Assessing Team Performance from a Socio-technical Congruence Perspective

Li Jiang
School of Computer Science, University of Adelaide, Australia
li.jiang@adelaide.edu.au

Kathleen M. Carley
Institute of Software Research Carnegie Mellon University, USA
kathleen.carley@cs.cmu.edu

Armin Eberlein
Dept. of Computer Science & Engineering American University of Sharjah, UAE
eberlein@ucalgary.ca

***Abstract*—There are many factors that provide input into the software development process, such as the values, beliefs, norms, practices, skills, behaviors, knowledge and goals of stakeholders. Research has shown that successful software system development relies on alignment or congruence between these factors. How to monitor the level of congruence between these factors and how to use the congruence as an indicator or a measure to monitor a software development process is a challenge in software engineering. This paper proposes a model that uses three congruence measures to examine the levels of social-technical congruence in software development processes. Using a controlled experiment with seven student teams developing a robot project, this paper demonstrates that the proposed congruence measures provide results consistent with the assessment by the course lecturers.**

***Keywords*-social-technical congruence; software engineering process; team performance**

I. INTRODUCTION

Software development has become an increasingly complex process [1]. Most often, software development involves teamwork and is embedded in a complex socio-technical context [2]. The significance of socio-technical factors, such as the values, beliefs, norms, practices, skills, behaviors, knowledge and goals of stakeholders, has been recognized as an important issue [3, 4]. Research has shown that successful software-system development relies on alignment or congruence between these factors [5-7]. However, how to measure the level of congruence between these factors is a challenge. In the last several years, there has been growing interest in addressing the impact of socio-technical congruence on software development [3, 4, 8]. A set of concepts, measures and methods has been developed to address social-technical congruence in the areas of task dependency and communication [9]. Typically, it appears that, firstly, there are many dimensions to socio-technical coordination and/or congruence, such as designer skill, knowledge, values and ability to learn, which also have a major impact on the product design and development process [10]. Thus, the research in the area of socio-technical congruence needs to address the issues of congruence requirements according to the needs of people sharing or improving knowledge. Secondly, socio-technical congruence tends to change during the software development process. It appears that the congruence level is lower at the beginning of the project and then gradually increases. However, our observation shows that the highest level of congruence occurs about half way through project development and lasts until the later parts of the project. This is because team-building and establishing harmonious working relationships in a team takes time. This is especially true when the project requires a lot of skills and knowledge that not all team members possess. Building on the task-dependent congruence measure proposed by Cataldo*, et al.*[9], this paper develops two additional congruence measures: Knowledge-dependent congruence and resource-dependent congruence measures. Together, these three congruence measures are used in a three-dimensional congruence measurement model that was applied to assess the socio-technical congruence of student teams that were involved in a semester-long software project.

We measure the level of congruence between the actual team communication and the communication required by the task and skill dependencies between the team members involved. We assume that the level of congruence of a student team reflects the actual quality of the software development process and is consistent with the performance of the student team. To examine this hypothesis, a case study is conducted with seven student teams. In the case study, the evolution of these measures in a software development team over time was also studied.

The rest of the paper is organised as follows: Section 2 presents related work. The measures are proposed in Section 3. The dataset and methodology are provided in Section 4. The results, observations and the threats to validity are presented in Section 5. Conclusions and future research comprise Section 6.

II. RELATED WORK

The idea of socio-technical coordination and/or congruence of software development teams was proposed by Conway in [6]. In [9], Cataldo *et al.* reported on their work on calculating socio-technical coordination and/or congruence of software development teams. They define 'congruence' as the matching of the communication and task dependency within a software organisation. Most importantly, Cataldo *et al.*'s paper proposed coordination

requirements based on the task-dependent relationships of a project and examined the impact of congruence on task performance. Anita *et al.* [11] further refined the congruence concept as being a state in which an organisation has aligned its coordination capabilities to meet the coordination demands of the technical products under development. They also discussed the nature, dimension and broader implications of congruence. Valetto *et al.* [12] described a model to compute socio-technical congruence based on measures derived from social network approaches. There is also some research exploring the use of social network approaches to study the developers' social network. For example, Madey and Freeh used social network approaches to help analyse the links between developers in an open source project [13]. Building on social network and data-mining approaches, Wolf *et al.* explored the collaboration issues of software teams in the Jazz project [14].

Assessment of team performance is a difficult issue [15] that is still not well-understood due to the intricate complexity of the software development process and the many uncertainties involved [16, 17]. In software engineering, various models and standards have been developed, such as CMM [18] (or CMMI [19]), ISO 9001 [20], ISO 12207 [21]. These standards and models provide comprehensive assessment information for a software team. Much research has been done that shows that a team performs better if a team or organisation satisfies or passes the assessment criteria required by these standards and models [22, 23]. However, singular measures of team performance are usually not adequate and measures from multiple dimensions are needed [24]. Moreover, a software system is a social-technical system as it involves both technical and social aspects. This paper studies social-technical congruence using a controlled case study of seven student teams.

The major differences between the work presented in this paper and the work discussed above are: Firstly, This paper extends the concept of congruence by adding the two dimensions of knowledge congruence and resource congruence. Secondly, the original coordination requirements proposed by Cataldo *et al.* [9] are based on a task dependency matrix. However, we extended the coordination requirements by including knowledge- and resources-dependent coordination requirements based on the computational social organisation network structure proposed by Carley [4, 25],. We also examined the relationships between the three types of coordination requirements and demonstrated the differences of the congruence measures in the case of the seven student teams. Thirdly, we examined the evolution of socio-technical congruence in a software project and the correlation of these measures to the performance of the student teams.

