An Evolutionary Perspective on Socio-Technical Congruence: The Rubber Band Effect

Stefanie Betz
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: stefanie.betz@bth.se

Darja Šmite
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: darja.smite@bth.se

Samuel Fricker
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: samuel.fricker@bth.se

Andrew Moss
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: andrew.moss@bth.se

Wasif Afzal
Bahria University
Islamabad, Pakistan
e-mail: wasif.afzal@gmail.com

Mikael Svahnberg
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: mikael.svahnberg@bth.se

Claes Wohlin
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: claes.wohlin@bth.se

Jürgen Börstler
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: jurgen.bostler@bth.se

Tony Gorschek
Blekinge Institute of Technology
Karlskrona, Sweden
e-mail: tony.gorschek@bth.se

***Abstract*— Conway's law assumes a strong association between the system's architecture and the organization's communication structure that designs it. In the light of contemporary software development, when many companies rely on geographically distributed teams, which often turn out to be temporarily composed and thus having an often-changing communication structure, the importance of Conway's law and its inspired work grows. In this paper, we examine empirical research related to Conway's law and its application for cross-site coordination. Based on the results obtained we conjecture that changes in the communication structure alone sooner or later trigger changes in the design structure of the software products to return the socio-technical system into the state of congruence. This is further used to formulate a concept of a rubber band effect and propose a replication study that goes beyond the original idea of Conway's law by investigating the evolution of socio-technical congruence over time.**

Keywords-Conway's Law, Socio-Technical Congruence, Evolution

I. INTRODUCTION

According to a widely held belief offshore sourcing enables cost reduction (in terms of time and money) [1]. Therefore, many software companies today transfer their software products or scale up the development organization utilizing human resources from different locations all over the world [2-4]. Software companies that are involved in our research projects are no exception. Some companies jump on an offshore outsourcing bandwagon and thus choose to work with company external developers, and, consequently changing the development organization during the lifecycle of software products. These changes, however, are found to impact efficiency and quality of the development [5]. In addition, if and when outsourcing relationships fail, the companies are forced to backsource the development or switch vendors [6]. Other companies choose offshore insourcing and expand through acquisitions or establishing their own sites, often in far offshore locations. This enables multi-national companies to select from a variety of choices of developing their software products here, there or everywhere. For example, Ericsson initiated development of a large software product in 2001 in Sweden, but was forced to scale up development in India in 2004, which led to a later transfer of the entire product development to India during 2009-2010, where the product is evolving to date [7]. Notably, both distribution and transferring of software development over geographic, temporal and cultural borders has been associated with numerous challenges in achieving a successful communication, coordination, and control [8, 9]. For example, tasks performed in globally distributed projects have in some cases been found to be delayed considerably compared to similar tasks in co-located projects [10]. In particular, achieving efficient communication and coordination across distance are especially challenging in geographically distributed development [9]. Consequently, industry is having a hard time in achieving the perceived benefits [1, 8].

One promising research direction that targets alleviation of cross-site coordination and communication is related to the concept of socio-technical congruence. In 1968, Melvin Conway proposed that the communication

structure is coupled with the software architecture. His thesis is formulated as follows: *"any organization which designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure"* [11]. In the past decades this law has been evaluated and revisited.

Baldwin and Clark [90] have refined Conway's law by stating that in the design of a complex system the technical and organizational structures *"will "mirror" one another in the sense that the network structure of one corresponds to the structure of the others"* [90]. Thus, they have complemented Conway's law by adding an additional direction of causality. Not only does the technical structure correspond to the organizational structure, in addition the organizational structure mirrors the technical structure.

Cataldo et al. [12] introduced a new concept based on Conway's law: socio-technical congruence. They defined socio-technical congruence as the alignment between coordination requirements extracted from technical dependencies among tasks, and the actual coordination activities performed by the engineers. Moreover, [12] provides evidence that socio-technical congruence on the task level shows a positive impact on software development productivity.

Furthermore, the organization's communication structure (i.e. social structure), design structure (i.e. technical structure) as well as the socio-technical traces used in the studies evaluating and refining Conway's law have been interpreted differently. The socio-technical traces explain how the social structures were mapped into the technical structures and vice-versa. Consequently, social structures have been illustrated using organizational structure, people structure (like teams), processes, locations, and their relations. Technical structures have been illustrated using intermediate and software artifacts and their relationships as well as derived information. The socio-technical traces used include processes, artifacts, and effects of socio-technical interaction (e.g. task assignment and file ownership).

Nevertheless, following [90] as well as [12] and assuming validity of social-technical congruence implies that changing the organizational structure alone or changing the technical structure alone is problematic. This has an important implication on software companies restructuring the development organization as a consequence of sourcing decisions. In this light, more empirical studies are needed to investigate the congruence between organizational and technical structures and its evolvement over time.

The research presented here summarizes findings from a systematic review (approach described in Section II) of empirical studies that validate the importance of socio-technical congruence by looking at the pros and cons of congruent and incongruent scenarios (presented in Section III). Based on related work we discuss how social and technical structures co-evolve in organizations that make sourcing decisions, and what can be expected in terms of compliance with socio-technical congruence. In response we present the concept of a "rubber band effect" and propose a longitudinal study that goes beyond the original idea of Conway's law by studying the evolution of socio-technical congruence over time (see Section IV).

II. Research Approach

Conway's law published in 1968 [11] marks the first documented conjecture that the structure of software under development cannot be separated from the organizational structure of the people designing it. In this paper a literature review based on forward Snowballing [13] was performed to understand how far Conway's law has been evaluated and for what purpose. The following research questions were answered with the review:

- How are socio-technical structures and traces measured?
- What are the purposes of the application of Conway's law?
- What is the empirical evidence for or against validity of Conway's law?

