A Weighted Congruence Measure

Irwin Kwan, Adrian Schröter, and Daniela Damian
Software Engineering Global interAction Lab, Department of Computer Science
University of Victoria, Canada
{irwink,schadr,danielad}@cs.uvic.ca

Abstract

Socio-technical congruence is an intuitive way to compare required coordination effort within a software development project with the actual ongoing coordination. The current model of congruence is limited because it builds on top of some simplifying assumptions. These assumptions, such as placing equal importance of coordination needs, often fail to reflect the actual nature of a project. We propose a model that derives actual coordination needs from fine grained task interdependencies and task assignments. This enables us to compare those needs with the real ongoing coordination other than just dichotomized measurements.

1. Introduction

Building good software requires coordination among developers and within teams. People who work on related software components should also coordinate their efforts with each other. Socio-technical congruence is a measurement that computes the alignment between the social coordination and with the technical dependencies as imposed by the project tasks [1]. The result is a ratio of how well the social connections among people match the technical dependencies among people.

The measurement of socio-technical congruence allows observers to study project with a "fine-grained" perspective of dependent tasks. A manager can use this measurement to view the alignment of his teams' coordination with the technical dependencies. Unfortunately, the measure of congruence presented by Cataldo, et al [1] uses dichotomized values for its relationships between people, resulting in a coarse interaction model.

We propose an improvement to the original congruence calculation. The improvement enables us to model the connections more fine grained by applying weights to each relationship. Weights provides a measure of strength between people and tasks, interdependent tasks, and between people. Thus, we can reflect organizational social and technical interactions more precisely. Hence, our resulting congruence measurement is a more precise measure of congruence than a measurement that uses dichotomized relationship strengths.

This refinement is necessary because the current way of modelling congruence is based on some implicit assumptions. Two of those assumptions are that every dependency is equally important, and that any coordination is good enough to satisfy a coordination need. Stepping away from those assumptions is important in order to more accurately model coordination, and to apply congruence to a larger set of projects.

In the remainder of this work, we start with describing the related work (Section 2), followed by the description of our measurement (Section 3). After discussing the benefits of this new measurement in Section 4, we end with a short conclusion (Section 5).

2. Related Work

In this section we discuss related work and current drawbacks in the concept of socio-technical congruence.

2.1. State of the Art

Socio-technical congruence indicates the *fit* between an organization's coordination requirements and an organization's social interactions. The coordination requirements indicate who should talk to whom based on their task assignments and the task interdependencies. The idea can be informally described as, "If I work on a task, and my friend is working on a task that's dependent on my task, then I should be coordinating with my friend". An organization is considered *congruent* if the social interactions match the coordination requirements.

Limited evidence shows that higher congruence leads to faster completion of modification requests in a software development project [1]. Hence, an interest in congruence gaps arose to determine how to increase congruence. Previous research on congruence gaps specifically looked into

the effect of gaps on distributed development. Ehrlich et al. [2] found that the presence of gaps increased the number of code changes.

Others like Valetto et al. propose remedial actions when socio-technical congruence gaps are diagnosed [5]. Examples of actions include closing a gap by augmenting coordination, eliminating the gap by refactoring software, and eliminating the gap by rearranging schedules.

2.2. Limitations of Congruence Measures

The socio-technical congruence calculation as introduced by Cataldo et al. [1] bears some limitations. As mentioned, two assumptions in the current congruence model are that every dependency is equally important, and that any coordination is sufficient to satisfy a coordination need. We believe these assumptions are not always true.

An ongoing debate in the socio-technical congruence community is whether gaps are "bad", and to what degree a gap is detrimental. Although limited studies show that high congruence correlates with higher performance, Marczak, et al. [3] illustrates that a project was able to deliver requirements on time despite the existence of gaps. For instance, there may be communication structures in the organization that compensates for a lack of congruence. Ehrlich, et al. [2] suggests that brokers may be able to mitigate the effect of gaps; this is corroborated by a study by Marczak, et al. [3]. If this is in fact the case, then the connections from individuals to the broker may be considered more important than other connections. Unfortunately, the current representation of congruence does not give a *priority* of which gaps are most severe.

There are different suggestions on how to improve the notion of socio-technical congruence. Valetto et al. suggest that edge weights can be incorporated in his socio-technical software networks [6]. They suggest that number of changes made to the same artifact, or the number of dependencies of a source file onto another are possible candidates for edge weighing, but do not provide concrete formulas. Schröter et al. [4] suggests annotating gaps in a time interval with respective software quality measures, allowing one to determine how problematic a gap can be with respect to software quality.

3. A Weighted Congruence Measurement

In this section we introduce our measurement to weigh dependencies between tasks and between people and tasks. This, combined with a weighted actual coordination, will result in a model that more accurately describes coordination.

