Electronic Research Archive of Blekinge Institute of Technology
<http://www.bth.se/fou/>

This is an author produced version of a conference paper. The paper has been peer-reviewed but may not include the final publisher proof-corrections or pagination of the proceedings.

Citation for the published Conference paper:

Title:

Author:

Conference Name:

Conference Year:

Conference Location:

Access to the published version may require subscription.

Published with permission from:

Socio-Technical Congruence Sabotaged by a Hidden Onshore Outsourcing Relationship: Lessons Learned from an Empirical Study

Darja Šmite^{1,2}, Zane Galviņa²

¹ Blekinge Institute of Technology (Karlskrona, Sweden)

² University of Latvia (Riga, Latvia)

Darja.Smite@bth.se, Zane.Galvina@lu.lv

Abstract. Despite the popularity of outsourcing arrangements, distributed software development is still regarded as a complex endeavor. Complexity primarily comes from the challenges in communication and coordination among participating organizations. In this paper we discuss lessons learned from participatory research carried out in a highly distributed onshore outsourcing project. Previous research established that socio-technical congruence principles alleviate distributed work. In practice we have found that alignment between the systems structure and organizational structure can be studied from different abstraction levels and also during different phases of project lifecycle. We have found that official organizational structure differed from the applied one, which meant that the planned alignment in task allocation strategies was broken. Our findings indicate that the lack of socio-technical congruence caused several implications, including unclear responsibilities, delays in problem turnaround, conflicting changes, and non-delivered parts.

Keywords: Distributed software development, onshore, outsourcing, socio-technical congruence, Conway's law

1. Introduction

The topic of distributed software development emerged with the popularity of globalization. Distributed projects involve geographically dispersed team members and despite the widespread utilization empirical studies show that distributed teams are far less productive than co-located teams [1]. The majority of the studies focusing on distributed software development investigate offshore outsourcing projects [2], which means that the work is performed in a sub-contracting relationship among companies from different countries. This is also the main focus in global software projects [3]. However, not all distributed software projects are necessarily global. While multinational organizations perform on a global arena, it is not uncommon that small and medium software development companies team up locally to be able to

compete for larger contracts. Furthermore, Balaji and Brown claim that current trends in outsourcing are moving toward a multi-vendor arrangement, in which multiple vendors are pooled together to achieve and exceed the overall expertise required for a project [4]. If distributed projects are challenging due to communication, coordination and control among the partners [1], it is fair to assume that the more participating companies, the more challenging is their collaboration. In fact, an empirical investigation demonstrated that increase in the number of sites decreased quality and profits for the company leading distributed software development projects [5].

In this paper we investigate a highly distributed project that involves four different companies involved in software development. The collaboration studied is onshore outsourcing — physically local companies, which are distinct entities [2]. The rest of the paper is organized as follows: in Section 2 we present related work. Section 3 describes research methodology. Findings are presented in Section 4 and discussed in Section 5. Section 6 concludes the paper with the answers to research questions and outline of future work.

2. Related work

While distributed work requires new process models for managing team relationships to deliver software on time and within budget, such dedicated models are yet to be developed [2]. Nonetheless, different aspects of distributed work are widely discussed in the research literature. Among the most popular are the topics related to coordination of work and task allocation in particular [1, 6, 7, 8], and difficulties regarding team efficiency [9]. It was found that coupling between tasks decreases productivity [1]. Empirical evidence also suggests that communication and coordination across different locations usually requires more people and thus results in overhead [10]. One way of reducing this overhead is to modularize development and minimize technical and thus social dependencies [6]. If the work is partitioned into autonomous units without any complex dependencies, it might enable concurrent development and might not necessarily result in lower performance [5]. This line of research applies the principles of the Conway's law, which suggests that software design mirrors communication structure of the organization that builds it [9].

