Dimensions of Consistency in GSD: Social Factors, Structures and Interactions

Outi Sievi-Korte[1], Fabian Fagerholm[2], Kari Systä[1], and Tommi Mikkonen[3]

[1] Tampere University, Tampere, Finland
[2] Aalto University, Espoo, Finland
[3] University of Helsinki, Helsinki, Finland
outi.sievi-korte@tuni.fi, fabian.fagerholm@aalto.fi,
tommi.mikkonen@helsinki.fi, kari.systa@tuni.fi

Abstract. Global software development (GSD) implies a distributed development organization, where coordination is needed to efficiently achieve development objectives. So far, socio-technical congruence has examined coordination needs and activities through software code dependencies. However, GSD requires coordination beyond software artifacts. In this paper, we present an interview-based study of software practitioners from companies engaged in GSD. The study examines how different dimensions of interactions are interrelated, and how they affect software development. Our study suggests that, in addition to the relationship between organizational and technical system structure, GSD performance is affected by consistency in communication, operational procedures, and social structures. These can only partially be impacted through formal procedures, and we suggest that distributing coordination work by empowering developers could lead to increased performance.

Keywords: Global software development · Human factors · Socio-technical system · Coordination · Communication

1 Introduction

Global software development (GSD) can be defined as "software work undertaken at geographically separated locations across national boundaries in a coordinated fashion involving real time (synchronous) and asynchronous interaction" [25]. GSD is said to have several benefits in terms of productivity, cost savings, skill pool access, and customer proximity [1], but it accentuates the need to coordinate work tasks among those involved. With group members being separated geographically, temporally, and culturally, coordination becomes more difficult.

The concept of socio-technical congruence (STC) captures the notion of a relationship between coordination needs and actual coordination activities: if there is a match between the two, congruence is high (and vice versa) [8]. STC can be measured through a family of techniques that are based on extracting task dependencies from source code repositories [26]. Files that are commonly changed together are assumed to have a technical dependency [8]. It is thus possible to

calculate to what extent current coordination activities, as played out through the communication and collaboration tools used by software developers, match the dependencies indicated by previous source code change sets.

Several researchers have identified factors that can hamper coordination in GSD, and ways in which sub-optimal coordination can surface as a worsening in STC metrics [26]. However, STC does not draw a complete picture of coordination challenges in GSD. STC measurements use after-the-fact data that describe some of the most detailed-level tasks operating closest to or directly on the source code. They do not consider the planning and deliberation that precedes these tasks, the coordination judgments that are made while the tasks are carried out, or the human relationships that form the basis of coordination. A study on architectural design in GSD [29] implies that many issues voiced by practitioners concern social interactions and the organization of communication.

Increasing STC requires additional effort in the form of specific coordination activities, which run the risk of decreasing developer productivity [26]. STC techniques do not describe how to avoid such risks. What is often overlooked in GSD is its effect on developers' behavior and habits, and a consideration of the cognitive, affective, motivational, and social processes involved. From that perspective, coordination in GSD is an active human process unfolding in a socio-technical system. Its results can be partially observed by measuring STC – but to understand the process itself, we must look beyond artifact repositories.

In this paper, we aim to uncover more of the social factors, structures, and interactions that are at play when the coordination process unfolds in GSD. We utilize the data collected by Sievi-Korte et al. [29] to find instances of social aspects of coordination in GSD. We describe factors that influence the quality of the coordination process, describe threats to that process, and discuss means by which those threats might be mitigated. Through this knowledge, we aim to assist organizations that wish to utilize GSD to gain more of the potential benefits while avoiding adverse effects on both internal and external stakeholders.

2 Background

When practicing GSD, temporal, geographical, and socio-cultural distance has a direct and immediate effect on developers' socio-technical environment. For example, time-zone differences make communication asynchronous. Emails and even instant messages receive delayed answers, and phone and video calls are difficult to schedule with no overlapping office hours between sites. Product development may be slowed, rather than allowing effective utilization of time zone differences [7]. Increased coordination needs present another challenge, and can arise due to delays, integration issues, and mismatches in required skills. Inspired by Conway's Law [9], researchers have examined STC, formal communication structures, and tool support as vehicles for improved coordination (e.g., [2]). To successfully operate in a GSD setting, developers should thus be able to consider how their own actions influence both the social and technical sides of the software development activity.

Difficulties in distributed software projects can be forecasted by various theories and empirical findings in the organizational, behavioral, and social sciences. For example, when groups bring their behaviors under normative control, those norms begin to regulate team members' behavior [11]. Norms usually develop informally from corporate and national cultures, but may be set collectively or by leaders through rewards and sanctions. Unless their norms are compatible, different groups may expect different behaviors, causing misinterpretation of actions and intentions, and resulting in inefficiency and enmity [6]. This provides an explanation of a social mechanism influencing software projects: norm alignment requires communication, and hinges on the complexity, richness, and speed of the real communication networks connecting developers. In other words, addressing the problem requires entering the social world of developers and agreeing on behavioral changes. It then becomes necessary to trust developers' social skills and adapting formal procedures and structures to support them.

