culprit. In such a model, we include not only $ActualCommU$, but $ActualCommU^2$, as shown in table 5. If it is the case that a small amount of communication is beneficial, but excessive communication is harmful, then both $ActualCommU$ and $ActualCommU^2$ would be significant with negative and positive coefficients respectively. While the coeffecients point in the correct direction, $ActualCommU$ is far from significant. Thus, we cannot conclude that the effect of communication differs based on the volume of communication.

4. DISCUSSION

These preliminary findings ran counter to our own expectations, but appear to be robust with multiple different data

Table 5: Analyzing the Squared Effect of Actual Communication in UIC

Variable	Estimate	Std Err	P
Intercept	2.837	0.152	<.001
$NumDevs$	1.581	0.093	<.001
$ActualCommU$	-0.065	0.064	0.308
$ActualCommU^2$	0.014	0.007	0.033
$ReqComm$	-0.303	0.022	<.001

sets from within the community. In particular, we note that while our expectation was that higher individual congruence would lead to higher performance, and this was true, we found that when broken into its constituent parts the more instances of required communication, *ReqComm*, an individual had the faster their bugs would get resolved, while the more individuals they communicated with, *ActualCommU*, the slower their time to resolve bugs. Furthermore, we found no difference between the impact moderate and large amounts of communication on overall performance.

There should be more examination on what the meaning of a high score of *ReqComm* means for an individual developer and how it compares to more traditional social network metrics such as centrality, degree, and betweenness. In the current implementation of the algorithms, an individual can have a high value of *ReqComm* from a variety of different cause. For example, a user who modified only a single file that happened to be a core system library would have a high number of coordination instances. Likewise, an individual who modified a number of smaller files in pairs could also have a high value. These are two very different scenarios and should be examined as they may have dramatically implications.

From an algorithmic point of view, we make a particular note regarding the derivation of WIC: a dichotomized version of $\mathbf{C_A}$ was used, which discards any meaning regarding the frequency of the communication – a situation that is no different from UIC and the original STC metric. This is a shortcoming in the metric as it is likely that links that have higher coordination requirements, as shown by a high value for $\mathbf{C_R}[i,j]$, will need to have more frequent communication (i.e. a high value of $\mathbf{C_A}[i,j]$). We have intentionally chosen not to address this issue because of the wide variety of difficulties it brings up in knowing if a communication between two people was to coordinate some aspect of work or not. Further work on understanding the context of communications and identifying such relevant communications would be highly beneficial for this family of metrics.

Finally, the data for this work differs significantly from the original work examining STC in that this data examines teams working on different projects in a widely distributed Open Source community as opposed to a unified team working on a single project in a commercial firm. This community has a fairly rigid set of work norms that roughly follow the general pattern for most open source (see Fogel's work for more detail[3]) and include large amounts of extra communication on mailing lists and bug trackers while keeping commits small and rarely breaking the build. In addition, we note that because of the open nature of our data, some typical control variables were not readily accessible or meaningful – for example, in an open source community it can be very difficult to obtain survey data about the experience level of each developer and the lack of defined roles makes seniority variables impossible to calculate.

In such cases of massively distributed environments, as is found in Open Source, there is little doubt that norms of documentation and communication reduce the amount of "work" that a developer can complete. However, developer efforts in communication are key for keeping the project vital and attracting new talent. Indeed, in such distributed cases it is likely that those with the highest individualized congruence have lower productivity when measured in terms of observable metrics such as lines of code and bug resolution time, but may be more productive in terms of keeping the project alive and healthy. If this is the case, the we must examine the issue of whether or not using such metrics is prudent for measuring individual productivity and what implications that has for STC.

Acknowledgements

This work is funded by the National Science Foundation IGERT Training Program at Carnegie Mellon University (DGE-9972762) and the Office of Naval Research.

5. REFERENCES

[1] J. Anvik and G. Murphy. Determining implementation expertise from bug reports. In *Mining Software Repositories 2007*, Minneapolis, MN, USA, May 2007.

[2] M. Cataldo, P. Wagstrom, J. Herbsleb, and K. Carley. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *2006 Conference on Computer Supported Cooperative Work*, pages 353–362, Banff, Alberta, Canada, Nov. 2006. ACM Press.

[3] K. Fogel. *Producing Open Source Software*. O'Reilly & Associates, Sebastapol, CA, 2005.

[4] D. German. The gnome project: a case study of open source, global software development. *Software Process: Improvement and Practice*, 8:201–215, Sept. 2004.

[5] G. Valetto, M. Helander, K. Ehrlich, S. Chulani, M. Wegman, and C. Williams. Using software repositories to investigate socio-technical congruence in development projects. In *Mining Software Repositories 2007*, Minneapolis, MN, USA, May 2007.