In the light of distributed software development the importance of Conway's law and its implications grows. The harmony between the organizational and architectural structures, which is often referred to as the socio-technical congruence, is not always easy to establish. One of the reasons for this is the rapidly changing nature of the organizations already discussed by Conway [9]. In distributed projects task allocation can be driven by on demand availability of resources, and change in the course of the project. Therefore, predicted inefficiency caused by coordination and communication breakdowns becomes evident. However, there is another important challenge. To reduce the costs even further some outsourcing service suppliers start delegating tasks to their subsidiaries or third parties. Thus, organizations who initially establish the project structure might not be aware of the true social structure, and thus fail to comply to the rules of socio-technical congruence.

In this paper we study the implications of an unintentionally non-congruent socio-technical structure of a highly distributed software development project. Social structures in distributed teams are represented by organizational units and distributed developers as suggested by [6, 11]. While the most studied technical structures are usually related to source code or development tasks [6, 11], these are not the only artifacts by which articulation of work can be carried out. Similarly to de Souza et al. who suggest that software artifacts can reveal the relationship between technical and social structure of large-scale development projects [11], in this paper we aim at exploring the socio-technical relationship using a number of different work products during the life cycle of a highly distributed software development project. Our case sheds the light on potential challenges for task allocation introduced by the differences between the planned and the actual social-technical structures.

3. Research methodology

Research reported in this paper was participatory in nature — the second researcher (Zane Galvina) was directly involved in an industrial software development project, in which she participated as a system analyst. Participatory research method addresses the gaps between the researchers and the researched people, and provides a great potential for rich empirical observations. In particular, participant observation are said to be useful for gaining a deeper understanding of the physical, social, and cultural contexts, relationships, ideas, norms, and events; and people's behaviors and activities, which were not necessarily included in the study design from the very beginning [12].

The research focused on investigating a highly distributed onshore outsourcing project and collaboration among four software companies which we for confidentiality reasons refer to as D1, D2, D3 and D4 (see also Figure 1). Due to a complex organizational structure and social relationships the project coordination was challenging. These challenges triggered an exploratory study and motivated us to seek the answers to the following research questions:

RQ1: Does the studied highly distributed project follow socio-technical congruence principles?

RQ2: What are the consequences of non-congruence?

The data was collected from a variety of sources (see Table 1). Official project plans were used to outline the planned social structures, while observations from participatory research activities formed the basis for identifying the actual social relationships. These were further supplemented by an analysis of different project artifacts that formed the basis for studying the technical structures and relationships. The socio-technical links were established through the task allocation strategies outlined in the project plans, and compared with those suggested by the Conway's principles. The socio-technical relationships are visualized with the help of diagrams that follow original Conway's notations [9]. In particular, we have created diagrams

for planned and actual social structures, technical structures, which are further linked through the socio-technical structures.

Table 1. Data collection

Project phase	Artifacts collected	Observations
Requirements analysis and design	<ul style="list-style-type: none"> • Project Management Plan • Software Requirement Specification • Software Design Specification • Problem reports 	<ul style="list-style-type: none"> • Interviews with users • Weekly meetings with D4 and D3 • Participation in two meetings among D1, D3, D4 to finalize the requirements and design documentation
Development	<ul style="list-style-type: none"> • Problem reports 	<ul style="list-style-type: none"> • Participation in the virtual weekly meetings at D4 • Participation in demo sessions at D1
Testing	<ul style="list-style-type: none"> • Problem reports 	<ul style="list-style-type: none"> • Participation in the weekly virtual meetings with D4 • Participation in demo sessions regarding fixes

Our research has several limitations. First of all, the focus of this exploratory study is to illustrate only one plausible challenge in coordinating work in a highly distributed project and by no means implies that similar socio-technically non-congruent projects would suffer from the same consequences discussed in this paper. Secondly, our findings may be affected by a single researcher's bias, since the case description is based on observations from the participatory research. This was mitigated through triangulation of the observations with the actual project documentation.