III. A Three Dimensional Model to Measure Congruence

As discussed above, software development is a process embedded in a dynamic and complex socio-technical system. At the macro level, the system includes social and technical components. At the micro level, numerous factors are involved in each component. Technical components include, e.g., the processes, tasks, techniques, knowledge and tools used in the software project. Social components include, e.g., people and their attitudes and behaviour, as well as organisational norms, rules and culture. It would be ideal to consider socio-technical congruence both at the macro level and the micro level. However, at present, the major focus of socio-technical congruence research is on the congruence between task dependency and developer interactions, that is, examining to what extent developer interactions and communication fits the requirements from the perspective of task dependency. For simplification of discussion in the following sections, some notations will be introduced to describe socio-technical coordination requirements.

The relationships between the elements of social and technical components are represented as a set of networks. A network, $\boldsymbol{N}$, consists of two sets of nodes, U and V, and a set of edges, $E \subset U \times V$. The element $e_{i,j} \in E$ indicates a relationship or tie between the nodes $u_i \in U$ and $v_j \in V$. A meta-network refers to a set of networks with multiple types of entities, such as people, knowledge, skills, resources and locations. To implement measures and calculate congruence, a network is represented as an adjacency matrix. Given a network $\boldsymbol{N}=((U, V), E)$, the cardinality of U and V is represented as $|U|$ and $|V|$, respectively. Figure 1 shows four examples of such networks, with A_1 to A_3 representing agents 1 to 3, and T_1 to T_5 representing Tasks 1 to 5. An element $e_{i,j}$ =1 in the

Agent-Task Matrix (AT)

	T1	*T2*	*T3*	*T4*	*T5*
A1	0	1	0	0	1
A2	0	0	1	1	0
A3	1	1	0	0	0

(a)

Task-Task Matrix (TT)

	T1	*T2*	*T3*	*T4*	*T5*
T1	0	0	1	0	0
T2	0	0	0	1	1
T3	1	0	0	0	1
T4	0	1	1	0	0
T5	0	0	0	1	0

(b)

Task-Agent Matrix $(AT)^T$ (transpose of AT)

	A1	*A2*	*A3*
T1	0	0	1
T2	1	0	1
T3	0	1	0
T4	0	1	0
T5	1	0	0

(c)

Agent-Agent Matrix ($CR_{(T)}$)

	A1	*A2*	*A3*
A1	0	2	0
A2	2	0	2
A3	1	2	0

(d)

Figure 1 Examples of representation of the networks with matrices

Agent-Task Matrix (*AT*) (Figure 1(a)) represents the relationship of an agent *i* having been assigned to task *j*. Otherwise, $e_{i,j}$ =0 means that there is no such assignment relationship.

According to Cataldo *et al.* [9], the socio-technical coordination requirements (denoted as CR_T) are defined as:

$$CR_T = AT \times TT \times (AT)^T \quad (1)$$

where CR_T is an agent-to-agent communication network (represented by $[cr_{i,j}]_{m \times n}$) and each $cr_{i,j}$ represents to what extent the two agents (*i, j*) are required to communicate reciprocally, as imposed by the task dependencies. The subscript *T* indicates that the requirements are derived based on task dependency. Thus, the coordination requirements are called 'task-dependent coordination requirements' in this paper. Information about *AT, TT* can be found in Table 1. $(AT)^T$ is the transpose of *AT*. Those values can be binarised (e.g. 1 represents that there is a communication requirement between two agents, 0 represents no communication requirement) in the matrices if the focus is on whether or not there is a dependency relationship.

Based on (1), a task-dependent congruence measure was derived to calculate the socio-technical congruence as:

$$Congruence\ (CR_T, AA) = 1 - \frac{diff\ (CR_T, AA)}{|CR_T|} \quad (2)$$

where *AA* is the actual communication matrix generated from communication records or communication observation of the project team; $diff\ (CR_T, AA)$=cardinality {differences between CR_T and $AA \mid cr_{i,j}>0$ & $a_{i,j}>0$ } and $|CR_T|$ = cardinality $\{cr_{i,j}>0\}$. Each element $a_{i,j}$ in *AA* represents actual communication recorded between the two agents *i, j*. The values of $a_{i,j}$ can be binarised (e.g. 1 represents that there was recorded communication between the two agents *i, j*; 0 represents no communication between the two agents).

For the purposes of this paper, the measures defined in [3] were slightly modified as can be seen from equation (2) above. For example, the number of differences between the values $cr_{i,j}$ and $a_{i,j}$ (at the same position *i, j* of the compared matrix CR_T and *AA*) is 5, and $|CR_T|$=20, then, $Congruence(CR_T, AA)$ =0.75.

TABLE 1. NAME AND ABBREVIATION OF NETWORKS USED

Symbol	Nodes: U	Nodes: V	Name and implication
AA	Agent	Agent	Communication Network. Contains information about who talks with whom.
AK	Agent	Knowledge	Knowledge (Skill) Network. Contains information about who knows what.
AR	Agent	Resource	Capabilities Network. Contains information about who uses which resource.
AT	Agent	Task	Assignment Network. Contains information about who has been assigned to do what.
KT	Knowledge	Task	Knowledge Requirement Network. Contains information about what knowledge/skill is required to complete a specific task.
RT	Resource	Task	Resource Requirement Network. Contains information about what resources are required to complete a specific task.
TT	Task	Task	Precedence Network. Contains information about task dependencies.

As discussed above, the coordination requirements proposed by Cataldo *et al.* [9] were based on task dependency. However, the communication needs of software teams are multi-dimensional [26]. Communication between developers can be a result of the need to discuss defects in code or requirements or design, to share or solicit information from each other, to schedule meetings or share resources. It can also be driven by a developer's need to obtain knowledge or learn specific skills necessary to complete his/her tasks [27]. Communication and sharing of knowledge between developers is necessary for software teams [28]. For a mature software organisation, communication and sharing knowledge between developers is even more important, as the growing size and complexity of software projects demands various skills and knowledge. Thus, it is necessary to look at congruence from the following perspectives:

- The communication needs that arise from sharing knowledge/skills between developers and/or junior developers obtaining desired skills from seniors to complete assigned tasks.
- The communication needs that arise from management perspectives, such as sharing resources, scheduling meetings, etc.

