Using Software Repositories to Investigate Socio-technical Congruence in Development Projects

Giuseppe Valetto, Mary Helander, Kate Ehrlich, Sunita Chulani, Mark Wegman, Clay Williams
IBM T.J. Watson Research Center
{gvaletto, helandm, katee, sunita_chulani ,wegman, clayw}@us.ibm.com

Abstract

We propose a quantitative measure of socio-technical congruence as an indicator of the performance of an organization in carrying out a software development project. We show how the information necessary to implement that measure can be mined from commonly used software repositories, and we describe how socio-technical congruence can be computed based on that information.

1. Introduction

One of the reasons why software development is inherently complex is because it is a *socio-technical* endeavor. Any non-trivial software development occurs within an intensively collaborative process, in which technical prowess must go hand in hand with the efficient coordination and management of a large number of social, inter-personal interactions across the development organization. Furthermore, those social and technical dimensions are not orthogonal. It has been recognized that the structure of a software product and the layout of the development organization working on that product correlate (*Conway's law* [1]); Parnas [2] has also observed that the subdivision of development responsibility tends to induce the modularization of a software product at least as strongly as its functional decomposition.

Therefore, measuring and understanding how people are organized and interact with one another whey they develop software can be important to improve productivity and quality. Given the mutual influence of social and technical aspects in software development, those social studies must be contextualized with respect to the technical work being done.

One way to investigate those issues is to measure if there is a good fit (or **congruence**) between the coordination structure mandated by the technical work units (tasks) on the one hand, and the actual social organization (as expressed for example by the communication paths observed among its members) on the other hand. Some studies, such as [3] suggest that high degrees of such *socio-technical congruence* are beneficial for software development performance.

Whereas those works assume the observability of inter-personal interactions within a software development organization, and – more importantly - a task-centered view of the development process, the tools most commonly employed in today's practice are artifact-centered. Therefore, we introduce a measure of socio-technical congruence that compares the structure of the software organization directly to that of the software product. Our choice can be intuitively motivated as follows: development of software artifacts that somehow depend on each other is likely to require coordination between the people responsible for those artifacts (their stakeholders). Conversely, lack of coordination across stakeholders of dependent artifacts may be a telltale of development problems. That intuition is also backed by quantitative studies, such as [4], which have found correlation between artifact dependencies and communication frequency among stakeholders involved with those artifacts.

Our measure of socio-technical congruence is derivable from data kept in software repositories commonly supporting the development process. This paper discusses how it can be defined and quantitatively measured on the basis of that data.

2. Socio-technical software networks

Software engineering researchers have recently turned their attention to the issue of social networking, since it enables the study of the actual organizational structure of software projects (for an overview, see Xu et al. [10]). Social networks models, combined with views of the software product [7] [8], to form *socio-technical software networks*, upon which various analyses can be carried out. To compute congruence, we use such a network, which combines three types of information (see Figure 1):

- communication/collaboration interactions between stakeholders which are mapped as undirected links between nodes laying on plane P, for *People*;

Fourth International Workshop on Mining Software Repositories (MSR'07)
0-7695-2950-X/07 $20.00 © 2007 IEEE

- inter-relationships between artifacts, which are mapped as directed arcs between nodes laying on plane S, for *Software*;
- "work relationships", which are mapped as directed arcs from plane P to plane S, and account for the work done by stakeholders on artifacts during the course of the project.

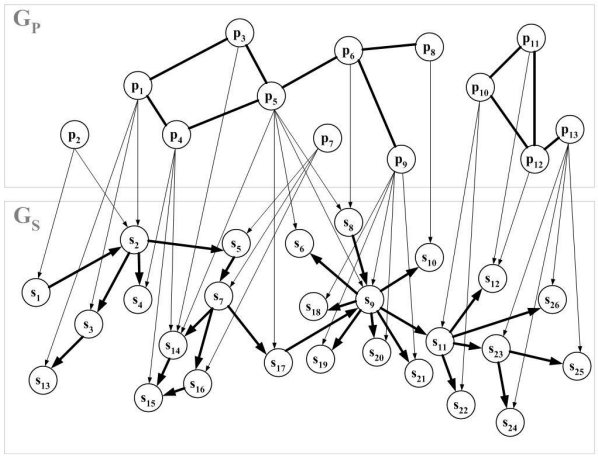

Figure 1: A socio-technical software network.