4. Results

In this section we present the results and lessons learned from the study. We start by describing the project background and socio-technical structures that we have found in different phases of the project life cycle. Our results illustrate how even having best intentions to comply with the rules of socio-technical congruence the project may suffer from inefficient structure due to unforeseen circumstances.

4.1. Project overview: social and technical structures

The project discussed in this paper was a bespoke software development project for two customer organizations, which started in the beginning of 2011. The aim of the project was to move an existing software system to a new platform and significantly enhance its functionality. The project was motivated by the necessity to comply to the

new legal requirements as well as technical obsolescence, while the platform change was necessary to enable complex integration with the other systems existing in the organizations. Customer organizations organized a tender and contracted development to the winning organization. Unfortunately, the winning software company neither possessed required experience and expertise in all knowledge domains nor had sufficient number of on demand available developers to fulfill the requirements on their own. Therefore the development in practice was performed by a network of small software companies collaborating on a joint project (see Fig. 1).

The project team consisted of 21 employees from all participating organizations. Organization D1 was the prime contractor who has won the tender and contributed with the largest share in the project. At the time of the study D1 was an SME employing approx. 140 employees; ten of them were involved in this project. D2 was a small company with only 16 employees and involved five developers in the project. Even smaller organizations were D3 and D4, each employing approx. ten people. Three employees from each of the latter organizations were involved in the project. It is worth noting that the customer organizations engaged 13 employees in the project, who were available during different phases of the lifecycle.

Fig.1 is created on the basis of official project management plan and supplemented with our observations. It demonstrates the organizational structure and contractual relationships made by the prime contractor (D1). The entity D4 is added to reflect the actual structure that was discovered through observations and participatory involvement in D3 activities. In particular, we have found that D3, one of the direct sub-contractors of D1, further outsourced parts of the work to another company (D4). However, the existence of this outsourcing relationship was hidden from the prime contractor (indicated by the dashed line in the figure) and the employees from D4 were presented as the employees from their contractor's company (D3). Conway in his proposition did not address hidden structures and their implication and we found politically flavored relationships and its implications interesting to explore.

Fig. 1. Organizational structure

The technical structure of the system that was developed comprised of two separate sub-systems, which utilize the same data for different purposes. Sub-system 1 allowed

users to enter the data, while Sub-system 2 was developed for dissemination purposes. In this paper we focus our attention on exploring the socio-technical dependencies within Sub-system 1. An overview of the technical system's structure is given in Figure 2. The figure also outlines the integration requirements to support internal and external interfaces. In the figures we use original notations proposed by Conway [9].

Fig. 2. Product structure

The project was broken into sub-projects and further managed applying Rational Unified Process (RUP). The delivery was expected by December 2011, however due to a three month delayed iteration in one of the sub-projects, the whole project was significantly delayed. To understand the reasons for the failure of this distributed project to meet the deadline we investigate the socio-technical congruence through studying the task allocation strategies in this onshore outsourcing network for different project artifacts.

4.2. Task allocation: socio-technical links

As mentioned earlier, modularization on the sub-system level motivated division of the project into two sub-projects. Due to unavailability of resources in one place, the work on Sub-system 1 was further split into four components, which were assigned to different organizations. According to Conway's law, the task allocation shall follow a

homomorphic approach [9]. In other words, each organizational unit can work on several components, while each component must be assigned to only one unit. From the task allocation strategy we can easily see that system development in the project studied did not fully apply the congruence principles proposed by Conway. In practice, Sub-system 2 was developed by two organizations (D1 and D2), three different organizations were involved in developing Component 1 for Sub-system 1, and while Component 2, 3 and 4 were allocated solely to a single organization, some of the social links necessary to support the technical dependencies did not exist. In particular, the interface between Components 1 and 4 was not supported by the links between D1 and D4.