Based on these two perspectives, the authors propose the following two additional congruence requirements to extend the task-dependent congruence model:

- Knowledge-dependent congruence: Knowledge-dependent congruence can be calculated by using the knowledge-dependent congruence requirements matrix (denoted as CR_K), which is defined in equation (3).
- Resource-dependent congruence: Resource-dependent congruence can be calculated by using the resource-dependent congruence requirements matrix (denoted as CR_R) given in equation (4).

$$\mathrm{CR}_K = AT \times (KT)^T \times (AK)^T \quad (3)$$

$$\mathrm{CR}_R = AT \times (RT)^T \times (AR)^T \quad (4)$$

where

- CR_K is an agent-to-agent communication network with each entry $cr_{i,j}$ representing the extent the two agents (*i, j*) are required to communicate, imposed by the knowledge required to complete the assigned tasks. The subscript *K* indicates the requirements are derived based on knowledge and task dependency.
- CR_R is an agent-to-agent communication network and each entry $cr_{i,j}$ represents the extent the two agents (*i, j*) are required to communicate, imposed

by the shared resources required to complete the assigned tasks. The subscript *R* indicates the requirements are derived based on resources and task dependency.

- Information about *AT, KT, AK, RT, AR* can be found in Table 1. $(AK)^T$, $(KT)^T$, $(RT)^T$, $(AR)^T$ are the transpose of the corresponding matrices. Similarly, we define the following congruence measures:

$$Congruence(CR_R, AA) = 1 - \frac{diff(CR_R, AA)}{|CR_R|} \quad (5)$$

$$Congruence(CR_K, AA) = 1 - \frac{diff(CR_K, AA)}{|CR_K|} \quad (6)$$

As discussed above, software development is a dynamic process which involves various activities and tasks using several resources during different stages of software development, regardless of the selected process model. Therefore, socio-technical congruence also likely evolves in this dynamic process. Taking temporal information into account, equations (1) to (6) are collectively reformulated as:

$$CR_T(t) = AT(t) \times TT(t) \times (AT(t))^T \quad (7)$$

$$CR_K(t) = AT(t) \times (KT(t))^T \times (AK(t))^T \quad (8)$$

$$CR_R(t) = AT(t) \times (RT(t))^T \times (AR(t))^T \quad (9)$$

$$Congruence\ (CR_T(t), AA(t)) = 1 - \frac{diff\ (CR_T(t), AA(t))}{|CR_T(t)|} \quad (10)$$

$$Congruence\ \ (CR_K(t), AA(t)) = 1 - \frac{diff\ (CR_K(t), AA(t))}{|CR_K(t)|} \quad (11)$$

$$Congruence\ \ (CR_R(t), AA(t)) = 1 - \frac{diff\ (CR_R(t), AA(t))}{|CR_R(t)|} \quad (12)$$

where *t* is a time parameter. It can represent a day, week, month or any other time unit that is adequate within the context. A week is used in this paper.

IV. Data and Methodology

This section briefly describes the background of the software project and the data collected on the teams involved in the project. The methods used to collect and analyse the data are also presented.

A. Dataset

The data was collected from students enrolled in the Software Engineering Group Project course in semester 2, 2010, at the University of Adelaide, Australia. The course is an important course for students to complete their degree. Three lecturers taught this course in that semester. Students were in the third year of a computer science major. The students were assigned to a team based on their marks of computer science courses in their second year of study to ensure a balance of capability and knowledge between each team. The students were asked to complete a robot mining project with about 50% of the requirements of the project being given. The students needed to talk to clients (lecturers) to gather more requirements to complete the project based on their understanding and analysis of the given requirements. The programming languages were Java and lejos which is a Java-based firmware used for programming on the LEGO Mindstorms NXT brick. Java was used for programming of the host side (a standalone PC) that contained GUI components (i.e., Map, Map Editing, Control Robot) and Communication components (establish links between host and robot, and control the robot). Students were required to follow a software process model selected by the team, such as Waterfall, XP, Scrum, RUP or combination of some of these models that are appropriate for the team. Students were also required to manage their own configuration and quality assurance process, and use software tools wherever feasible and helpful for improving team productivity and product quality. The students were required to spend 12 hours per week on the course during a semester-long period. The teams were required to meet with lecturers for 30 minutes each week to report their progress and obtain feedback and guidance. Several artefacts were required to be delivered, including a software project management document (SPMP); software requirements document (SRS) and software design document (SDD). The SPMP included detailed information about the task definition, the task plan (with Gantt Chart), and other management-related information. According to the course requirements, all artefacts were required to be put into *Subversion (*SVN), a revision (or version) control system, including code and weekly journals by each individual team member. The teams were composed of six students with mixed levels of academic achievement. For each team, the dataset includes:

1) SVN Repository that Includes the Following Components:

a) The SVN-Commits of all Members of the Team. Each commit contained information about the revision number, the ID of each team member, the time at which the commit was made, a message about each commit, the folder used for committing the file, the type of action taken regarding the committed file, the file name and its location, and files changed in each commit. For instance, more than 747 revisions were made from 9 August to 25 October 2010 by Team 1, a span of ten weeks (the first two weeks were used for team-building and understanding the requirements of the project and the tools). From these revisions, 2932 records were generated that show the changes in the files, including programs, data items and documents. An example of such information is shown in Table 2 (the last column and the generation of the task names will be discussed later in this paper).