We did not perform a full systematic literature review [14], but a literature review with a systematic approach as opposed to ad-hoc. In our review, papers were identified based on references to Conway's original publication [11]. A search string ("How do Committees Invent") was applied to the Scopus database on February 7, 2012, resulting in 128 papers that were published between 1987 and 2011. Overall, six researchers participated in the literature review. The initial set of papers was reduced by 11 papers using the following exclusion criteria:

- Not accessible: 6
- Not peer reviewed: 4
- Duplicates: 1

The resulting 117 papers were then classified by one researcher according to the studied areas as follows: software development (9 papers), development productivity (21), global software development (20), open source (12), evolution (4), tooling (5), scalability (4), project archeology (9), software architecture (5), software use (6), software reverse engineering (5), and software quality (3) and others (14). The latter were papers mainly not focusing on Software Engineering but on history, myths, education, research, telescopes, and alternative theories. These papers have been excluded. If later in the literature review process a researcher realized a mapping was not correct, the paper has been mapped to the corresponding area.

Next, based on a discussion between the researchers taking part in the literature review, the areas of most interest in the papers for the reviewers have been selected for further data analysis. Namely:

- Major application areas: software development, software reverse engineering, and software quality;
- Software development specialties: productivity, project archeology, and architecture;

- Productivity concerns for commercial organizations: evolution and global software development.

Table I provides an overview of the selected applications and the corresponding paper count resulted from the paper classification. This selection allowed us to gain an overview of the application of Conway's law and provided sufficient depth for understanding current knowledge related to social and architectural change. As a result, in total we have analyzed 76 papers. In order to do so, each of the researchers taking part in the review selected one to three application areas for analysis. If the researchers had overlapping interests, it had been discussed and decided who is taking which area.

TABLE I. SELECTED APPLICATIONS AREAS OF SOCIO-TECHNICAL CONGRUENCE

Area	References	# of studies
Development Productivity	[12, 24-43]	21
Global Software Development	[1, 10, 44-61]	20
Software Development	[15-23]	9
Project Archeology	[66-74]	9
Software Architecture	[75-79]	5
Software Reverse Engineering	[80-84]	5
Evolution	[62-65]	4
Software Quality	[85-87]	3
	Total:	**76**

The data extraction form was developed by two researchers and tested using several publications. Besides bibliographic information regarding the publication, the following data has been extracted:

- Is Conway's law discussed, explored or validated?
- Is there evidence for or against it?
- How are the social and technical structures defined?
- What are the socio-technical traces used?
- How social and technical structures interact with each other?
- Which concepts are introduced?
- Which research questions drive the study?
- What is the information regarding the empirical studies (research method, data collection, and data analysis)?
- What are the main conclusions?

In the following, we discuss the results of the literature review. We start with the different forms of social and technical structures and the socio-technical traces identified in the analyzed papers. The socio-technical traces explain how the social structures were mapped into the technical structures and vice-versa. Two researchers have conducted the actual mapping of the references to the identified forms of technical structures, the social structures, and the socio-technical traces (see Table II-IV). One researcher conducted the initial mapping by taking all identified structures from the data extraction sheet, grouping them together and mapping the corresponding references. Next, the second researcher mapped the corresponding references to the identified grouped structures. In case of disagreement a mutual consent has been sought-after. If this was not possible the second researcher decided.

Then, we present the identified purposes of the application of the socio-technical congruence in software engineering projects. Finally, we discuss the identified evidence for and against the validity of socio-technical congruence.

III. RESULTS: SOCIO-TECHNICAL CONGRUENCE BASED ON CONWAY'S LAW IN RESEARCH LITERATURE

Different forms have been suggested in literature to illustrate the social structure in software development, see Table II. Multiple entries are possible because sometimes the social structure is measured several times using different forms. Thus, social structures have been described based on how organizations were structured into units and teams, how people were collocated and moved, people "attributes" and relationships, how work was structured with processes and tasks, how people collaborated through shared databases, meetings, how people communicated and interacted, and the co-authorship resulting from the software development.

TABLE II. FORMS OF SOCIAL STRUCTURES IDENTIFIED

Forms of Social Structures	References	# of studies
Teams	[1, 12, 18, 19, 45, 46, 49, 52, 59, 61, 73, 79]	12
Collocation and People Movement	[10, 12, 38, 49, 50, 56-58, 79]	9
Organizational Structure and Attributes	[19, 23, 38, 39, 41, 57, 83, 85]	8
Communication Channels and Traces	[12, 21, 47, 48, 62, 80]	6
Collaboration, Interaction, and Meetings	[12, 38, 41, 47, 59]	5
Co-Authorship	[22, 68, 71, 80]	4
Task Dependencies, Attributes, and Correlation	[48, 68, 74]	3
People Attributes	[22, 79]	2
Process Attributes	[19, 41]	2
Interpersonal Relationships	[53]	1
Programmer Coupling	[37]	1
Legacy Inheritance	[41]	1

Likewise, different forms have been suggested in literature to illustrate the technical structure in software development, see Table III. So, the technical structures have been described based on the identification of software items and their relationships captured in terms of software architecture, source structure, interfaces, code dependencies, item properties, co-editing, and modification request dependencies.