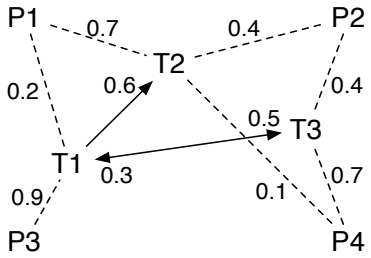


Figure 1. An assignment/task-dependency network. 70% of person P1's time is assigned to task T2, whereas T1 depends 60% on task T2.

3.1. Weighted Coordination Requirements

Our measurement applies weighted edges between 0 and 1 to the task assignment, task-dependency, and actual coordination matrices. Figure 1 shows an example of weights assigned to edges in a task assignment network. These weights are incorporated into the matrices required to calculate coordination requirements.

Weighted Task Assignment matrix is a $m \times n$ matrix where m is the number of selected people and n denotes the number of selected tasks. Each entry in the matrix defines the strength of the connection between a person i and a task j.

The weighted task assignment matrix may show how many hours one person is supposed to spend on a task. To break those values down to be between 0 and 1, they should be seen in the relation to the maximum work hours that person can spend.

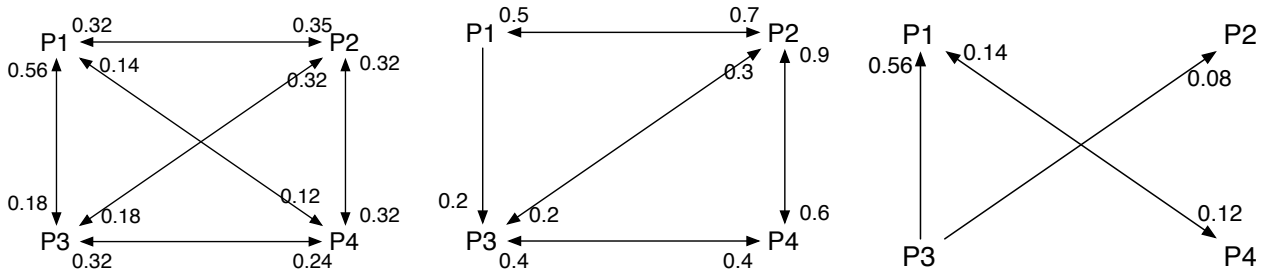
One can also measure the expertise a person has about a task. This would yield the amount of knowledge a person can share about a task.

Weighted Task Dependency matrix is a $n \times n$ matrix where n denotes the number of selected tasks. Each entry in the matrix describes the strength of the relation between two tasks.

The weighted task dependency matrix may show the ratio of dependencies a task has with another over the number of total dependencies that task has on others. For example, a task strongly coupled with another has a high task dependency.

To calculate the coordination requirements matrix, we follow the calculation proposed by Cataldo, et al. [1]:

$$\text{Task Assignment} \times \text{Task Dependency} \times (\text{Task Assignment})^T$$



(a) Coordination requirement derived from the given task assignment and task dependencies.

(b) The coordination that happened.

(c) The coordination that is lacking.

Figure 2. Compare coordination requirements with actual coordination to find lacking coordination.

This calculation will yield the coordination requirement matrix, which is a $m \times m$ people by people matrix (Figure 2(a)). The coordination requirement matrix tells you how strong the relations between people are supposed to be.

3.2. Weighted Congruence

To derive the final congruence coefficient, the coordination requirements matrix needs to be compared to the actual observed coordination. The comparison of the coordination requirements with the actual coordination follows three steps.

1. Create a people by people coordination matrix using our method (Figure 2(a)).
2. Subtract the actual coordination matrix (Figure 2(b)) from the coordination requirement matrix. This step differs from existing methods (e.g. [1, 6]) that compute congruence with a ratio.
3. Set values that are less than zero to zero (Figure 2(c)).

The resulting matrix illustrates areas that lack sufficient coordination to satisfy the coordination requirements. This matrix is a *lack-of-coordination matrix* (Figure 2(c)) that identifies each gap along with its weight. The larger the value, the more severe the lack of coordination is.

To convert this lack-of-coordination matrix to a single socio-technical congruence measurement, sum the edge values and divide them by the sum of the edge values in the coordination requirements matrix. Then, subtract the result from 1 to get the congruence measurement.

3.3. Weighing Actual Coordination

Different actual coordination measurements may be used for the congruence calculation. We present a few examples of techniques that can be used to weigh actual coordination matrices:

Communication A social network such as a communication network can be weighed according to the amount of ongoing communication. Lets consider Alice and Bob that work on depended tasks:

1. If Alice talks to Bob about his interdependent tasks, then for each task discussed the weight is increased by $a \cdot 1/n$, where a is the weight of the communication and n is the number of Alice's tasks that depend on Bob's tasks.
2. In case Alice discusses a task dependency very frequently with Bob, then, given a three point scale "occasionally/frequently/very frequently", a is set to 1. For another task that is only occasionally discussed, a would be $0.\bar{3}$.