In order to ensure that the collected data was relevant to the project, the dataset was first cleaned up to obtain the information relevant to the overall project. This included removing those commits or

TABLE 2. EXAMPLES OF RECORDS GENERATED FROM SVN REPOSITORY

Author ID	Revision number	Commit date	Message	Actions taken	Type of files	Resource name	Task name
s78	298	2010-09-12T02:07:23.627302Z	Working with *XXX* Fixed the GUI bug, modified the layout of window	M	file	/src/host/gui	GUIDevelopment1
s05	325	2010-09-13T14:31:40.951205Z	Edited pictures and updated contents on SDD_HumanInterfaceDesign.tex and SDD_ResourceEstimates.tex. However, pictures of GUI yet to be updated.	M	file	/docs/SDD	SDD1

Notes: For ethical reasons, the author IDs in the first column of the table are the codified IDs and the team member's name mentioned in the 'Message' column is replaced with *XXX*.

records that were: 1) purely related to a delete operation; or 2) related to individual work, such as committing journals or commits made to the branches that are not relevant to the project. Finally, 2265 (out of 2932) records were obtained that contained SVN-commits meaningful to the project work.

b) *Journals of each Team Member for each Week*. Submitting a journal was part of the course requirements stated at the beginning of the course. The journal contained information about what each team member had done, the hours they had spent, and with whom they had worked. An example of a team member's partial journal information is given in Table 5. The actual team members' communication/interaction information can be extracted from the information contained in the journals and the information left in the message column of the SVN-commits records created from *a)*.

2) Survey Data which Contains Information about the Knowledge/Skills of each Team Member. Ten key knowledge/skill assessment items were developed by the course lecturers based on the course objectives. A survey of all team members was conducted at the end of the semester to assess their knowledge and skills. The reason for assessing the team members' skill at the end of the semester is that each team member can give objective assessment based on their peers' actual capability and skills demonstrated in the project. Each member was asked to fill in the skill assessment form both for themselves and for the rest of their team members. An example of such survey results from a team member is given in Table 6, which contains both the skill list and the survey result. The final dataset related to the team members' knowledge/skills was generated based on the aggregation of the survey results of all team members.

3) Student Discussion Forums. A student discussion forum was created for each group where the group members in each group could communicate with each other. The communication information captured in these forums and the weekly journal provide data about the actual communication that occurred during the student projects.

In the research, the seven teams were randomly selected out of 20 teams at the beginning of the course, and the processes of these teams were observed and the data related to these teams were analysed.

B. Methodology

1) Creating the Network Representation of the Data Obtained. In this research, network approaches were intentionally used—that is, all dynamic behaviour of team members' weekly activities were represented with networks and their relationships. Thus, network measures and tools could be used to help analyse the team members' dynamic behaviour in the project. Based on the data records obtained, the following networks were developed:

a) *Agent-to-Task Network (AT).* To determine who committed what information, each change was mapped to a specific task based on the message generated by each commit, each student's journal, and the software project plan made in the team's SPMP. A list of tasks defined in SPMP was used. Two examples of such tasks are given in Table 2. The information contained in AT in each meta-network was different. An example of an AT network from Week 4 of the team project is shown in Table 7.

b) *Agent-to-Resource Network (AR).* The folders and files used in SVN, as illustrated in Table 1, were considered to be the immediate resources shared and used by the entire team. The AR network was developed from the information in columns 1 and 7 (see Table 2) across the entirety of the obtained records.

c) *Task-to-Task Network (TT).* This network was developed from the team's project plan in SPMP, where the detailed task list and task dependency were presented. For more detailed information, please refer to [29].

d) *Resource-to-Task Network (TR).* This network was developed from the task and resources columns in Table 2. Within the context of the paper, the resources refer to the files and directories of the SVN repository shared by the team.

e) *Agent-to-Knowledge Network (AK).* This network was developed based on the survey of team members' skill/knowledge, as discussed above. This

network contained information about who had the knowledge required by the project (see Table 6 for more information).

f) *Knowledge-to-Task Network (KT).* This network was developed by mapping each task defined in the team's SPMP to the actual knowledge/skill required to complete the task. Readers can refer to [29] for more information.

g) *Agent-to-Agent Network (AA).* This network was developed based on students' weekly journals (as shown in Table 5) and student forums, in which the information about each individual's work was stated. Moreover, the message of each SVN-commit also provided a mechanism to verify that communication occurred between the team members involved.

The collected data was organised on a weekly basis according to the time stamp on each commit in SVN. All seven networks described above were created for each team for each week and a meta-network was created by aggregating the seven networks for each team and for each week. Finally, a dynamic network was formed for each team by aggregating ten meta-networks (one meta-network per week) of each team, as discussed above. In this research, ORA [30], a dynamic network analysis tool, was used to process some parts of the data obtained, such as extracting data from *Excel* files, creating dynamic networks, and computing some measures. As an example, the dynamic network and the meta-network created with ORA for Team 1 are shown in Fig. 2.

2) Developing a Set of Metrics to Measure Team Performance. To examine the assumptions proposed in the first section, a subset of the software processes proposed in [21] were monitored and assessed for all seven teams selected in the research. A set of metrics were defined to assess the quality of these processes. The aggregation of the metrics forms the team performance measure for each week. The information about the assessment metrics, artifacts and engineering activities of the seven teams are illustrated in Table 8. The reason for selecting these metrics and processes are: a) it is feasible to collect the data related to these metrics during a semester-long project as all of the artifacts and processes were part of course requirements and/or assessment components; b) these metrics collectively present information about the performance of the teams and quality of the software processes used by the teams.

3) Studying the Relationships between each of the Three Congruence Measures and Team Performance. Adopting the measures defined above, and the measures from network science [31] and software process engineering [32], the following research questions were investigated:

a) How do the three congruence measures change over ten weeks?

b) Are these three congruence measures (dependent variables) correlated with the overall performance of each team in each week? (performance measures are independent variables and defined in Table 8)

c) Which measure provides reliable information that allows the assessment of the quality of the software process of the teams?