Relying on developers' social skills requires that the management is more informed regarding the mechanisms of social behavior among developers. An understanding of the social aspects of software development and promoting social awareness and constructive social behavior among developers (see [21]) can begin to address the deeper issues involved. Those issues are present at the level of individuals, but they also become built into the processes, working methods, and artifacts involved in software projects – all part of the socio-technical system in which software development happens. Failing to take social behavior into account may lead to accumulating social debt in the organization, which in turn leads to deterioration of the software itself [32].

Through practitioner workshops, Rothman and Hastie [24] found issues related to how the socio-technical environment is set up. For example, enforcing processes does not work if inter-team practices are not considered. Not enabling meetings ends up costing more than arranging meetings between sites as issues accumulate due to lack of trust. Teams struggle with handling inequality and accommodating differences. However, the reasons behind these issues were not studied further. We attempt to address this gap in the present paper.

Solutions to software engineering problems are often sought in technical methods and practices, but many problems in GSD require a different approach. Sievi-Korte et al. [27] used a systematic literature review (SLR) to create a conceptual model of software architecting in GSD. The model encompasses areas such as ways of working, knowledge management, and task allocation – all subject to communication and process challenges that could potentially be mitigated by architecting guidelines and technical solutions. These challenges and potential solutions were further investigated in an empirical study [29]. The empirical study revealed an abundance of issues that stemmed from social interactions between stakeholders and the organization of communication. Mechanisms based on repository measurement or tools were not able to bring adequate solutions. Such social and behavioral issues are further investigated in the present paper.

3 Research Approach

We utilize data from an earlier study by Sievi-Korte et al. [29], that consists of semi-structured interviews with 13 software developers or architects from seven companies engaged in GSD. All participants had several years of experience from globally distributed projects. Five companies had their headquarters in Finland and two in other countries (Europe and North America). The interview questions were based on the results obtained in an SLR [27].

The previous study [29] collected challenges and practices that practitioners faced doing software architecture design in the context of GSD. In this study, we focus only on material that did *not* consider architecting challenges or practices. Instead, we reanalyze the data to examine the social side of software development. An initial extraction of data relevant to this theme gave us a working set of 109 quotes[4]. As Sievi-Korte et al. [29] only concentrated on findings related to architectural design, this paper thus complements their findings.

Research Questions. As mentioned above, current methods based on STC that utilize the structural dependency between technical artifacts and communication seem insufficient to tackle challenges that have been reported in GSD. We should thus expand the picture and assume that more flexible structures are needed in large-scale software development. We should more carefully consider the social mechanisms that underlie the challenges in GSD also from other perspectives than development activities and related artifacts. We pose the following research questions in the context of software product design in GSD:

RQ1: *How does the coordination process in GSD manifest in terms of social factors, structures, and interactions?*
RQ2: *What factors influence the quality of the coordination process?*
RQ3: *How can the threats to the coordination process be mitigated?*

Research Process. The research process for this study is depicted in Fig. 1 and described below. During the process we used thematic analysis [5]. First, in an *initial analysis* of the interviews, conducted jointly by all authors, there appeared to be repeating patterns in the challenges described by the practitioners. As an experiment, we divided individual quotes from the interviews equally between all authors and attempted to extract anti-patterns from them, if possible. Each found anti-pattern was expected to include context containing a problem, (wrong) solution to the problem, and (negative) consequences.

We then conducted a joint workshop to *validate* the found anti-patterns. During the workshop we examined each anti-pattern on completeness and discussed it in relation to the original quote. At this time we could immediately see that 1) not all quotes had resulted in anti-patterns, and 2) in many cases the anti-pattern was incomplete, i.e., either there was not enough context in the quote, or the solution or consequences was unclear. We decided not to pursue

[4] The transcripts were coded in full by the first author. The codes "practice" and "challenge" were predetermined; other codes were freely generated during the coding process. The coding process has been reported elsewhere in detail [28]

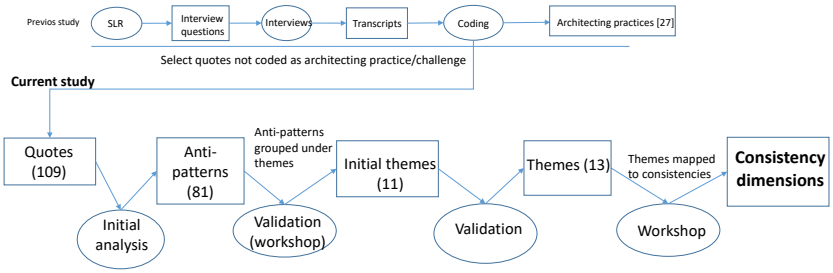


Fig. 1. Research process

with anti-patterns in the sense that they would be presented as a result of this study. However, we did continue using the anti-pattern drafts in our analysis, as they had allowed us to abstract the vast number of often very verbose quotes into a condensed format, easing their further analysis. Reviewing the set of anti-patterns, we could identify a number of recurring *themes* from keywords found in the patterns. Based on the validation workshop notes, one author created a set of 11 initial themes, and coded the anti-patterns according to the themes. Doing so, it appeared that each anti-pattern supported two themes, and thus was coded with a primary and secondary theme.