In the end, if only two of Alice's tasks depend on Bob's tasks, the weight of Alice actual coordinating with Bob would be $1.0 \cdot 1/2 + 0.\bar{3} \cdot 1/2 = 0.6\bar{6}$.

Distance We can also measure the relationship between Alice and Bob by looking at different types of distance. We can measure

- how far they are apart in terms of physical distance, like meters or kilometres,
- the time difference between their time zones, and
- the number of organizational units that part them.

These values are normalized to fit within 0 and 1. All three of those measures result in a symmetric matrix, meaning Alice will always have the same distance to Bob and vice versa.

After we weigh all entries within the task assignment, task dependency, and actual coordination matrix, we can compare the coordination requirements with the actual coordination using the method described in Section 3.1.

3.4. Illustration of Weighted Congruence

In Figure 1 we see two overlaid networks. One connecting tasks with each other, the other connecting people with tasks. In the task dependency network shows for example that task T_1 depends on task T_2 and the strength is denoted with 0.6. Whereas the allocation network in case of P_1 and T_1 that person P_1 is allocated to work on task T_1 with dependency of strength 0.2. This dependency strength can for instance denote the percentage of time the person is supposed to dedicate to that task per day.

Figure 2(a) depicts the coordination requirements between people. Looking at P_1 and P_3 shows that P_3 needs to coordinate quite a lot with P_1 but not the other way around. After subtracting the actual coordination (Figure 2(b)) from the coordination requirements yields a sparse network illustrated in Figure 2(c). Only in four cases is coordination lacking. The sum of the edge weights in Figure 2(a) is 3.37. Thus, the resulting congruence for this example is $1-[(0.56+0.14+0.08+0.12)/3.37] = 0.74$, whereas the traditional coefficient would have yielded a value of 0.9, since there is only one coordination requirement not fulfilled (between P_1 and P_3).

4. The Benefit of Weighted Congruence

Using a weighted congruence model allows us to deal with situations that are not handled with regular congruence. We can investigate relationships between people and the tasks in finer-grained detail, including situations where people are partially allocated to tasks. The measure can also provide a ranking of missing coordination: It suggests which lack-of-coordination gaps are most important to fill. Using the weighted congruence measure allows a person diagnosing the organization to utilize *locality* and *priority*.

Locality allows us to identify which area of a network has an issue to resolve. Not only does this give us an overall measure of congruence, but now we can identify potential coordination problems in the network. Although the original model does give us some limited locality in the sense of identifying congruence gaps, it is far more coarse grained. For instance, using our weighted model, we can identify people working on two interdependent tasks who do not discuss both tasks.

Priority ranks people pairs where more coordination might be needed by the computed lack of congruence. Ranking is a considerable improvement over Cataldo's model since it allows us to only extract an unordered list of gaps. Because filling congruence gaps can be risky [5, 2], a ranking can provides us with a better assessment of which gaps to fill first if at all.

By allowing better locality and priority, we believe that our weighted model of congruence provides a better way to study congruence and to diagnose issues.

5. Conclusions and Future Work

We extend the socio-technical congruence measurement by proposing a model that involves weighted edges. By weighing each edge with a value between zero and one, we provide a method to weigh both gaps to decide which is more critical, as well as ongoing coordination that should be intensified. This concept builds can identify properties of organization, such as locality to closer investigate the coordination network, and priority to see critical areas for improvement.

We are currently exploring additional schemes for weighting edges, and intend to conduct studies to determine their applications. Our future plans include determining techniques that allow us to weigh the relationships between people and tasks, tasks and other tasks, and people and people. We then intend to empirically investigate the weighing techniques that we suggest by applying them to various case studies. By weighing people and tasks relationship, task interdependencies, and people interrelationships, we hope to more easily diagnose and resolve coordination issues.

References

[1] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *CSCW '06: Computer-supported Cooperative Work, November 4–8*, 2006.

[2] K. Ehrlich, M. Helander, G. Valetto, S. Davies, and C. Williams. An analysis of congruence gaps and their effect on distributed software development. In *MSR'07: Mining Software Repositories, May 20–26*, 2007.

[3] S. Marczak, D. Damian, U. Stege, and A. Schröter. Information brokers in requirement-dependent social networks. In *16th Intl Requirements Engineering Conference (RE08), Barcelona, Spain, Sept. 8–12*, 2008.

[4] A. Schröter, I. Kwan, L. D. Panjer, and D. Damian. Chat to succeed. In *RSSE '08: Proceedings of the 2008 international workshop on Recommendation systems for software engineering*, pages 43–44, New York, NY, USA, 2008. ACM.

[5] G. Valetto, S. Chulani, and C. Williams. Balancing the value and risk of socio-technical congruence. In *Socio-Technical Congruence Workshop (in conj. ICSE 2008)*, May 2008.

[6] 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 *MSR'07: Mining Software Repositories, May 20–26*, 2007.