TABLE III. FORMS OF TECHNICAL STRUCTURES IDENTIFIED

Forms of Technical Structures	References	# of studies
Items	[1, 19, 45, 50, 52, 58, 61, 62, 68, 71, 80, 85]	12
Software Architecture	[21, 39, 52, 79, 81]	5
Source Structure	[19, 22, 57, 62, 68]	5
Modification Request Dependencies	[46, 49, 57, 61]	4
Code Dependencies and Software Coupling	[12, 37, 38, 48]	4
Item Properties and Collections	[22, 41, 73]	3
Interfaces	[10, 18, 59]	3
Footprints	[12, 62]	2
Traceability	[41]	1
File Ownership	[70]	1

Table IV describes the socio-technical traces used in the analyzed studies, in particular, processes, artifacts, and effects of socio-technical interaction.

TABLE IV. FORMS OF SOCIO-TECHNICAL TRACES IDENTIFIED

Socio-Technical Traces	References	# of studies
Responsibility and Task Assignment	[1, 12, 21, 46, 47, 52, 56, 58, 61]	9
File Ownership	[36, 69, 70, 71, 73, 74, 85]	7
Authorship and Code Commits	[22, 46, 48, 62, 68]	5
Work Item Contribution	[10, 50, 58]	3
API Use or Implementation	[38, 39]	2
Awareness Network	[39]	1
Coordination Matrix	[59]	1
Developer Spread	[68]	1
Code Dependencies	[85]	1

Analysis of the studies citing Conway's original proposition that we have included in the review suggests that Conway's law inspired research work can be classified into several directions based on the exploratory purpose of the authors:

- Understanding and managing development organization [18, 22, 62, 68, 69, 74];
- Coordinating development work, including cross-site collaborations [39, 45, 52, 57, 59];
- Improving software engineering [19], in particular, development efficiency [12, 56];
- Minimizing communication and coordination problems and overhead [1, 39, 52, 58, 47];
- Managing quality [10, 85].

From the 76 papers we analyzed, 2 papers assumed that Conway's law is valid, 13 explored the law, the majority discussed it (44), 2 only mentioned it, and 15 reported evidence for or against the validity of Conway's law. Overall, more evidence was reported for Conway's law, than against the law. The way the social and the technical structures are interpreted and how congruence is measured affected the amount and direction of evidence. In order to give an overview of existing evidence we present in the following some of the cases reporting quantitative or qualitative evidence.

Quantitative evidence gathered shows that *socio-technical congruence affects development efficiency* in two cases and that the *modular structure of the architecture is highly consistent with the communication structure* in one case.

- A case study concerned 114 developers grouped into eight development teams distributed across three development locations during 39 months of development [12]. When the developers' coordination patterns were congruent with coordination needs, the resolution time of modification requests was reduced by 32%. Also, geographical and communication congruence affected resolution time positively.
- A case study concerned 30 developers distributed across two development locations during 30 months of development [40]. The productivity of developers was higher when they worked on tasks assigned to few people and that required work in few modules than when they worked on tasks with the opposite characteristics.
- A case study concerned the third instance of a student project at Siemens that simulated industrial software development among distributed sites [59]. The project employed an architecture that enabled an efficient communication structure. The solution's design structure had the potential to guide task assignment and team coordination.

There is also quantitative evidence that shows that *socio-technical congruence affects the success of some types of builds positively and other types negatively*.

- The case concerned 151 developers distributed across 7 development locations during 12 months of development [42]. For continuous builds an increase in congruence correlated to build success probability. However, for integration builds an increase in congruence decreased build success.

Qualitative evidence shows that the *software architecture affects a programmer's ability to work in isolation and enables collaboration.*

- A multi-case study concerned a case of up to seven developers during 27 months of development and a case of six to 14 developers during 21 months of development [37]. The project with unmanaged coupling had experienced less programmer productivity than the project with partial coupling. The study concluded that unmanaged coupling affects

a programmer's ability to work in isolation, hence the velocity of the programmer.
- A multi-case study concerned a case of 34 staff members divided into two groups, the developers and the verification and validation staff, and a case of 57 staff divided into five groups, each developing a different part of the solution [39]. Developers displayed their knowledge about the software architecture when they followed code dependencies to identify the teams to coordinate their work with.

Qualitative evidence was inconclusive regarding *how socio-technical congruence affected software integration.*

- A multi-case study concerned ABB Robotics, Ericsson System Management, Ericsson Core Network Development, TietoEnator Telecom and Media, Volvo Construction Equipment, Volvo Car Corporation, and Bombardier Transportation [41]. A majority of the interviewees stated that their company's organization often mirrored the system architecture and vice versa. Such socio-technical congruence was important to ease software integration and validation and to define interfaces between organizational units.
- A case study concerned 57 developers divided into five teams, each developing a part of the solution. APIs were used as contracts between organizations, hence reified the organizational structures that defined the team boundaries [38]. The software development project relied on APIs both for technical design and social coordination. APIs established a common language, but led to information overload, instability, integration problems, and lack of awareness.

As described above, evidence has been reported for and against the validity of socio-technical congruence. In addition, there are several studies mentioning, assuming, discussing, and exploring Conway's law. But, the papers analyzed investigate only if congruence exists and for example how strong the congruence is, e.g. [12]. But, we argue that socio-technical congruence evolves over time with respect to social or technical changes. This has implications for companies and needs to be investigated further. Therefore, we propose the "rubber band effect", which is introduced and discussed together with its implications below.