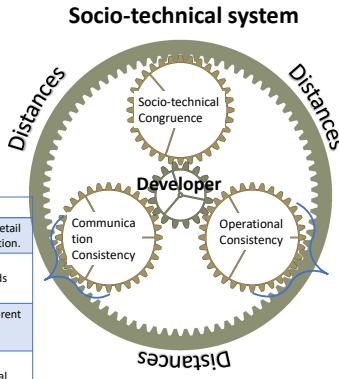
The themes and coding were *validated* by all other authors. Each of them individually and separately attempted to code the anti-patterns using the initial themes. When we cross-checked our coding, we could find a number of conflicts. Upon solving conflicts in the coding, the set of themes was redesigned to contain 13 different themes, which all authors agreed upon. We then conducted a second joint *workshop* to resolve the remaining conflicts. As a way to resolve the conflicts, we revisited the data in a more holistic manner, going back to the original quotes and the context surrounding them. We shortly noticed that while STC could easily be seen as an underlying phenomenon behind many of the themes, they appeared to touch other dimensions as well that were yet undefined. In the end we could see that the intertwined dimensions of individual developer and organization could complement STC. These dimensions are discussed in the following.

4 Results

Our analysis uncovered two sets of themes that we call *consistency dimensions*. Communication between and within teams and sites, in all forms reported by the participants, were grouped as *communication consistency*. The relationships between technical processes, organizational hierarchy and the social interactions between developers were termed *operational consistency*. We also saw how the dimensions of social interactions, communication, and processes were linked to STC. We see these consistency dimensions as independent of, yet interlinked with, STC. Whereas STC is grounded in detailed code artifacts, our consistency

dimensions concern other parts of the socio-technical system where software development happens. They currently lack a numerical operationalization like the STC metrics discussed previously. In this paper, they are instead represented by the themes that specify concrete problems related to each consistency dimension.

Fig. 2 presents the themes forming our model of social interactions in GSD organizations. The themes either describe communication (in)consistency or operational (in)consistency. Our two consistency dimensions are linked to STC both as concerns that individual developers may have to address and as aspects of the overall socio-technical system (represented by the outermost gear with its corresponding distances; see, e.g., [13,1]). At the heart of our model is the developer, who must find means to communicate with others, follow processes, and produce technical artifacts – and engage in social interactions during such activities. In the following, we first describe the themes (highlighted in bold) together with quotes from our interviewees, to illuminate the abstract consistency dimensions, and then discuss the model as a whole.



Theme	Theme description
Working culture	Differences in level of detail required in communication.
Access to knowledge	Realization of communication methods between sites.
Verbal communication	Differences in how different cultures are addressed.
How to read documentation	Matching of intended interpretation and actual interpretation of document.
Assumptions	Assumptions on the status or meaning of items.
Social learning curve	Differences in learning how to work with different cultures.

Theme	Theme description
Organisational hierarchy	Differences in hierarchies (flat vs. layered) between sites.
Processes	Matching of defined processes and daily activitites.
Differences between sites	Differences in equipment, human resources and access to knowledge between sites.
Expectations	Varying expectations on outcomes.
How to write documentation	Different levels of expected details between sites.
Commitment	Commitment to tasks.
Technical learning curve	Differences in learning new technologies due to varying backgrounds.

Fig. 2. Social interactions in a GSD organization.

4.1 Communication consistency

Communication consistency concerns the match between *ways* (technical arrangement of communication), *forms* (presentation of content) and *balance* (amount of interaction) of communication in different parts of the organization and between individuals. Major issues in communication may arise from **differences in working cultures** between sites. Cultural misunderstandings can result in severe difficulties, such as avoidant behavior, as in this example: *"the issue is that in some cultures you don't tell the manager something won't be done,*

you only tell him when you have no choice and that's because it is due now, and it just isn't there. And that can be a big problem." This is not necessarily a factor of any particular culture but rather reflects the lack of a trustful relationship between people from different cultures. One party may be accustomed to very direct communication and have a long history with colleagues, whereas another party is used to a more subtle form or has just joined the development effort. In general, it is known that developers' views on productivity varies [19] and that they communicate with their managers in different ways, so it is not surprising that issues arise in GSD when the communication bandwidth is limited or trust has not yet been built.