It is possible to construct a network like the one in Figure 1 by using the data maintained in various software repositories. Several works, such as CVSAnalY [6] or Augur [9], have shown how the nodes in plan P and S, plus the arcs connecting those planes, can be readily mined from arguably the most commonly used software repositories, that is, source control and management systems, such as ClearCase, CVS, Subversion, and others. The metadata associated to each commit record, such as the ID for the stakeholder committing the change, the modified file(s), the timestamp, etc. can be employed to that end.

The arcs in plan S can also be derived from the source management system, by extracting and post-processing coherent software configurations, as shown for example by Kenyon [11]. By running static analysis upon the source code files, one can draw the dependencies between the corresponding compilation units (such as function invocation, field access, inheritance, containment, etc.).

To represent the links in plane P, which signify collaboration between stakeholders, other repositories, such as developers' mailing lists and discussion groups, can provide reliable traces of collaboration, provided a method to resolve identities across them [15]. With an appropriate tool, e.g. [5], communications could even be monitored and integrated into the network as they happen. In case other data sources are not available, an approximation can still be extracted from source management systems. A basic approach is drawing collaboration arcs between all stakeholders that have ever worked on the same artifacts. For better accuracy, one could draw an arc only if stakeholders have worked on a common artifact within some time window[1].

We also propose to enrich graph components with attributes, such as arc *weights* (e.g. the number of changes to the same artifact by the same stakeholder, or the number of dependencies of a source file onto another), roles played by stakeholders in relation to artifacts, *confidence* (specifically for person-to-person arcs, representing the likelihood that two persons have in fact collaborated[2]), *size of contribution*, *timestamp*, etc. With that data at our disposal, we can obtain a rich and reliable representation of the network. On its basis we can compute a measure of socio-technical congruence with the technique detailed in Section 3.

3. Measuring socio-technical congruence

In this section, we mathematically formalize socio-technical congruence based on the topology of the network described in Section 2.

Let $G_P=(P,E_P)$ denote the digraph of people and their relationships, where the node set is P and the edge set is E_P. Edges in E_P correspond to relationships between pairs of developers in set P. For example, (i,j) in E_P may mean that persons i and j have communicated where $i,j \in P$. We use undirected edges, because we assume that communication relationships between people are always reciprocal.

Let $G_S=(S,A_S)$ denote the digraph representing software artifacts as nodes (set S) and their relationships (denoted by arc set A_S), and let J denote a set of "joins" – i.e., arcs connecting developers in P to artifacts in S. In interpreting set A_S, we assume arc direction implies dependency, for example invocation of one artifact by another. Similarly, directions implied by arcs in J imply an affiliation or involvement. For example, in Figure 1, (p_7,s_5) could indicate that person p_7 has an assigned role, such as owner, to artifact s_5, and (p_{10},s_{11}) could indicate that person p_{10} has a history of committing changes to artifact s_{11}.

To measure congruence, we first consider a proportion of relationships between software artifacts (e.g., arcs in A_S) that are mirrored in E_P, where detection of mirroring is facilitated by the join set J. That is, if (i,l) is in A_s and (k,i) and (h,l) are in J, then the arc (i,l) is "arc mirrored" if these two conditions are

[1] The value for the time window can be pre-set or can be computed separately for each single project, for instance based on frequency and distribution of multiple-users commits.

[2] The likelihood of collaboration by two stakeholders on a given artifact could for example "decay" by considering the time elapsed between commits on the artifact by the stakeholders.

true: (a) k and h are distinct nodes in P, and (b) either arcs (k,h) or (h,k) (or both) are in E_P. The number of arc actually mirrored divided by the number of times arcs in A_S could be mirrored in E_P provides a proportion that reflects one aspect of congruence, as is illustrated in Figure 2a. Figure 2b illustrates a second aspect of congruence we refer to as a "node tie", that is, when two distinct developers (here, h and k) both have some relationship to the same artifact (i.e. node $i \in S$), then we expect to see a relationship between h and k.