IV. Implications and Idea

A. Implications for Globalizing Software Companies

Assuming validity of socio-technical congruence implies that changing organizational structure only or changing software architecture only is problematic (if not impossible). This has an important implication on software companies restructuring the development organization as a consequence of sourcing decisions. Multi-national companies are nowadays confronted with a variety of choices of developing their software products here, there or everywhere. Our industry partners are facing the following situation. Attempts to decrease development cost and to access development resources often lead to replacing the current development organization with a new one, or distributing the development organization across multiple locations. These changes, however, are found to impact efficiency and quality of the development [5]. Thus, investigating socio-technical congruence in software development projects could assist companies in distributed development and sourcing decisions. In addition, assuming that offshoring is initiated with cost savings in mind, it is fair to assume that companies might be unwilling to invest into preparations with regards to socio-technical congruence. In fact, [3] claims that the principal error in offshoring has been associated with an underestimation of the importance of preparation. This means that even if software development evolves compliant with Conway's law before sourcing, changes in the development organization lead to a new organization working on an old architecture, which is not necessarily following socio-technical congruence.

Also, architectural changes need to be considered. Software product development with a long lifecycle tends to result in the growing entropy of the products and costly maintenance, as well as undergo other radical changes, such as merging with other related products. Attempts to increase software evolvability often require finding ways to change the software architecture. Assuming validity of socio-technical congruence implies that changing the software architecture will have an impact on the organizational structure and it leads to the question how this change will look like and how to proactively adapt to these changes. In order to investigate this further, we propose the "rubber band effect" and a research study as described below.

B. Rubber Band Effect

Socio-technical congruence is said to be a natural consequence of the communication and coordination needs of the people developing the software. Such congruence affects the organization and activities performed in that organization on the one hand and the characteristics of the software artifacts on the other hand. In the light of software evolution it has been argued that the software teams and the system architecture are the two important foundations for successful software evolution, which may aid or hinder changes during evolution [88]. The changes however are inevitable. Changes often involve making trade-offs between short-term development success and evolvability of the developed software. This means that the protection of the congruence is not always proactively addressed.

The problems associated with non-congruence reported in the research literature include poor efficiency, poor quality, coordination overhead, etc. We therefore conjecture that changes in social structures evoke technical changes, and vice versa, by this following a "rubber band effect", which helps to restore the socio-technical congruence. Thus, we define the rubber band effect as: "*changes in the social structure of an organization will*

inevitably produce changes in the technical structure of the organization, and vice versa, in order to restore the socio-technical congruence". These changes may occur by chance or mistake and lead to perhaps serious problems. Consequently, there are a couple of questions that need to be answered in connection with this idea. Is it, for example, possible that a change of either the social or the technical structure is so abrupt and strong that the rubber band will break? And where would this threshold lay? Another important question is how much time is needed for either the social or the technical structure to adapt to the change in the other structure so that the socio-technical congruence is restored again? Consequently, understanding the interdependences between the organizational and architectural changes could help to mitigate the complexities associated with non-congruent socio-technical structures and proactively address the development needs. It is important to distinguish proactive and reactive changes, what are the causes and the effects, and help organizations in finding the most efficient ways of implementing the changes.

C. *Proposed Research Design*

Retrospective analysis may help in understanding the nature of the changes and time estimates for reactive changes to take place. The proposed replication of Conway's law-related studies in particular and socio-technical congruence in general shall target at testing the "rubber band effect" by analyzing the co-evolution of a complex development organization and a complex software solution. In particular, we suggest seeking answers to the following research questions:

- RQ 1.: Does socio-technical congruence exist?
- RQ 1.1: Is socio-technical congruence an equilibrium state (does a "rubber band effect" exist)?
- RQ 1.2: Are there changes that irreversibly destroy socio-technical congruence?
- RQ 2.: How does socio-technical congruence relate to development productivity?

To be able to answer these questions, we propose a partial replication of the study by Cataldo et al. [12], which was chosen because of the focus on development efficiency in a globally distributed setting by investigating the impact of socio-technical congruence on development productivity. The social structures used in the study are: team membership, geographical location, modification request co-commenting, and IRC communication. The technical structures are: files edited for a modification request and references between files. The social-technical traces are: file authorship. Consequently, the socio-technical congruence is measured by the alignment between coordination requirements extracted from technical dependencies among tasks, and the actual coordination activities performed by the engineers [12]. These activities can take different forms, meaning the social structure could be measured using different forms. Thus, the authors [12] have used four different ways to measure the actual coordination activities and based on this the authors computed four different socio-technical congruence measures (structural congruence, geographical congruence, modification request communication congruence, and IRC communication congruence).

We plan to conduct only a partial replication because we use similar forms of social and technical structures as [12], but not exactly the same. In particular, we do not take the communication into account when investigating the social structure. Thus, we plan to compute only the structural and the geographical congruence. Besides, we are focusing on the change of social and technical structure and the impact of these changes on development productivity (quantitative) and other product development values (qualitative), e.g. innovation. The unit of analysis is a software project in the software evolution stage, which has gone through several organizational and/or architectural changes. The investigation focuses on qualitative and quantitative analysis of each change in conjunction with consequent related changes. Whereas [12] focuses on congruence measures from a global perspective not related to changes. Consequently, we are planning to collect the following data:

- Social structure: people (attributes: e.g. roles, experience...), team membership and geographical collocation;
- Technical structure: files edited for a modification request, and references between files;
- Evolution: activation and resolution of modification requests, and timing of changes to technical elements, releases;
- Socio-technical traces: authorship from the source code, and commenting of modification requests in task management system.