While **verbal communication** is understandably challenging between different sites, a bigger issue is that when live meetings are not feasible, developers are forced to settle for **reading documentation**. Misunderstandings easily arise, like *"when somebody writes [something that] is perfectly clear to the person who wrote it and their nearby colleagues [but that] can be interpreted entirely differently at a remote site, and that sometimes comes true in differences of implementation"*. These challenges are linked to operational consistency through procedures for how documentation is written. The process and practices for writing documentation should consider that documentation may be used to substitute verbal communication, and thus it needs to be self-explanatory and unambiguous.

Above all else, problems arise from actual lack of communication and from communication that parties are not able to understand. A big issue for practitioners is **assumptions** about what people at other sites know or how they view the project's status and goals. Team mental models – representations of key elements in the environment shared across team members [20] – are critical for mutual understanding among individuals. In GSD, mental models must be shared across geographically separated teams [3]. The salience of key elements may differ considerably between teams at different sites, and the environment may even be completely different, with varying understanding of common key elements. Convergence in mental models does not occur by itself over time even within the same geographically collocated team. Rather, as team member roles become more differentiated, interaction can decrease and the shared mental models decline [16]. These conditions can cause wildly different mental models across distributed teams, leading to assumptions that stifle communications.

Conflicting views may even result in using incorrect information to question solutions. As noted in our material, *"people quite easily develop their own notion of what's going on and their own image on what we're doing, and then all of the work is filtered through that notion, and then they wonder why we're doing this, why aren't we doing that, and they might not realize at any point that we're talking about two different things."* Apart from assumptions on the technical aspects, information may be interpreted incorrectly because of assumptions arising from cultural differences. Easing the **social learning curve** would help communication between developers from different cultures. A concrete example

from a case company is to arrange multi-cultural parties to celebrate diversity, instead of treating it as an obstacle.

These issues describe the effects of poor *communication consistency*. Communication has already been identified as a part of STC as a structural property of an organization [8] and as a factor that is associated with development speed [13]. We extend the viewpoint from aspects of communication that are measurable on the surface (number of meetings and interactions) to considering ways, forms, and balance of communication, as well as expectations regarding them. These are factors that stem directly from individual developers, and how they are enabled to and actually do communicate with each other.

Our presentation of communication consistency further supports the Media Synchronicity Theory [10], which states that working communication requires working processes both for *conveyance* (i.e., distributing information), and *convergence* (i.e., reaching a common understanding of the information). Communication consistency in ways, forms and balance are required to enable good conveyance and convergence. The issues we have found are often due to insufficient means (or will) for distributing particular information, or inadequate processing of that information between parties, leading to conflicting views.

To increase communication consistency, two core issues must be addressed. The first is to ensure consistency in how communication is technically arranged. All sites and teams must be provided with adequate means to communicate with each other, thus providing consistent **access to knowledge**. The second is ensuring consistency in how content is communicated, which is much more difficult. Teams and individuals should be able to reflect on their communication mechanisms and recognize critical differences and similarities that prohibit working communication. A method to handle issues leading to inconsistent communication should then be constructed. The method should also promote equality – communication should be a dialogue, effectively requiring developers to use and develop their social skills.

4.2 Operational consistency

With operational consistency, we mean the extent to which social structures in an organization are aligned with operational structures governing how work is performed. When these are inconsistent, the formal, operational structure pulls in a different direction than what employees do, creating tension and problems. Such issues are amplified when there are inconsistencies between sites.

Differences in **organizational hierarchy** between sites easily create disruptions. Developers working with certain assumptions of organizational hierarchy may find it difficult to interact with others operating under different assumptions, as illustrated in our material: *"because in Central Europe [there tends to be] a steeper hierarchy. You cannot dismiss the rules of conversation. Architects only talk to architects and software developers to software developers. In Finland, there's no need for such hierarchy. We can speak freely with anyone."* Furthermore, many operational practices include assumptions of organizational structure, e.g., many agile methods assume cross-disciplinary and self-organized

teams. This theme is also linked with communication consistency, particularly working culture – the norms and guidelines developers from different sites and cultures are accustomed to, and how they should be communicated to be understood correctly. Operational consistency focuses on organizational structures; communication consistency is concerned with how developers talk about them.