V. Results

A. Measuring Congruence and Team Performance

Based on the measures defined in (10) to (12), it was possible to compute the three socio-technical congruence measures of the teams over time. As an example, Figure 3 illustrates the three socio-technical congruence measures calculated for Team 1 over the 10 weeks. As can be seen from the figure, the values of congruence measures for the team changed over the ten weeks. The authors believe that these changes are reasonable; typically, when a new team starts a new project, team-building and learning various tools takes time, especially in the case of student teams. This might also be true for a mature team that starts a new project in which requirements are not entirely clear or new knowledge and skills are required. Further analysis has shown that the changes of knowledge-dependent congruence and resource-dependent congruence present, interestingly, a similar pattern. They both increase, generally, over the time, even though the increments are uneven. When examining and comparing the weekly

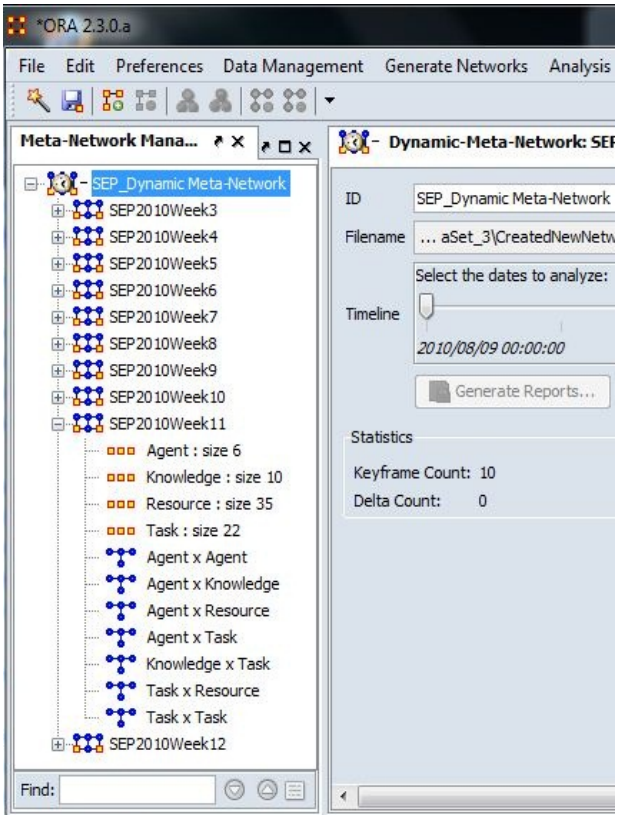

Figure 2. Parts of the dynamic network and an example of Week 11 meta-network

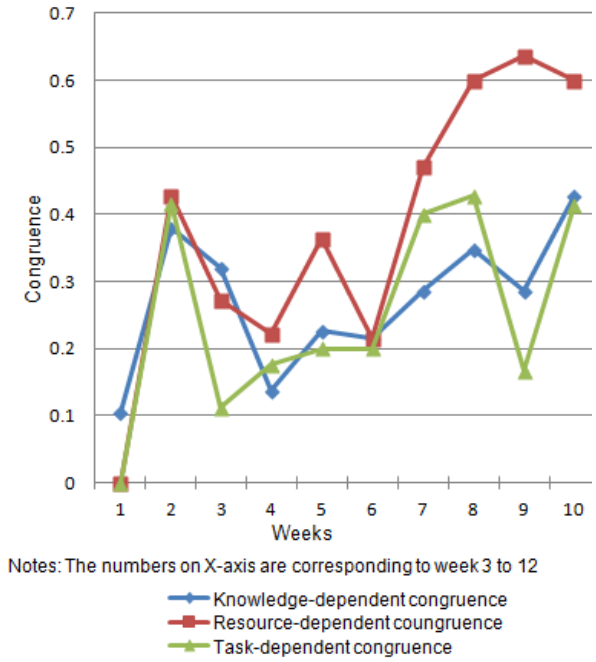

Figure 3. Comparisons of the three congruence measures over time

meeting records kept about all teams, it was found that the changes reflected the performance changes of the teams, which gradually matured, achieving more and progressing faster.

We found that the changes in the pattern of task-dependent congruence are different from those of knowledge-dependent congruence and resource-dependent congruence. Task-dependent congruence exhibits a slight increase but high variability. An example of the changes of the three congruence measures for Team 1 is illustrated in Fig. 3. There are many possible causes for this variability; for example, team members' involvement in certain task may not have been needed at the time when it is supposed to be, due to changed circumstance; or some team members may have been sick or a certain task may have been too big and more resources had been allocated to the task; or some team member might have been incapable of accomplishing the task assigned. All these situations lead to a mismatch between communication matrix and task dependent-matrix. These, in turn, reduce the values of task-dependent congruence.

Moreover, the task dependency might change due to changes in requirements or a better understanding of the requirements. Even though there are some differences, the general consistency of these trends suggests that the overall team socio-technical congruence did increase over time.

The knowledge congruence measure and performance mark of the seven teams over ten weeks are shown in Table 3. The correlation relationships between the three measures and each team's performance mark over the 10 weeks are presented in Table 4. It can be seen from Table 3 and 4 that:

1) the correlation coefficients between the knowledge congruence measure and the performance mark for all teams are higher than those between the task congruence measure and the performance mark.

2) the correlation coefficients between the resource congruence measure and performance mark for all teams are higher than those between the task congruence measures and the performance mark.

3) the teams with higher correlation coefficients in all three measures generally have better organized processes, and better project assessment marks during and at the end of the project. For instance, team 3, 5, 6, and 7 have used better processes during the project, and obtained better marks at the end of the project compared to teams 1, 2, and 4.

4) All the correlation coefficients are statistically significant even though some correlation coefficients are low. For instance, the correlation between task congruence and each team's performance mark for teams 1 and 2 is weak; yet they are statistically significant.