Fig. 3. Task allocation illustrates socio-technical non-congruence in Sub-system 1

Lessons learned: Work on Component 1 aimed at developing the database solution for the Sub-system, and the tasks were shared between D1, D3 and D4. It was decided that each organization would develop the part that represents and communicates with the other components that are developed by respective organizations. This however breaks the homomorphic principle of task allocation. In practice, the three organizations shared the work on the same component level and notably two of them did not have any direct contact (see Figure 4). The missing link in this case impacted the way changes were handled. When D1 implemented the changes necessary for supporting the interface between sub-systems, they impacted the parts that were responsible for supporting the interface between Component 1 and Component 4. Since no direct communication was established between D1 and D4, and in the light of poorly documented interface specifications, the changes remained unnoticed until it

caused a failure when D4 tested their parts of the sub-system. In result, it took approximately two weeks for D4 to find the cause of this failure. Thankfully their solution to the problem did not cause another loop of errors.

Fig. 4. Missing relationship between organizational units working on Component 1

Significant challenges were introduced by the work on Component 4, which was assigned to organization D4. The component has one external and two internal interfaces. Since the prime organization (D1) was formally responsible for the integration tasks, communication was required between D1 and D4 to effectively handle these tasks. This link was, however, not established (see Figure 5).

Fig. 5. Missing relationship between organizational units working on Component 4

In practice, interface specifications were poorly documented and since D4 was not involved in specifying these requirements they assumed that all integration would be solved by D1. Due to missing direct contact between the two organizations, and misunderstood responsibilities, the integration part was missing. This was discovered one week before the delivery deadline and caused a loop of blaming between D1 and D3, which was formally responsible for the work.

4.3. The impact of socio-technical non-congruence during requirements analysis and design

In order to get a better understanding of the task allocation problems and project coordination breakdown, we further illustrate the responsibilities shared by the four organizations in the requirements analysis and design phases. During systems analysis each organization (except for the hidden organization D4) interacted with the customers in order to elicit requirements for the system parts within their responsibility. Each organization contributed in development of the software requirement specification (SRS) and software design specification (SDS), which were finally reviewed and integrated into one package by D3.

Lessons learned: Fragmentation during the requirements elicitation led to dissatisfaction of the customers, since different organizations contacted the same prospective users and often asked the same questions.

Most importantly, the reasons of failure in delivering the integration parts could be traced back to requirements analysis and design phases. Since formally each organization focused on specifying the functionality of their respective components and sub-systems, integration parts were poorly documented. Organizational dispersion resulted in limited social interaction, and confusion regarding the interfaces remained unsolved. While formal responsibility for integration, as well as the overall project coordination, was assigned to the prime organization (D1), hidden parts of the organizational structure were invisible. Therefore threats to coordination of work on Component 4 and related interfaces were not foreseen.

4.4. The impact of socio-technical non-congruence during testing

Testing activities were performed stepwise. First of all, responsibility for testing the developed parts was assigned to each respective organization (D1, D2, D3, and D4). Then an additional systems testing was performed by the prime organization (D1). This included testing systems integration, during which a missing interface was detected.

Later acceptance testing was coordinated by D1. Trouble reports from the customers were gathered centrally by D1, and then assigned to responsible organizations (D2 or D3 by the D1, and further to D4 by D3, if necessary).

Lessons learned: From this study we learned that strict modularization of work and isolated functional testing that excluded testing the interfaces prevented early identification of the missing integration links.

Additionally, coordination of trouble report resulted in significant time overhead. Mediation of problems prolonged communication paths and therefore resolution intervals. The delay was especially noticeable in coordination of changes in Component 4, for which reports were reassigned twice. Direct interaction between D1 and D4 would have shortened the turnaround paths of the tasks and potentially communication overhead in case of misunderstandings and clarifications.