Processes are the primary means to organize software development activities over time, and they contain many assumptions about social work organization. For instance, how strictly processes are followed, how exactly they are assumed to specify what to do, and what developers can do beyond what processes say. Issues related to processes often escalate due to mismatches in **expectations** about results, progress, and what others do: *"it might be that in some places [...] you are told very specifically what you must do and how you must do it, and initiative is not part of the process. Then again others might think that if you just say something very broadly, like start doing something like this, [...] people expect you to be more active [or] you make more independent choices."*

Differences between sites can lead to differences in available resources (both people and equipment). For example, sites A and C in our material were both working on a hardware-dependent piece of software, but had different versions of the hardware. Tests that passed at Site A would fail at Site C. Developers at Site A would *"think that Site C are in the wrong, and then no one knows where the problem is."* Finding the root cause of a problem in such a situation can take a long time, particularly when the other site feels like they are being blamed for problems they have not created. **Commitment** is closely linked to site differences. Feeling blamed for problems, experiencing a lack of respect and not being involved in decision-making will quickly lead to uncommitted developers, particularly at remote sites. Lack of commitment, in turn, will affect how well work is carried out, how processes are followed, and how communication is handled, and thus commitment issues should be carefully handled.

Participants reported issues arising from assumptions that everyone has understood things the same way. Integrating **documentation instructions** into the operational procedures is one way of combating this problem, as in this example from a case company: *"we aim to say every obvious thing out loud ... and then we also write these so-called self-evident truths down in the design documents."*. Here, developers had noted how mismatches in assumptions could lead to serious problems, and had created a consensus to help each other via documentation. However, inconsistency may arise if sites differ in how they approach the documentation activity: *"I have seen some examples [where expectations], particularly in Site B, for documentation was to be extremely detailed. It's almost instructions [for] 'how to write code', and that's not how we work here."*. We can see how both parties can quickly become equally frustrated – Site A from feeling the need to document items they feel are self-evident, and Site B from feeling they are not being given all the information they need. Flexibility from developers on both sites is required to find a working process.

Operational inconsistency may appear as differences in **technical learning curve**. Developers working at different sites often have different educational

backgrounds. Thus, learning new tools and technologies is easier for some than others. If an organization does not give sufficient support via training to bridge this gap, differences in the skill base may result in conflicts.

The aforementioned issues describe poor *operational consistency*. Development processes, practices, ways of working, and other operational methods may be implemented or understood differently at different organizational sites due to differences in organizational structure, local organizational or national culture and norms, or other social factors. Low operational consistency leads to clashes between planned work practices and people's natural social interactions and dynamics around work organization. Operational consistency can be improved if the organization, operational practices, and social interaction between individuals match. Thus, to enforce operational consistency, managers should allow teams of developers (in effect, the organization structure) to flexibly restructure itself around tasks. An organizational structure or mode of working should not be imposed on teams as a matter of standard policy, but such decisions should be based on situational evidence and followed by attention to the social interactions that emerge, leaving room for individuals to take initiative. A key prerequisite of operational consistency is trust. The less trust people at different sites have, the heavier and less flexible processes need to be used. Thus, establishing trust between developers should be a high priority for GSD managers [22].

4.3 Wheels in motion

To aid the aforementioned challenges in people management and soft issues, we need to explore the GSD organization beyond STC. Our analysis reveals two dimensions affecting GSD – operational and communication consistency – that complement the existing and widely recognized STC. The new consistency dimensions are distinct from STC as they do not directly concern the software artifacts under development. Rather, they reflect the fit between work imagined on the drawing board and how developers are actually able to carry it out.

The different consistencies in our model are interlinked, and *if one wheel breaks, the whole system comes to a halt*. Problems in one organizational dimension will reflect on and spread to other dimensions (see Fig. 2). Poor operational consistency is often caused, e.g., by processes that do not support daily activities, such as those with hierarchies and actions that do not fit agile development within teams. Lack of support for daily activities will easily lead to developers not knowing what they should be doing and with whom. This will soon be visible as erratic communication between developers – a decrease in communication consistency. Vice versa, developers who are unable to properly communicate (inconsistent communication) will likely be unable to follow desired processes. Thus, inadequate support for operational consistency (poor match between daily work and process) will manifest as poor communication, and poor communication consistency will reflect back on operational consistency. The two will impact STC, as they either disrupt coordination activities or deteriorate the software architecture in such a way that coordination no longer matches what the organization is capable of handling. Improvement and maintenance actions that strive

to make GSD work better must focus on the root causes within the consistency dimensions. Similarly – if communication between developers is fluent, hick-ups in processes are easier to solve, and when processes support daily tasks, there is both less need for communication and better grounds for fruitful, two-way conversations.

Finally, distances affect the consistency dimensions. We saw social distance particularly in issues related to organizational hierarchies, inequality, commitment, working culture, assumptions and social learning curve. Temporal and geographical distance were visible particularly in processes and communication mechanisms – how to handle development tasks and communication when face-to-face meetings are not possible. Distances also often create distrust, and unclear reasoning (often due to poor communication) can also easily create fear. Fear and distrust can cause inconsistency, and that inconsistency can feed distrust in a vicious cycle. Similar findings, stressing the need to agree about the norms of work and build discipline toward the process are reported by Piri [22].