Overall, the knowledge-dependent congruence and resource-dependent congruence measure provide better pictures of the performance changes of the teams than that of task congruence measure. These results are consistent with our empirical observation on students group dynamic during the project:

- A student team tends to communicate more to share knowledge when the knowledge required to complete the tasks are very demanding during a certain period of time in the project based on our seven teams' performance records. This can be seen from Fig. 3 where the knowledge congruence measure changes over the ten weeks indicating the changes of the communication.
- A student team tends to perform better if team members communicate more to share knowledge, discuss about using resources, coordinate tasks effectively, and help each other frequently.

This conclusion also shows that our assumption made in the first section is valid. Thus, it is reasonable to infer that the three congruence measures collectively provide convincing information about the quality of software processes and team performance. Consequently, it is possible to use these three measures as indicators for assessing quality of the team processes and the performance of the teams. Using the model to further examine the performance of more teams is part of our future research.

B. Threat to Validity

Although considerable effort was expended in collecting information on the student teams, it is likely that some interactions were missed. For example, although student forums were provided for students to discuss the project, students likely used email to discuss some project-

TABLE 3. KNOWLEDGE CONGRUENCE MEASURE AND PERFORMANCE MARKS OF EACH TEAM IN EACH WEEK (OVER 10 WEEKS)

Weeks	Knowledge congruence measure for each team							Team performance marks						
	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7	Team1 mark	Team 2 mark	Team 3 mark	Team 4 mark	Team 5 mark	Team 6 mark	Team 7 mark
Week3	0.1053	0.0937	0.0842	0.0828	0.0854	0.1032	0.1000	5.21	6.00	5.20	6.90	5.90	6.76	7.00
Week4	0.3793	0.3258	0.0910	0.3179	0.3063	0.3870	0.2560	5.21	6.60	6.90	7.00	6.20	7.14	7.00
Week5	0.3200	0.3168	0.9980	0.2918	0.3067	0.3393	0.3147	9.44	9.35	9.23	8.00	9.05	8.48	9.00
Week6	0.1379	0.1429	0.1472	0.1683	0.1572	0.2000	0.1708	7.08	7.33	7.55	8.00	8.07	8.80	8.77
Week7	0.2273	0.2250	0.2627	0.2170	0.2538	0.2593	0.2283	8.89	8.80	9.00	9.00	9.93	8.00	8.93
Week8	0.2143	0.2571	0.2546	0.2156	0.2973	0.2357	0.2188	10.00	9.50	9.00	9.70	9.00	10.00	10.00
Week9	0.2857	0.2543	0.2711	0.3021	0.3011	0.2860	0.3371	6.67	5.94	7.00	7.00	7.03	7.01	7.87
Week10	0.3478	0.2783	0.3249	0.3826	0.3373	0.3409	0.4162	8.26	8.80	8.00	9.00	10.67	8.82	9.88
Week11	0.2857	0.2417	0.2923	0.0279	0.2997	0.3086	0.3467	9.44	8.90	9.00	9.00	10.00	9.72	9.00
Week12	0.4286	0.3823	0.4766	0.5057	0.4847	0.4671	0.4320	10.00	9.00	10.00	10.00	10.00	10.90	10.00

TABLE 4. CORRELATION BETWEEN TEAM'S MARK AND MEASURES OF KNOWLEDGE CONGRUENCE, RESOURCE CONGRUENCE, AND TASK CONGRUENCE

		Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7
Correlation between knowledge congruence and each team's weekly mark (over 10 weeks)	Correlation	0.3112	0.4609	0.592	0.2689	0.5608	0.451	0.5453
	p-value	1.67 E-07	6.82 E-06	2.66 E-06	1.17E-09	4.15E-08	6.26E-09	1.68 E-07
Correlation between task congruence and each team's mark in each week (over 10 weeks)	Correlation	0.004	0.0745	0.6394	0.1037	0.3959	0.2712	0.3713
	p-value	1.54E-07	7.32E-06	1.60E-06	1.07E-09	3.47E-08	5.87E-09	1.60E-07
Correlation between resource congruence and each team's mark in each week (over 10 weeks)	Correlation	0.3812	0.3585	0.7052	0.2899	0.6625	0.4327	0.4741
	p-value	1.65E-07	7.21E-06	2.30E-06	7.94E-10	3.87E-08	5.15E-09	1.61E-07

related issues and did not report this in the weekly journal. This would impact the calculation of congruence. To examine this, members of all teams were queried about their email activity. The members of all teams confirmed that the information provided in the weekly journal was accurate except for two team members who were not available to answer. This verification suggests that the journals are a reasonable reflection of actual interaction. A second validation issue is that SVN-commits might not present the entire picture of who had done what; for example, some tasks committed by an individual team member might have actually been completed with help by others.

We acknowledge that the scale of projects in industry is different from student term projects even though a controlled experiment approach was used and a rigours data collection process was followed. There are also a number of constraints for student teams to maximize communication as students also have other courses at the same time when they were doing the project. We believe that professional software developers have a much higher level of communication among team members resulting in a higher level of social-technical congruence. However, it should be mentioned that the results reported in the research were achieved with student teams. The generalisation of these conclusions to industry requires more rigorous investigation and is subject to future research.

VI. CONCLUSION AND FUTURE WORK

This paper proposed using socio-technical congruence measures to analyse software process quality and the performance of student teams. Two new socio-technical congruence measures: knowledge-dependent congruence and resource-dependent congruence were proposed based on the task-dependent congruence measure proposed in literature [3, 4]. Data was collected from seven student teams working on a semester-long project. Ten meta-networks were created, one per week, for all seven teams. Each meta-network contained all the records of team members' weekly SVN-commits; weekly interactions and task completion information; resources used information; and team members' knowledge/skill information. Congruence measures defined in the research were calculated for each team over the ten weeks. Using this data, team dynamics were examined and it was found that the increase in knowledge and resource congruence was not reflected in an increase in task congruence, and that the congruence measures are correlated to the teams' performance. This research has four main contributions:

1) Two new socio-technical coordination requirements matrices were proposed to supplement task-dependent congruence: knowledge-dependent

congruence and resource-dependent congruence requirements matrices.