5. Discussion

Related studies suggest that distributed projects shall have a clear distribution rationale and only consider well structured, well understood and stable projects, decomposable into discrete tasks [7]. This is in line with the socio-technical congruence principles proposed by Conway [9] who promoted the alignment, or a homomorphic relationship, between the tasks and organizational units. To understand the existence and impact of these relationships we have performed a study of a highly distributed onshore outsourcing project. We have found that identification of the socio-technical congruence requires careful attention, as parts of the organizational structure can be hidden. The project described in this paper involved a hidden outsourcing relationship between two organizations (D3 and D4), which the prime contractor, who is responsible for responsibility allocation, was not aware of. The implication of this is that the socio-technical congruence was significantly affected.

When studying the degree of alignment to understand the reasons for poor task allocation, we also noticed that different levels of abstraction provide us with a different view (see Fig. 6).

- From the customers' perspective the system and its developing organization are perfectly aligned, since the development of the system is assigned by the contract to the prime contractor (D1).
- When studying the prime contractor's perspective, we see a strict decomposition of the system with only one shared component. The technical structure of the system contains the database in Sub-system 1, which is usually difficult to isolate, as it is used by other components. Also the social structure is simple — D1 communicates directly with D2 and D3. We therefore conclude that the socio-technical structure was designed to ensure homomorphic relationship between the organization (social structure) and the system (technical structure) with one exception.
- However, our findings indicate that the actual socio-technical structure was different from that planned by D1. When adding D4 into the social structure, the homomorphic socio-technical relationship breaks significantly. We observe that one component is now assigned to three organizations, two of which are not socially linked. Isolation of problems and coordination of responsibilities for this vulnerable component (the database in Sub-system 1) was thus problematic, as could be predicted by the Conway's proposition.

Customers' perspective

Prime contractor's perspective

In practice

Fig. 6. Task allocation illustrates socio-technical non-congruence in Sub-system 1

The misalignment identified had an impact on the way the work was coordinated, and also resulted in several misunderstandings. The missing link between D1 and D4 meant that there was no direct communication and thus all the necessary clarifications or problem escalations were organized through D3. This caused delays in problem turnaround and inability to react on the changes during the course of the project, which confirms existing findings from studying speed and communication in distributed teams [10]. At the same time, we have found that some of the interfaces were not developed on time due to confusion regarding responsibilities, and that shared components caused misunderstandings when implementing the changes to existing working parts of the system. The problem of unclear roles and responsibilities is also discussed in existing research on global teams. For example, Kotlarsky et al. [13] found that participants of a global software development project often had different views on their own or their colleagues' responsibilities.

We believe that many of these problems could be avoided, if the interfaces between the system's components and sub-systems would be well documented, and if the organizational structure would have been clear.

Lings et al. suggest that a distributed project can be partitioned functionally according to organizational and systems structure or by process during other life-cycle phases prescribing natural divisions of work in relatively small bundles [7]. We have found that the work allocation strategies during requirements analysis, development and testing did not follow the same pattern. Notably, the major challenges in this respect can be related to documenting requirements for, developing and testing of interfaces among the systems components.

Although it has been noted that increase in the number of sites usually results in the increase productivity (since the development capacity grows) [5], the same study demonstrates that the quality of the developed software decreases. Our observations indirectly support this view, as the growing complexity of coordination development of interfaces between the components that were allocated to different sites, resulted in a missing interface.

6. Conclusions and further research

In this paper we have discussed lessons learned from a highly distributed project. Our observations indicate that distribution makes projects more complicated, and we have traced the major sources of complexity to be triggered by the missing communication links between the participating organizations. One of the most interesting findings is related to the fact that the official organizational structure differed from that in practice. In particular, a hidden outsourcing relationship existed, which the prime contractor, who was responsible for task allocation, was not aware of. The lack of awareness of the true organizational structure and missing direct communication between the prime contractor and the hidden organization propagated into all phases of the project lifecycle.