5 Discussion

Our empirical material suggests that social factors in GSD can be viewed as a multi-dimensional system, having two consistency dimensions interacting with STC, all affecting one another and being influenced by distances. In the following we will discuss our research questions and the limitations of our study.

5.1 The Socio-Technical System

Based on our study, we propose reconsidering how we view GSD. We set out to study social interactions in GSD, starting with RQ1: *How does the coordination process in GSD manifest in terms of social factors, structures, and interactions?*. The coordination process in GSD manifests in terms of social factors, structures, and interactions as a model with two consistency dimensions. With STC, these form a system (as given in Fig. 2) that can explain why coordination works or breaks down in the interactions between individual developers and the overall socio-technical system that is active in GSD.

We wanted to probe deeper in to the coordination process with RQ2: *"What factors influence the quality of the coordination process?"*. The quality of the coordination process is influenced by the degree of consistency in the communication and operational dimensions. Our interview material highlights the difficulties practitioners face when there are inconsistencies between processes and practices. A lack of consistency commonly caused increased dissatisfaction, lack of motivation, and frustration among the practitioners. These co-occur with delays in schedule and with sub-optimal quality of the resulting software.

Finally, our attempt was to elicit a way of correcting found issues in addition to just identifying them, as we posed RQ3: *"How can the threats to the coordination process be mitigated?"*. The first step is to identify how inconsistencies in communication and operations can threaten effective coordination of software

development. These threats can then be mitigated by distributing coordination work and empowering developers to coordinate their parts of the socio-technical system. Distributing coordination work requires workforce training and a high degree of trust. It also requires improving ways, forms, and balance in communication and aligning the formal, operational structure in the organization with the natural social interaction structures that exist on an interpersonal level. These activities could be built into the software development process by including distributed coordination activities on all levels of the organization.

Accepting that the development of any large piece of software requires social interaction, the stereotypical image of a programmer as technology-oriented and socially inept is challenged in GSD. A developer should be capable of adapting to different communication styles to increase communication consistency as well as to different ways of coordinating work to increase operational consistency. Several of the comments by participants in our study shows that some developers are already tacitly aware of such issues.

Achieving wide improvement in GSD escapes formal definitions of organization, processes, and procedures, and relies on organizations' ability to foster social developers and their cooperative skills. Furthermore, relying on developers' social skills requires that the management is more informed regarding the mechanisms of social behavior among software developers.

5.2 Related Work

Mariani [17] presents recent advances in coordinating socio-technical systems, and stresses the need to include socio-cognitive aspects in technical solutions from the very beginning of design. While current work in GSD mostly relates to the technical-to-social mindset as defined by Mariani, we could utilize the principles and theories of social-to-technical mindset to improve coordination. Similar relationships between social protocols, coordination mechanisms (operations) and communication, as presented here, have also been given as a framework by Giuffrida and Dittrich [12]. They base the framework on theories of coordination mechanisms and communicative genres, and support the rather abstract framework with ample examples from their empirical study. However, this framework concentrates heavily on communication alone, while we present a more balanced view of the different social aspects in GSD projects.

Furthermore, Jolak et al. [15] discuss communication aspects of GSD in the light of geographical and social distance. They analyzed the categories of collaborative discussions in joint design tasks and noted that collocated and distributed teams differed in the quality of the communication – the amount of creative debate was smaller in distributed cases. This can be seen as lack of communication consistency and is assumed to lead poorer design since creative thinking and constructive criticism are reduced. Robinson [23], in turn, reports findings how team members spread across sites do not feel that they belong into the same team, even though week-long face-to-face meetups are arranged twice a year. This highlights the need for continuously upholding communication consistency. Similar problems in communication consistency, and particularly the *form* and

balance of communication have been identified by Stray et al. [31], who studied the use of Slack in virtual agile teams. They noted that even when using such direct messaging, there are significant differences in levels of activity, stemming mainly from language skills and knowledge level. Further, using too much personal mode was found a barrier - supporting our identification of social learning curve as an essential theme in communication consistency.

Problems similar to what we found with regard to operational consistency have also been reported by Hussain et al. [14] in the context of requirements management and informal change requests. Informal requirement changes are a direct product of deviating from defined processes, and contain elements of challenges related to organizational hierarchy, varying expectations and inequality between sites – all elements of operational consistency. Furthermore, Björn et al. [4] also report issues around similar themes in their empirical study on how agile methods were adopted in a global setup – severe challenges were discovered as a direct result of inadequate matching of operations (development methodology) and social interactions.

Finally, Sigfridsson [30] reports empirical findings from an organization where consistency was well supported. Teams would actively work on and adjust their practices to allow for better collaboration across sites, and the organization had several actions alleviating, e.g., inequality between sites, such as worldwide seminars to learn new technologies. In the organization in question, teams did not think of distribution as a problem, but just as a mundane thing. This work shows how truly important it is to support and achieve a balance of consistencies.