By conducting the proposed study, we would like to identify visible changes in technical and social structures during the evolution, such as a transfer of software development activities from one team to another, merges of several sub-systems or major refactoring within a product, and whether these provoke the rubber band effect. In particular, if socio-technical congruence exists, then we would like to identify whether changes in one of the structures at one point of time evoke subsequent changes in the other, to help restoring the congruence. The restoration of the congruence might require time, and thus a longitudinal study for comparison analysis is needed. We are also interested in measuring how long such changes take. Additionally, we assume that there might be cases, in which the socio-technical congruence is irreversibly destroyed due to the changes. We are interested to explore the reasons for this, if such cases are identified. This would ultimately require studying a long-term product development with multiple social and technical changes, or multiple product development cases.

Finally, we expect that the proposed study shall enable evidence-based recommendations for how to proactively deal with organizational and architectural changes and help software organizations avoid making mistakes that lead to lengthy reactive recoveries.

V. LIMITATIONS

The presented literature review and the proposed research design have several limitations. First, the conducted literature review is not a Systematic Literature Review as proposed in [14]. We do not use a review protocol and we have not covered all relevant scientific databases. Thus, it is likely that we have missed relevant papers, especially those not citing Conway's paper but conducting research on socio-technical congruence. However, the chosen "snowballing" approach to conduct a literature review has proven its effectiveness [89]. Some other steps of the systematic literature review method [14] have been omitted, e.g. a rigorous inclusion/exclusion process, a formal quality assessment process or a validation of review steps. In addition, we have not reviewed all the papers found by our search. Instead we classified them and chose several clusters to look at based on the theme. Thus, we, as researchers, might have introduced a bias in the outcome of the review. Nevertheless, the goal of our review was not to conduct a Systematic Literature Review, but to get an overview of the existing research on Conway's law and the empirical evidence for or against the validity of Conway's law and by conducting the review we have achieved this goal. In addition, our literature review can serve as a starting point for a future mapping study or a systematic literature review.

Second, regarding the proposed research design, it is possible that we will not get all the data we plan to collect. If this happens, we will have to adapt the data collection to what is possible to collect. In order to minimize this threat we have already started discussing the proposed study in a steering group meeting including researchers and industrial members.

VI. CONCLUSIONS

Literature shows that there are significant differences in interpreting and applying Conway's law. Many different forms for social and technical structures as well as socio-technical traces have been suggested. The social structures have been described based on how organizations were structured, how work was structured (e.g. on a task level), how people were "structured" (e.g. collocated, collaborated, and communicated), and the co-authorship resulting from the development. The technical structures have been described based on the identification of software items and their relationships captured in different terms of e.g. software architecture, code dependencies, item properties, co-editing, and modification request dependencies. Finally, the socio-technical traces used in the reviewed literature to explain how the social structures were mapped into the technical structures and vice-versa are effects of socio-technical interaction (e.g. awareness network), processes and tasks (e.g. responsibility and task assignment), and people traces on artifacts (e.g. file ownership, authorship and code commits).

Similarly to the social and technical structures the purposes of the application of Conway's law differ. Thus, we have identified several directions based on the exploratory purpose of the authors. Namely: understanding and managing development organization, coordinating development work, including cross-site collaborations, improving software engineering, in particular, development efficiency, minimizing communication and coordination problems and overhead, and managing quality. However, the research work citing Conway's law that we analyzed can be summarized under understanding, managing, and improving software engineering.

Most of the papers we analyzed primarily discussed Conway's law. Nevertheless, evidence for or against the validity of Conway's law was reported. In general, more evidence was reported to support Conway's law, than against it. It seems that this depends on the way the social and the technical structures are interpreted and how the structures are mapped by the socio-technical traces. This is a very interesting finding and offers direction for future research, including, for example, a systematic literature review investigating the influence of the social structures, the technical structures and the socio-technical traces mapping them.

Overall, the evidence for socio-technical congruence encourages the use of software architecture to facilitate organizational growth, split or merger. At the same time it also encourages the use of organizational structure to facilitate software refactoring needed to increase evolvability and ability to innovate with the software. The proposed concept emphasizes the necessity to combine organizational and technical changes. Continuous focus on socio-technical congruence can help companies reduce or prevent the negative impact of changes that break the congruence. The suggested replication study will be based on the research design by Cataldo et al. [12], although adapted as explained and based on information available at the company where the case study will be conducted. Investigations of the information available and hence the final research design is ongoing work, and hence we cannot report exactly how the research design will differ from the one by Cataldo et al. [12].

The proposed research is intended to increase the understanding of whether and how architecture affects the organization and vice-versa to improve decision-making in situations where the organization or the software architecture is changed. The "rubber band effect" might be effective for facilitating or actively blocking change. So for example, is it possible to change the team structure to facilitate architectural change or to change the architecture to facilitate change of the development organization? We therefore encourage other researchers to follow our idea and collect evidence for or against its existence.

ACKNOWLEDGEMENT

This research is supported by BESQ+ project (2010/0311) from the Knowledge Foundation in Sweden.

VII. REFERENCES

[1] E. Ó Conchuir, H. Holmström, P. J. Ågerfalk, and B. Fitzgerald, "Exploring the Assumed Benefits of Global Software Development," in *Proceedings of the First International Conference on Global Software Engineering*, pp. 159–168, 2006.

[2] J. D. Herbsleb and D. Moitra, "Global Software Development," *IEEE Software*, vol. 18, pp. 16-20, 2001.

[3] E. Carmel and P. Tjia, *Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce*. In: Cambridge University Press, 2005.

[4] D. Damien and D. Moitra, "Guest Editors' Introduction: Global Software Development: How far have we come?," *IEEE Software*, vol. 23, pp. 17-19, 2006.