In response to **RQ1** we have studied the task allocation strategies from a socio-technical congruence perspective, and realized that the answer to the research question is not trivial. We learned that the project was designed to comply with the homomorphic principles proposed by Conway [9] and discussed by other researchers [6, 7], but in practice failed to follow the plan. The congruence was sabotaged by the hidden onshore outsourcing relationship.

In response to **RQ2** our findings suggest that although the project task allocation followed modularization, several important practices were missing. Incompliance resulted in unclear responsibilities assigned for documenting, developing and testing the interfaces among the modules that were further complicated by the missing communication links between the prime contractor and the hidden supplier. This caused delays in problem turnaround, conflicts with change implementation and non-delivered parts.

In conclusion we expect that a task allocation strategy that is compliant with the Conway's proposition is more likely to minimize similar problems.

Our findings are based on qualitative analysis of the gathered material. Further work will focus on qualitative measurements of the lead times for change requests during the development phase. We also hope to have insights into the destiny of the project during its maintenance.

Acknowledgement

We thank our industrial partners for the opportunity of following the project.

This work has been supported by European Social Fund project No. 2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044, as well as BESQ+ (20100311) grant from

the Knowledge Foundation in Sweden, and Center Of Nordic Excellence in Software Engineering (CONES) grant from NordForsk.

References

1. Lamersdorf, A., Münch, J., Rombach, D., "A Decision Model for Supporting Task Allocation Processes in Global Software Development". In: proceedings of the International Conference on Product Focused Software Development and Process Improvement, PROFES 2009 Oulu, 2009, pp. 332-346
2. Prikladnicki, R., Audy, J.L.N., and Shull, F., "Patterns in Effective Distributed Software Development". IEEE Software, v. 27, 2010, pp. 12-15
3. Šmite, D., Wohlin, C., Feldt, R., and Gorschek, T., "Empirical Evidence in Global Software Engineering: A Systematic Review", In: Journal of Empirical Software Engineering, Vol. 15, Nr. 1, February 2010, pp. 91-118
4. Balajjand, S., and Brown, S. A., "Strategic IS Sourcing and Dynamic Capabilities: Bridging the Gap" In: Proc. of the 38th Hawaii Int. Conf. on Systems Sciences (HICSS) — Track 8, IEEE CS Press, 2005, pp. 260
5. Ramasubbu, N., Cataldo, M., Balan, R. K., and Herbsleb, J. D., "Configuring global software teams: a multi-company analysis of project productivity, quality, and profits". In: Proceedings of the 33rd International Conference on Software Engineering (ICSE '11), 2011, pp. 261-270
6. Cataldo, M., and Herbsleb, J., "Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity". In: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM '08), 2008, pp. 2-11
7. Lings, B., Lundell, B., Ågerfalk, P. J., and Fitzgerald, B., "A reference model for successful Distributed Development of Software Systems". Proceedings of the 2nd International Conference on Global Software Engineering (ICGSE), Munich, Germany, 2007, pp. 27-30
8. Avritzer, A., Paulish, D., and Cai, Y., "Coordination Implications of Software Architecture in a Global Software Development Project". In: Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA) 2008, pp. 107-116
9. Conway, M., "How do Committees Invent?" Datamation, Vol. 14, 1968, pp. 28-31
10. Herbsleb, J. D., and Mockus, A., „An empirical study of speed and communication in globally distributed software development“. IEEE Transactions on Software Engineering, 29(6), 2003, pp. 481-494
11. de Souza, C., Froehlich, J., and Dourish, P., "Seeking the Source: Software Source Code as a Social and Technical Artifact", Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work, 2005, pp. 197-206

14 **Darja Šmite, Zane Galviņa**

12. Mack, N., MacQueen, C.W.K.M., Guest, G., and Namey, E., "Qualitative Research Methods: a data collector's field guide". Family Health International, 2005
13. Kotlarsky, J., van Fenema, P.C., and Willcocks, L.P., "Developing a knowledge-based perspective on coordination: The case of global software projects". In: Inf. Management, 45(2), 2008, pp. 96-108