Congruence based on arc mirroring between graphs $G_P=(P,E_P)$ and $G_S=(S,A_S)$ with join set J is given by:

$$C(G_P, G_S, J) = \kappa / \gamma$$

defined when $\gamma>0$, where κ and γ are computed by the following algorithm for detecting and counting arc mirror patterns:

step 1: set κ, γ = 0
step 2: For each arc (i,l) in A_S:
  step 2a: Let J_i=all arcs in J incident on node i; let J_j=all arcs in J incident on node j.
  step 2b: For each (a,i) in J_i and (b,l) in J_j:
    If a≠b, then $\gamma=\gamma+1$.
    If(a,b) $\in E_P$, then $\kappa=\kappa+1$.

A similar computation finds socio-technical congruence based on node ties. As is the convention for measurements of congruence (see [12]), the quantification results in a value between zero and one, where a value closer to one indicates high congruence and a value closer to zero indicates low congruence.

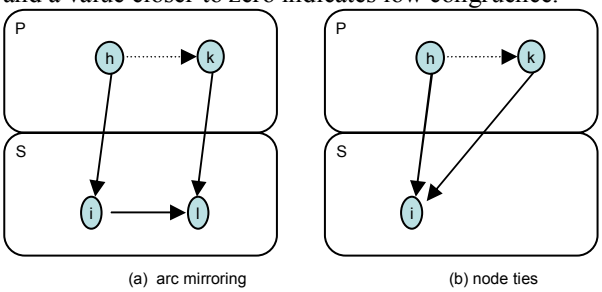

Figure 2: Congruence measurement concepts.

As an example, congruence measured based on *arc mirroring* for Figure 1 is 20%. There are six mirrorings in A_S: (s_2,s_4) by (p_1,p_4); (s_8, s_9) by (p_5,p_6), (s_8,s_9) by (p_6,p_9); (s_{11},s_{12}) by (p_{10},p_{11}), (s_{11},s_{12}) by (p_{10},p_{12}); and (s_{14},s_{15}) by (p_4,p_5). There are 24 cases where a mirroring is missing – for example, for arc (s_2,s_5), persons p_1 and p_7 are related to artifacts s_2 and s_5 respectively, but there is no arc between p_1 and p_7 (or persons p_2 and p_7) in E_P. The congruence measurement based on *node ties* is instead 57%. That is, software artifacts s_8, s_{12} and s_{14} each have developers who are related to each other; and artifacts s_2, s_9 and s_{14} each have at least one pair of developers who are not.

Note that both definitions of socio-technical congruence allow for global (i.e., over the whole network), as well as local measurement (i.e., over a region of interest, by restricting consideration to any subset of A_S or S). Note also that the two congruence measurements may be considered separately or combined, e.g. as a weighted combination.

Several extensions are also possible, for example by considering additional information provided by attributes, such as weights or confidence levels. The measurements are also easily adaptable to consider super nodes (i.e. aggregated nodes that represent a number of nodes), to enable working at different levels of granularity in the software project, e.g. compilation unit, module, subsystem, and the organization, e.g. individual, pair, team, department. Such adaptability enables scaling the analysis to very large projects.

4. Applications and Future Work

Our concept of socio-technical congruence describes the degree of alignment between social relationships and software relationships. Therefore a *global* congruence measurement provides a quick index of how well the organization is actually aligned with the planned sub-division of responsibility in the project. This measure can be made available to project leads, so that they can better *govern* the software development process and organization. *Local* values of congruence can also provide more insight: for example, they can be used to detect certain process tasks or system areas in which the collaborative effort is likely to be struggling, and implement remedies, such as, modify responsibilities or the organization layout to facilitate communication flow in the affected areas. Examining the *evolution* of the congruence value over time is also useful, for example for auditing purposes.