[5] R. Jabangwe and D. Smite, "An Exploratory Study of Software Evolution and Quality: Before, During and After a Transfer," in *Proceedings of the 7th IEEE International Conference on Global Software Engineering*, pp. 41-50, 2012.

[6] N. B. Moe, D. Smite, and G. K. Hanssen, "From Offshore Outsourcing to Offshore Insourcing: Three Stories," in *Proceedings of the IEEE 7th International Conference on Global Software Engineering*, pp. 1-10, 2012.

[7] D. Smite and C. Wohlin, "Lessons Learned from Transferring Software Products to India," *Journal of Software: Evolution and Process*, vol. 24, no. 6, pp. 605-623, 2012.

[8] D. Smite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical Evidence in Global Software Engineering," *Empirical Software Engineering*, vol. 15, pp. 91-118, 2010.

[9] H. Holmstrom, E. Ó Conchuir, P. J. Agerfalk, and B. Fitzgerald, "Global Software Development Challanges: A Case Study on Temporal, Geographical. and Socio-Cultural Distance," in *Proccedings of the IIEEE nternational Conference of Global Software Engineering*, pp. 3-11, 2006.

[10] J. D. Herbsleb and R. Grinter, "Architectures, Coordination, and Distance: Conway's Law and Beyond," *IEEE Software*, vol. 16, pp. 63-70, 1999.

[11] M. Conway, "How do Committees Invent?," *Datamation*, vol. 14, no. 4, pp. 28-31, 1968.

[12] M. Cataldo, J. Herbsleb, and K. Carley, "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 Dymposium on Empirical Software Engineering and Measurement*, pp. 2-11, 2008.

[13] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly,* vol. 26, no. 2, pp. 13-23, 2002.

[14] B. A. Kitchenham, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Software Engineering Group, School of Computer Science and Mathematics, Keele University, Keele, Staffs, ST5 5BG, UK and Department of Computer Science, University of Durham, UK, 2007.

[15] A. Atlas, "Accidental adoption: The story of scrum at amazon.com," in *AGILE'09*, pp. 135-140, 2009.

[16] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does distributed development affect software quality?: an empirical case study of Windows Vista," *Communications of the ACM*, vol. 52, no. 8, pp. 85-93, 2009.

[17] B. Boehm, "The future of software processes," in *Unifying the Software Process Spectrum*, M. Li, B. Boehm, and L. J. Osterweil, Eds. Berlin Heidelberg: Springer, pp. 10-24, 2006.

[18] C. R. B. de Souza, D. Redmiles, L.-T. Cheng, D. Millen, and J. Patterson, "Sometimes You Need to See Through Walls — A Field Study of Application Programming Interfaces," *Nature*, vol. 6, no. 3, pp. 63-71, 2004.

[19] J. D. Herbsleb, "Beyond computer science," in *Proceedings. 27th International Conference on Software Engineering*, pp. 23-27, 2005.

[20] P. Wallin, S. Johnsson, and J. Axelsson, "Issues related to development of E/E product line architectures in heavy vehicles", in *Proceedings of the 42nd Hawaii International Conference on System Sciences*, pp. 1-10, 2009.

[21] C. Amrit and J. I. Van Hillegersberg, "Detecting coordination problems in collaborative software development environments," *Information Systems Management*, vol. 25, no. 1, pp. 57-70, 2008.

[22] G. Robles, J. Gonzalezbarahona, and J. Merelo, "Beyond Source Code: The Importance of other Artifacts in Software Development (a Case Study)," *Journal of Systems and Software*, vol. 79, no. 9, pp. 1233-1248, 2006.

[23] S. Giesecke, "Taxonomy of architectural style usage," in *Proceedings of the Conference on Pattern Languages of Programs*, pp. 32-41, 2006.

[24] D. Balasubramaniam, R. Morrison, R. M. Greenwood, and B. In Software Architecture, 2007. WICSA'07. Warboys, "Flexible software development: From software architecture to process," in *Proceedings of the Working IEEE/IFIP Conference on Software Architecture*, p. 14, 2007.

[25] M. Balint, T. Girba, and R. Marinescu, "How developers copy," in *Proceedings of the 14th IEEE International Conference on Program Comprehension*, pp. 56-68, 2006.

[26] M. Bass, V. Mikulovic, L. Bass, H. James, and C. Marcelo, "Architectural misalignment: An experience report," in *Proceddings of the Working IEEE/IFIP Conference on Software Architecture*, p. 17, 2007.

[27] A. Beckhaus, L. M. Karg, and D. Neumann, "The Impact of Collaboration Network Structure on Issue Tracking's Process Efficiency at a Large Business Software Vendor," in *Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS)*, pp. 1-10, 2010.

[28] J. Bosch and P. Bosch-Sijtsema, "From integration to composition: On the impact of software product lines, global development and ecosystems," *Journal of Systems and Software*, vol. 83, no. 1, pp. 67-76, 2010.

[29] O. Greevy, T. Girba, and S. Ducasse, "How developers develop features," in *Proceedings of the 11th European Conference on Software Maintenance and Reengineering*, pp. 265-274, 2007.

[30] R. E. Grinter, "Recomposition: Coordinating a web of software dependencies," *Computer Supported Cooperative Work (CSCW)*, vol. 12, no. 3, pp. 297-327, 2003.

[31] S. Marczak and D. Damian, "How interaction between roles shapes the communication structure in requirements-driven collaboration," in *Proceedings of the 19th IEEE International Requirements Engineering Conference (RE)*, pp. 47-56, 2011.

[32] P. Wallin, S. Cedergren, S. Larsson, and J. Axelsson, "Limiting practices in developing and managing software-intensive systems: A comparative study," in *Proceedings of Technology Management for Global Economic Growth*, pp. 1-9, 2010.