2) It was demonstrated that examining changes in these congruence measures provides insight into the project life cycle and the evolution of software development processes.

3) It was found that knowledge- and resource-dependent congruence has similar trends over time and can be at odds with that of task congruence.

4) The three congruence measures are correlated to the quality of the processes and performance of the teams over the ten-week project. This indicates that the measures provide reliable information about the quality of software processes and the performacne of teams.

The authors' future work will focus on:

- using the congruence measures as indicators to monitor and assess the quality of software processes of more student teams.
- applying the congruence model to an industry project to validate the model.
- developing an effective tool to support mining SVN data, computing and analysing the congruence measures.

TABLE 5. AN EXAMPLE OF A TEAM MEMBER'S WEEKLY JOURNAL IN WEEK 8 AND WEEK 11

Name	***Person1***		**Group No. 1**
ID	49778		
Week No.	**Activities**	**Duration/ time (hours)**	**Details**
……	……	……	……
8	Meetings	2	
	SDD design	5	explain the Architecture of our System to *Person2* and *Person3*
	SDD Writing	5	Write the SDD documentation for 'Class Diagram' section
	SDD proof reading	8	Proof reading with Person2 for SDD chapter 4, 5, 6
	maintain the svn	2.5	maintain the SVN folder, add draft2 folder for SPMP and final folder for other documents
	back up the milestone	1.5	back the milestone, some compile problem occurs
……	……	……	……
11	Meetings	2	
	SDD re-write	4	Work with *Person2* and *Person3* produce the final version of sdd
	rewrite the delete nogozone function	2	rewrite the delete nogozone function, and change the behaviours of it
	continue working on the manual operation of robot	9	working with *Person4* to implement the behaviour of robot's manual control. Mine detecting and marking
	assembling the whole project	5	assembling the whole project, implement the build file to each aspects. Make sure every make file is working.
	write the jar commands	3	package the whole project into an executed jar file

Notes: (1) The content is presented as it was in the original document, with no grammar correction and no content modification except for font adjustment and substituting team members' names with *Person1* to *Person4.*
(2) *SDDs* are mapped to SDD1 or SDD2 according to the time scheduled in the team's SPMP document, where SDD1 represents the first draft and SDD2 represents the final version of the document.

TABLE 6. RESULTS OF KNOWLEDGE/SKILL SURVEY

Skills	**St1**	**St2**	**St3**	**St4**	**St5**	**St6**
Programming	9	5	5	7	7	4
Problem analysis	8	7	7	8	7	6
Architecture design	9	7	7	7	7	6
Robot design	7	7	9	8	7	8
Interface design	9	6	6	5	6	4
Testing	8	7	7	7	9	6
Using Tools (Make, Ant, SVN, Eclipse, testing tools, etc.)	10	8	8	7	7	7
Communication (including social skills)	7	9	7	8	5	4
Writing (documentation)	7	9	8	6	7	6
Capability of learning	8	8	8	8	8	8

Note: Each team member, (represented as St1 to St6 in the table) was asked to enter numbers 1 to 10 in the cells above. Number 1 represents that the team member (including himself/herself) had the lowest skill, and number 10 represents that the team member had the highest skill.

REFERENCES

[1] A. Fuggetta, "Software process: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering. Limerick, Ireland* 2000, pp. 25-34.

[2] M. Jirotka, "Requirements engineering: social and technical issues," *London: Academic Press.,* 1994.

[3] M. Cataldo*, et al.*, "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity," in *Proceedings, Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* Kaiserslautern, Germany, 2008, pp. 2-11.

[4] M. Ashworth, and Kathleen M. Carley, "Who you know vs. what you know: The impact of social position and knowledge on team performance," *Journal of Mathematical Sociology,* vol. 30, pp. 43-75, 2006.

[5] L. McLeod and S. G. MacDonell, "Factors that affect software systems development project outcomes: A survey of research," *ACM Computing Surveys (CSUR),* vol. 43, p. 24, 2011.

[6] M. E. Conway, "How do committees invent?," *Datamation,* vol. 14, pp. 28-31, 1968.

[7] E. S. Raymond, *The new hacker's dictionary*: The MIT Press, 1996.

[8] R. M. Burton*, et al.*, *Strategic organizational diagnosis and design: Developing theory for application*: Kluwer Academic Pub, 1998.

[9] M. Cataldo*, et al.*, "Identification of Coordination Requirements: Implications for the design of collaboration and awareness tools," presented at the Conference on Computer Supported Cooperative Work, Banff, Alberta, Canada, 2006.

[10] R. P. Bostrom and J. S. Heinen, "MIS problems and failures: a

TABLE 7. AN EXAMPLE OF AGENT-TO-TASK NETWORK IN WEEK 4

	s49778	s49780	s64836	s74493	s80109	s80205
ManagementTask	10	1	0	1	7	0
SRS1	6	8	8	1	0	9
SPMP1	2	0	0	0	0	0
GUIDevelopment1	10	0	0	4	3	0
RobotMove1	3	0	0	3	3	0
RobotMove2	0	0	0	0	0	0
ArchDesign	5	0	0	5	0	0
MapDesign	0	0	0	0	0	0
CommunicationDevelop	0	0	0	0	0	0
SDD1	2	0	0	8	0	0

Note: (1) For the purpose of presenting the table effectively, the task-to-agent network from Week 4 is presented, which is the transpose of the agent-to-task network from Week 4.
(2) The number in each cell represents how many changes were made by each team member with respect to the tasks.
(3) Only the tasks relevant to Week 4 are presented in the table.