To validate the significance of socio-technical congruence, it is however necessary to correlate it to other software properties, and see whether it can be used as a predictor for them. Previous works have observed how hurdles to communication, like those often suffered by distributed teams, seem generally conductive to inefficiencies in the development of inter-dependent software artifacts [13][14]. We are currently working with quality metrics, such as defect density or frequency of modification requests (MRs), which can also be mined from commonly used software repositories, such as ClearQuest, Bugzilla and other MR tracking databases. If socio-technical

0-7695-2950-X/07 $20.00 © 2007 IEEE

congruence is an indicator of performance of inter-dependent development tasks, we expect to see - for instance - an increased number of defects where congruence is low.

Investigation on socio-technical congruence can progress in many other directions. We have initially focused on source code artifacts, mainly because the mining of source management systems is well understood and can provide high-quality, fine-grained information for our purposes. That, however, limits the scope of congruence as a metric to the implementation phase of a project. Other relationships (such as traceability, i.e. how artifact *derive from* – as opposed to *depend on* - others) could be used to expand socio-technical networks to span the whole software life cycle. That would enable using socio-technical congruence as an earlier indicator of performance. The scarcity of tools in the current practice that track and record those relationships, however, makes mining the necessary information reliably at all phases of the development process particularly difficult.

Finally, we are also interested in understanding the best way to incorporate socio-technical networks and congruence into development and management tools. This will empower concerned stakeholders to make more informed project decisions.

5. References

[1] M. E. Conway, "How Do Committees invent?" Datamation, 14(4):28-31, April 1968.

[2] D.L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, 15(12):1053-1058, December 1972.

[3] M. Cataldo, P.A. Wangstrom, J.D. Herbsleb, and K.M. Carley, "Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools", in Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'06), Banff, Alberta, Canada, November 4-8, 2006.

[4] M. E. Sosa, S.D. Eppinger, M. Pich, D.G. McKendrick, S.K. Stout, "Factors that influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry," IEEE Transactions on Engineering Management, 49(1):45-58, February2002.

[5] S.B. Fonseca, C.R.B. de Souza, and D.F. Redmiles, "Exploring the Relationship between Dependencies and Coordination to Support Global Software Development Projects", in Proceedings of the International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, October 16-19, 2006.

[6] L. Lopez, J.M. Gonzalez-Barahona, and G. Robles, "Applying Social Network Analysis to the Information in CVS Repositories", in Proceedings of the 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, May 25, 2004.

[7] C. Amrit, J. Hillegersberg, and K. Kumar, "A Social Network Perspective of Conway's Law", in Proceedings of the CSCW Workshop on Social Networks, Chicago, IL, USA, November 2004.

[8] C. de Souza, P. Dourish, D. Redmiles, S. Quirk, and E. Trainer, "From Technical Dependencies to Social Dependencies", in Proceedings of the CSCW Workshop on Social Networks, Chicago, IL, USA, November 2004.

[9] J. Froehlich and P. Dourish "Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams", in Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, May 23-28, 2004

[10] J. Xu, S. Christley, and Greg Madey, "Application of Social Network Analysis to the Study of Open Source Software", in *The Economics of Open Source Software Development*, J. Bitzer and P.J.H. Schröder eds., Elsevier Press, 2006.

[11] J.E. Bevan, J. Whitehead, S. Kim, and M. Godfrey, "Facilitating software evolution research with Kenyon", in Proceedings of the European Software Engineering Conference/International Symposium on Foundations of Software Engineering (ESEC/FSE), Lisbon, Portugal, 2005.

[12] Z. Liu and R. Laganiere "Phase congruence measurement for image similarity assessment", Pattern Recognition Letters, 28 (1), 166-172, January 2007.

[13] J.D. Herbsleb, and R.E. Grinter, "Splitting the organization and integrating the code: Conway's Law revisited", In Proceedings of the 21st International Conference on Software Engineering (ICSE 1999), Los Angeles, CA, May 16-22, 1999.

[14] J.D. Herbsleb, and A. Mockus, "An Empirical Study of Speed and Communication in Globally-Distributed Software Development", IEEE Transactions on Software Engineering, 29(6):1-14, June 2003.

[15] G. Robles, J.M. Gonzalez-Barahona, "Developer Identification Methods for Integrated Data from Various Sources", in Proceedings of the 2nd International Workshop on Mining Software Repositories (MSR 2005), St. Louis, MI., May 17, 2005.