[33] M. Wermelinger, Y. Yu, and M. Strohmaier, "Using formal concept analysis to construct and visualise hierarchies of socio-technical relations," in *Proccedings of the 31st International Conference on Software Engineering-Companion Volume,* ICSE-Companion, pp. 327-330, 2009.

[34] S. Wong, Y. Cai, G. Valetto, G. Simeonov, and K. Sethi, "Design rule hierarchies and parallelism in software development tasks," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, pp. 197-208, 2009.

[35] Z. Lixin, "A Project human resource allocation method based on software architecture and social network," in *Proceedings of the I4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-6, 2008.

[36] F. Beck and S. Diehl, "On the congruence of modularity and code coupling," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 354-364, 2011.

[37] J. W. Cain and R. J. McCrindle, "An investigation into the effects of code coupling on team dynamics and productivity," in *Proceedings 26th Annual International Computer Software and Applications*, pp. 907-913, 2002.

[38] C. R. B de Souza and D. F. Redmiles, "On the roles of APIs in the coordination of collaborative software development," *Computer Supported Cooperative Work (CSCW)*, vol. 18, no. 5-6, pp. 445-475, 2009.

[39] C. R. B. de Souza and D. F. Redmiles, "The Awareness Network, To Whom Should I Display My Actions? And, Whose Actions Should I Monitor?," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 325-340, 2011.

[40] J. D. Herbsleb and A. Mockus, "Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering," in *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, pp. 138-147, 2003.

[41] G. Mustapic, A. Wall, C. Norström, I. Crnkovic, K. Sandstrom, J. Froberg, and J. Andersson, "Real world influences on software architecture - interviews with industrial system experts," in *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture*, pp. 101-111, 2004.

[42] I. Kwan, A. Schroter, and D., Damian, "Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 307-324, 2011.

[43] 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 *Proceedings of the Fourth International Workshop on Mining Software Repositories, ICSE Workshops MSR'07*, p. 25, 2007.

[44] A. Aneesh, "Global Labor: Algocratic Modes of Organization," *Sociological Theory*, vol. 27, no. 4, pp. 347-370, 2009.

[45] S. Beecham, J. Noll, I. Richardson, and N. Ali, "Crafting a Global Teaming Model for Architectural Knowledge," in *Proceedings of the 5th IEEE International Conference on Global Software Engineering*, pp. 55-63.

[46] M. Cataldo and J. D. Herbsleb, "Communication networks in geographically distributed software development," in *Proceedings of the ACM conference on Computer supported cooperative work*, pp. 579-588, 2008.

[47] M. Cataldo, M. Bass, J. D. Herbsleb, and L. Bass, "On Coordination Mechanisms in Global Software Development," in *Proceedings of the International Conference on Global Software Engineering, ICGSE*, pp. 71-80, 2007.

[48] S. B. Fonseca, C. R. B. de Souza, and D. F. Redmiles, "Exploring the Relationship between Dependencies and Coordination to Support Global Software Development Projects," in *Proceedings of the IEEE international conference on Global Software Engineering, ICGSE*, pp. 243-244, 2006.

[49] L. D. Panjer, D. Damian, and M. A. Storey, "Cooperation and coordination concerns in a distributed software development project," in *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, pp. 77-80, 2008.

[50] T. Nguyen, T. Wolf, and D. Damian, "Global software development and delay: Does distance still matter?," in *Proceedings of the IEEE International Conference on Global Software Engineering, ICGSE*, pp. 45-54, 2008.

[51] P. Ovaska, M. Rossi, and P. Marttiin, "Architecture as a coordination tool in multi-site software development. Software Process: Improvement and Practice," vol. 8, no. 4, pp. 233-247, 2003.

[52] M. Cataldo and J. D. Herbsleb, "End-to-end features as meta-entities for enabling coordination in geographically distributed software development," in *Proceedings of the ICSE Workshop on Software Development Governance*, pp. 21-26, 2009.

[53] I. Heitlager, R. Helms, and S. Brinkkemper, "Evolving relationship structures in multi-sourcing arrangements: the case of mission critical outsourcing," in *Global Sourcing of Information Technology and Business Processes*, I. Oshri and J Kotlarsky, Eds.: Springer Berlin Heidelberg, pp. 185-201, 2010.

[54] N. Ramasubbu and R. K. Balan, "Towards governance schemes for distributed software development projects," in *Proceedings of the 1st international workshop on Software development governance*, pp. 11-14, 2008.

[55] F. Salger, "On the use of handover checkpoints to manage the global software development process," in *Proceedings of the On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, pp. 267-276, 2009.

[56] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 481-494, June 2003.

[57] A. Mockus and D. M. Weiss, "Globalization by chunking: a quantitative approach," *IEEE Software*, vol. 18, no. 2, pp. 30-37, 2001.

[58] F. Salger, "Software Architecture Evaluation in Global Software Development Projects," in *Proceedings of the On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, pp. 391-400, 2009.

[59] A. Avritzer, D. Paulish, Y. Cai, and K. Sethi, "Coordination implications of software architecture in a global software development project," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1881-1895, 2010.

[60] D. Mishra and A. Mishra, "A review of non-technical issues in global software development," *International Journal of Computer Applications in Technology*, vol. 40, no. 3, pp. 216-224, 2011.

[61] M. Cataldo and J. D. Herbsleb, "Communication patterns in geographically distributed software development and engineers' contributions to the development effort," in *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, pp. 25-28, 2008.