TABLE 8. METRICS USED TO ASSESS THE TEAMS' PERFORMANCE IN THE SOFTWARE DEVELOPMENT PROCESSES

No.	Metrics	Assessment Artefacts and/or Assessment Time	Activities	Processes Defined in ISO 12207
1	Number of open-ended questions asked, number of close-ended questions asked	Weekly meetings (mainly at the first two weeks) with lecturers	Requirements elicitation	Acquisition process
	Number of requirements presented in SRS, and the number of requirements has high level of clarity and are correct in the SRS; percentage of requirements presented in the SRS (completeness)	The quality of requirements in software requirements specification (SRS).		
2	Percentage of functional requirements covered in the architecture design and detailed design; percentage of non-functional requirements covered in the architecture design and detailed design.	Architecture design, and detailed design in software design document (SDD)	Design	Development process
	Lines of code committed and quality of code measured by the level of conformance of the code to the coding convention and coverage of system functionality implemented by the code.	Code in SVN	Coding, integration,	
	Number of testing cases executed and the ratio of system requirements covered by the testing cases; the level of using testing tool in the testing.	Testing cases committed in SVN	Testing	
3	Level of addressing version convention issue, and change management issues in SPMP; level of using SVN measured by the frequency and adequacy of using the tool in each week.	Configuration management plan in software project management plan (SPMP) and SVN commits made in each week.	Configuration identification, control and evaluation	Configuration management process
4	Number of lines of code reviewed by the team members by following code review procedure and using code review template	Code review documents in SVN on weekly basis	Quality review	Quality assurance process
5	Number of function requirements implemented correctly in the demo in each week; quality of the demo measured by the successful rate and failure rate of the functionality presented.	Software demos in weekly meeting with lecturers	Code verification and integration verification.	Verification process
6	The quality of the SRS,SDD, SPMP measured by the marks given to these documents. (There is a rubric for each document, and each document is assessed twice for the draft version and the final version)	SRS(assessed in weeks 4, 5, 12), SPMP(assessed in weeks 6,7, 12), SDD (assessed in weeks 8, 9, 12),	Documentation activities	Documentation process
7	The effectiveness of the team organisation measured by the presentation mark of each team assessed by lecturers in each week	Weekly meetings and timely delivery of the required artifacts	Management activities	Management process

socio-technical perspective, part II: the application of socio-technical theory," *MIS quarterly,* pp. 11-28, 1977.

[11] A. Sarma*, et al.*, "Challenges in measuring, understanding, and achieving social-technical congruence," in *Proceedings of Socio-Technical Congruence Workshop, In Conjuction With the International Conference on Software Engineering*, Leipzig, Germany, 2008.

[12] G. Valetto*, et al.*, "Using software repositories to investigate socio-technical congruence in development projects," in *Proceedings of the Fourth International Workshop on Mining Software Repositories* 2007, p. 25.

[13] G. Madey*, et al.*, "The open source software development phenomenon: An analysis based on social network theory," in *Proceedings of the Americas Conference on Information Systems (AMCIS2002)*, 2002, pp. 1806-1813.

[14] T. Wolf*, et al.*, "Mining task-based social networks to explore collaboration in software teams," *Software, IEEE,* vol. 26, pp. 58-66, 2009.

[15] C. F. Kemerer, "An agenda for research in the managerial evaluation of computer-aided software engineering (CASE) tool impacts," 1989, pp. 219-228 vol. 2.

[16] R. L. Glass, "The ups and downs of programmer stress," *Communications of the ACM,* vol. 40, pp. 17-19, 1997.

[17] S. Sawyer and P. J. Guinan, "Software development: Processes and performance," *IBM Systems Journal,* vol. 37, pp. 552-569, 1998.

[18] M. C. Paulk, *The capability maturity model: Guidelines for improving the software process* vol. 441: Addison-Wesley Reading, MA, 1995.

[19] R. CONSTANTINESCU and I. M. IACOB, "Capability Maturity Model Integration," *Journal of Applied Quantitative Methods,* vol. 2, p. 187, 2007.

[20] R. Tricker*, et al.*, *ISO 9001: 2000 in Brief*: Butterworth-Heinemann, 2001.

[21] IEEE, "Software life cycle processes," in *(ISO/IEC 12207) Standard for Information Technology*, ed, 1998.

[22] G. H. Subramanian*, et al.*, "Software quality and IS project performance improvements from software development process maturity and IS implementation strategies," *Journal of Systems and Software,* vol. 80, pp. 616-627, 2007.

[23] J. Y. C. Liu*, et al.*, "The impact of software process standardization on software flexibility and project management performance: Control theory perspective," *Information and Software Technology,* vol. 50, pp. 889-896, 2008.

[24] S. Sawyer*, et al.*, "Social interactions of information systems development teams: a performance perspective," *Information Systems Journal,* vol. 20, pp. 81-107, 2010.

[25] K. M. Carley, "Smart agents and organizations of the future," in *The Handbook of New Media*, L. Lievrouw, and Sonia Livingstone, Ed., ed Thousand Oaks, CA: Sage, 2002, pp. Chapter 12, 205-220.

[26] C. B. Seaman and V. R. Basili, "Communication and organization in software development: an empirical study," *IBM Systems Journal,* vol. 36, pp. 550-563, 1997.

[27] M. Hertzum, "The importance of trust in software engineers' assessment and choice of information sources," *Information and Organization,* vol. 12, pp. 1-18, 2002.

[28] T. Chau*, et al.*, "Knowledge sharing: agile methods vs. Tayloristic methods," in *Proceedings of Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003*, 2003, pp. 302-307.

[29] L. Jiang*, et al.* Measuring team performance using socio-technical congruence, Technical Report [Online]. Available: http://cs.adelaide.edu.au/~ljiang/Publications_TechnicalReport.htm

[30] K. Carley*, et al.* (2010, 5 March 2011). *ORA User's Guide 2010*. Available: http://www.casos.cs.cmu.edu/publications/papers/CMU-ISR-10-120.pdf

[31] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge: University of Cambridge Press, 1995.

[32] J. Lonchamp, "A structured conceptual and terminological framework for software process engineering," in *Second International Conference on the Continuous Software Process Improvement*, 1993, pp. 41-53.