5.3 Limitations and Validity

There are some limitations to be addressed regarding our study, namely completeness, potential bias and limits to data synthesis. In addition to limitations, as with any study, we must consider the potential threats to the validity of our results. We address threats to validity following Maxwell's categorization [18].

Limitations We first consider potential bias. While the first author was involved in a previous study utilizing the same interview material as here, none of the other three authors were involved in the previous study in any way. There were thus three researchers, who had a neutral and objective stance to the material, validating all the findings. As all steps in the research process were defined jointly and included validation with all authors involved, we are confident that there are no substantial risks related to bias.

Second, we consider the completeness and coverage of our interview material. While one could always wish for a larger set of interviewees, thirteen interviewees already gives us a credible sample when we consider the variance in companies. Our interview material was gained from seven different companies, covering together almost 20 different sites, with headquarters in three different countries. The companies also varied in sizes and domain.

Finally, we need to consider the limits of our data synthesis. We used a form of thematic analysis [5], which potentially lacks in transparency. To avoid this potential drawback, all steps leading to creation of the consistency model have

been validated - anti-patterns derived from the quotes, coding of anti-patterns under themes, and arranging of themes in the consistency model. Thus, we have traceability from our model back to raw data (quotes) as well as validated outcomes. However, the actual creation of our consistency model did not follow a strict procedure, but is a product of joint interpretative synthesis conducted in a workshop setting, and we acknowledge the weakness of its repeatability.

Threats to validity Descriptive validity concerns accurate recording and presentation of the data, based on which conclusions are made. All the interviews were recorded, and the recordings were transcribed by an independent professional. Transcriptions were copied verbatim into an analysis tools (NVivo), from which individual quotes have been extracted. In case a single quote was difficult to interpret, using the tool it was easy to find the context surrounding the quote.

Threats related to theoretical validity are ultimately concerned with whether we captured what we intend to in relation to our hypothesis. Our study was exploratory. As we had no hypothesis, but an open research question, there were no risks that interview subjects, questions or the process would be biased towards confirming our hypothesis. However, there are validity threats regarding how well our material is suited to answer our research questions. The interview protocol was not designed to uncover issues or practices related to social interactions. Thus, issues discovered in the interviews may lack sufficient context or details.

Interpretive validity threats concern correct interpretations of the material. As discussed above, we consider researcher bias not to be a real risk in this study, as we did not have a clear hypothesis or a pre-determined vision of how the material would answer our research question. The results were derived purely based on the research process and combined analyses of the researchers. To strengthen the validity of our joint analyses, we always referred back to the original interview quotes and checked that the quotes supported our findings.

Regarding generalizability, we are confident that internal generalizability (within the field of software engineering) is fairly well satisfied due to variance in the represented companies in terms of size, involved sites and domains. There is no reason to believe that the results would not in general apply to companies in the field of software engineering involved in GSD.

6 Conclusion

We explored social factors affecting GSD beyond those defined as a part of STC. We specifically wanted to answer questions related to the coordination process in GSD, it's quality and how can threats to the process be mitigated. Our research has led to a new model of the GSD organization, composed of two consistency dimensions, STC, the developer at the center, and distances affecting all actions.

Solving the problems of GSD requires continuous attention to multiple, complex and interlinked phenomena with organizational, architectural, operational, cultural, and communication factors. Given the complexity of this challenge, it is unlikely that a formal, top-down approach alone will be successful. The role of management shifts towards incentivizing, coaching and mentoring, and pro-

viding developers with resources they need to accomplish organizational goals. By identifying threats related to inconsistencies and carefully considering the interlinked nature of organizational dimensions as revealed in our study, GSD organizations can increase consistency in all dimensions and make their software development engine run more smoothly.