[62] C. Del Rosso, "Comprehend and analyze knowledge networks to improve software evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 21, no. 3, pp. 189-215, 2009.

[63] U. Eklund and C. M. Olsson, "A case study of the Architecture Business Cycle for an in-vehicle software architecture," in *Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, WICSA/ECSA*, pp. 91-100, 2009.

[64] O. Nierstrasz, M. Denker, T. Gîrba, A. Lienhard, and D. Röthlisberger, "Change-enabled software systems," in *Software-Intensive Systems and New Computing Paradigms*, M. Wirsing, J. P. Banâtre, M. Hölzl, and A. Rauschmayer, Eds.: Springer Berlin Heidelberg, pp. 64-79, 2008.

[65] Á. Szőke, "A feature partitioning method for distributed agile release planning," in *Agile Processes in Software Engineering and Extreme Programming*, A. Sillitti, O. Hazzan, E. Bache, and X. Albaladejo, Eds.: Springer Berlin Heidelberg, pp. 27-42, 2011.

[66] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 24-35, 2008.

[67] C. R. B De Souza, J. Froehlich, and P. Dourish, "Seeking the source: software source code as a social and technical artifact," in *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pp. 197-206, 2005.

[68] D. Ganesan, D. Muthig, J. Knodel, and K. Yoshimura, "Discovering Organizational Aspects from the Source Code History Log during the Product Line Planning Phase--A Case Study," in *Proceeding of the 13th Working Conference on Reverse Engineering*, pp. 211-220, 2006.

[69] T. Girba, A. Kuhn, M. Seeberger, and S. Ducasse, "How Developers Drive Software Evolution," in *Proceedings of the Eighth International Workshop on Principles of Software Evolution, IWPSE'05*, pp. 113-122, 2005.

[70] J. Han, C. Wu, and B. Lee, "Extracting development organization from open source software," in *Proceedings of the 2009 16th Asia-Pacific Software Engineering Conference, APSEC '09*, Washington, DC, USA, pp. 441–448, 2009.

[71] H.-Y. Huang, Q. Le, and J. H. Panchal, "Analysis of the structure and evolution of an open-source community," in *Journal of Computing and Information Science in Engineering*, vol. 11, no. 3, 2011.

[72] A. Jermakovics, A. Sillitti, and G. Succi, "Mining and visualizing developer networks from version control systems," in *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '11*, New York, NY, USA, pp. 24–31, 2011.

[73] M. Lungu, M. Lanza, T. Girba, and R. Heeck, "Reverse engineering super-repositories," in *Proceedings of the 14th Working Conference on Reverse Engineering, WCRE*, pp. 120–129, 2007.

[74] J. Tsay, H. K. Wright, and D. E. Perry, "Experiences mining open source release histories," in *Proceedings of the 2nd workshop on Software engineering for sensor network applications, SESENA*, p. 208, 2011.

[75] L. Bass, P. Clements, R. Kazman, and M. Klein, "Evaluating the software architecture competence of organizations," in *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture, WICSA*, pp. 249-252, 2008.

[76] M. Broy, G. Reichart, and L. Rothhardt, "Architectures of software-based functions in vehicles," *Informatik Spektrum* , vol. 34, no. 1, pp. 42-59, 2011.

[77] F. Buschmann, "On architecture styles and paradigms," *IEEE Software*, vol. 27, no. 5, pp. 92-94, 2010.

[78] K. Rossi, M. Purao, S. Smolander, "Software architectures: Blueprint, Literature, Language or Decision?," *European Journal of Information Systems*, vol. 17, no. 6, pp. 575-588.

[79] A. Cockburn, "The interaction of social issues and software architecture," *Communications of the ACM*, vol. 39, no. 10, pp. 40-46, 1996.

[80] A. Bacchelli, "Exploring, exposing, and exploiting emails to include human factors in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering, ICSE*, pp. 1074-1077, 2011.

[81] I. T. Bowman and R. C. Holt, "Software architecture recovery using Conway's law," in *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research*, p. 6, 1998.

[82] S. Ducasse and D. Pollet, "Software architecture reconstruction: A process-oriented taxonomy," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 573-591, 2009.

[83] H. Hadaytullah, O. Räihä, and K. Koskimies, "Genetic approach to software architecture synthesis with work allocation scheme," in *Proceedings of the 17th Asia Pacific Software Engineering Conference (APSEC)*, pp. 70-79, 2010.

[84] J. Herbsleb, "Talking about concerns," in *Proceedings of the tenth international conference on Aspect-oriented software development* , pp. 281-282, 2011.

[85] N. Nagappan, B. Murphy, and V. R. Basili, "The Influence of Organizational Structure on Software Quality: An Empirical Case Study," in *International Conference on Software Engineering, ICSE*, pp. 521-530, 2008.

[86] N. Bettenburg, "Mining development repositories to study the impact of collaboration on software systems," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 376-379, 2011.

[87] C. Del Rosso, "Software performance tuning of software product family architectures: Two case studies in the real-time embedded systems domain," *Journal of Systems and Software*, vol. 81, no. 1, pp. 1-19, 2008.

[88] V. T. Rajlich and K. H. Bennett, "A staged model for the software life cycle," *IEEE Computer*, vol. 33, no. 7, pp. 66-71, July 2000.

[89] T. Greenhalgh and K. R. Peacock, "Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources," *BMJ - Information in Practice*, vol. 331, pp. 1064-1065, 2005.

[90] L. Colfer and C. Y. Baldwin, "Mirroring Hypothesis: Theory Evidence and Exceptions," working paper, Harvard Business school, 2010.