References

1. Ågerfalk, P.J., Fitzgerald, B., Olsson, H.H., Conchuir, E.Ó.: Benefits of global software development: the known and unknown. In: Proceedings of International Conference on Software Process (ICSP'08). vol. 5007, pp. 1–9. Springer (2008)
2. Bano, M., Zowghi, D., Sarkissian, N.: Empirical study of communication structures and barriers in geographically distributed teams. IET Software **10**(5), 147–153 (2016)
3. Bass, M.: Monitoring gsd projects via shared mental models: A suggested approach. In: Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner. pp. 34–37. GSD '06, ACM, New York, NY, USA (2006)
4. Bjørn, P., Søderberg, A.M., Krishna, S.: Translocality in global software development: The dark side of global agile. Human–Computer Interaction **34**, 174–203 (2019)
5. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative Research in Psychology **3**(2), 77–101 (2006)
6. Carmel, E., Agarwal, R.: Tactical approaches for alleviating distance in global software development. IEEE Software **18**(2), 22–29 (2001)
7. Casey, C., Richardson, I.: Implementation of global software development: A structured approach. Journal of Software Evolution and Process **14**(5), 247–262 (2009)
8. Cataldo, M., Herbsleb, J.D., Carley, K.M.: 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. pp. 2–11. ACM (2008)
9. Conway, M.E.: How do committees invent? Datamation **14**(4), 28–31 (1968)
10. Dennis, A.R., Fuller, R.M., Valacich, J.S.: Media, tasks, and communication processes: A theory of media synchronicity. MIS Q. **32**(3), 575–600 (Sep 2008), http://dl.acm.org/citation.cfm?id=2017388.2017395
11. Feldman, D.C.: The development and enforcement of group norms. The Academy of Management Review **9**(1), 47–53 (1984)
12. Giuffrida, R., Dittrich, Y.: A conceptual framework to study the role of communication through social software for coordination in globally-distributed software teams. Information and Software Technology **63**, 11–30 (2015). https://doi.org/10.1016/j.infsof.2015.02.013, http://www.sciencedirect.com/science/article/pii/S095058491500049X
13. Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering **29**, 481–494 (2003)
14. Hussain, W., Zowghi, D., Clear, T., MacDonell, S., Blincoe, K.: Managing requirements change the informal way: When saying 'no' is not an option. In: 2016 IEEE 24th International Requirements Engineering Conference (RE). pp. 126–135 (Sep 2016). https://doi.org/10.1109/RE.2016.64

15. Jolak, R., Wortmann, A., Chaudron, M., Rumpe, B.: Does distance still matter? revisiting collaborative distributed software design. IEEE Software **35** (2018)
16. Levesque, L.L., Wilson, J.M., Wholey, D.R.: Cognitive divergence and shared mental models in software development project teams. Journal of Organizational Behavior **22**(2), 135–144 (2001)
17. Mariani, S.: Coordination in socio-technical systems: Where are we now? where do we go next? Science of Computer Programming **184**, 102317 (2019). https://doi.org/10.1016/j.scico.2019.102317, http://www.sciencedirect.com/science/article/pii/S0167642319301157
18. Maxwell, J.A.: Understanding and validity in qualitative research. Harvard educational review **62**, 279–301 (1992)
19. Meyer, A., Fritz, T., Murphy, G., Zimmermann, T.: Software developers' perceptions of productivity. In: In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. pp. 19–29. ACM (2014)
20. Mohammed, S., Ferzandi, L., Hamilton, K.: Metaphor no more: A 15-year review of the team mental model construct. Journal of Management **36**(4), 876–910 (2010)
21. Oshri, I., Kotlarsky, J., Willcocks, L.P.: Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects. The Journal of Strategic Information Systems **16**(1), 25–49 (2007)
22. Piri, A., Niinimäki, T., Lassenius, C.: Fear and distrust in global software engineering projects. Journal of Software: Evolution and Process **24**, 185–205 (2012)
23. Robinson, P.: Communication network in an agile distributed software development team. In: Proceedings of the ACM/IEEE 14th International Conference on Global Software Development (ICGSE). pp. 90–94 (2019)
24. Rothman, J., Hastie, S.: Lessons learned from leading workshops about geographically distributed agile teams. IEEE Software **30** (2013)
25. Sahay, S., Nicholson, B., Krishna, S.: Global IT Outsourcing: Software Development Across Borders. Cambridge University Press (2003)
26. Sierra, J.M., Vizcaíno, A., Genero, M., Piattini, M.: A systematic mapping study about socio-technical congruence. Information and Software Technology **94**, 111–129 (2018)
27. Sievi-Korte, O., Beecham, S., Richardson, I.: Challenges and recommended practices for software architecting in global software development. Information and Software Technology **106**, 234–253 (2019)
28. Sievi-Korte, O., Richardson, I., Beecham, S.: Protocol for an Empirical Study on Software Architecture Design in Global Software Development, Lero Technical Report No. TR_2019_01. https://www.lero.ie/sites/default/files/TR_2019_01_Protocol_for_GSD_Arch_Design_Framework.pdf (2019)
29. Sievi-Korte, O., Richardson, I., Beecham, S.: Software architecture design in global software development - an empirical study. The Journal of Systems and Software p. in press (2019)
30. Sigfridsson, A.: A conceptual framework to study the role of communication through social software for coordination in globally distributed software teams. Ph.D. thesis, University of Limerick, Department of Computer Science and Information Systems (2010)
31. Stray, V., Moe, N.B., Noroozi, M.: Slack me if you can! using enterprise social networking tools in virtual agile teams. In: Proceedings of the ACM/IEEE 14th International Conference on Global Software Development (ICGSE). pp. 101–111 (2019)
32. Tamburri, D.A., Kruchten, P., Lago, P., van Vliet, H.: Social debt in software engineering: insights from industry. J Internet Serv Appl **5** (2015)