Coordination of software development teams across organizational boundary – An exploratory study

Anh Nguyen-Duc
IDI, NTNU
Trondheim, Norway
anhn@idi.ntnu.no

Daniela S. Cruzes
IDI/NTNU and SINTEF IKT
Trondheim, Norway
dcruzes@idi.ntnu.no

Abstract— **Coordinating teams across geographical, temporal and cultural boundaries has been identified as a critical task to achieve the success of global software projects. Organizational boundary is another dimension of global distribution, which is a less visible but equally important factor that influences team coordination. This study investigates attributes of the organizational boundary that inhibits coordination and development activities. Besides, we explore a set of effective coordination practices to overcome organizational boundary. The data were collected from two projects involving four different software development organizations. We found that the variety on collaboration policy, team organization, engineering process, and development practices contributes to extra coordination efforts, insufficient communication, team awareness and mistrust. The study also highlights that coordination practices, such as face-to-face contact, process synchronization and shared collaborative development are compulsory but not sufficient for effective team coordination across organizational boundary.**

Keywords; global software development, organizational boundary, coordination, collaboration, empirical study

I. INTRODUCTION

Global software development (GSD) promises shortening time-to-market, fastening technology innovation, increasing operational efficiency and reducing the proximity to customers [1]. However, there are a number of problems related to economical, social and technical issues that still need to be solved before the full potential of GSD can be reached. One of the main problems is that team coordination in GSD is insufficiently supported [2].

Coordination, understood as activities to manage dependencies between tasks and task holders, is essential for orchestrating a large software project. In collocated software projects, poor coordination is accounted for uncertainty and complexity of tasks [3, 4]. In GSD context, distributed software team needs to handle not only the complexity and evolution of requirements and technical tasks, but also dynamic interdependencies between tasks and task holders across multiple global boundaries. Consequently, tasks that spread through many locations take much longer time to complete than collocated tasks, and suffer from lower team productivity and software quality [5, 6].

A recent trend of research on GSD has focused much on improving socio-technical coordination [2, 7], in which project stakeholders coordinate on multiple layers of artifact dependency, task dependency and social dependency, as shown in Figure 1. Many empirical studies target to these layers, assuming that stakeholders belong to a homogeneous organizational environment. An often-neglected dimension of socio-technical coordination is the organizational boundary among stakeholders, as not all of them belong to the same organization. Organizational boundary means that there will be differences in engineering processes, role structure, functional and strategic concerns occurred among organizational units [8]. Curtis et al. noted that coordination breakdowns were likely to occur at organizational boundaries, and coordination across these boundaries was often extremely important to the success of the project [10]. Espinosa et al. stated that organizational boundaries can affect project outcomes by reducing awareness of context and shared identity [8].

Figure 1 illustrates two types of organizational boundaries occurred at departmental level (between Team A1 and Team A2) and firm level (between Team A1 and Team A3). These boundaries introduce barriers between developer D1 and D2 and also between D1 and D3, influencing the implementation of task A1 and A2. While organizational distance between two firms is expected, we argue that there are also attributes of organizational boundary that matters for coordinating two development units within the same organization.

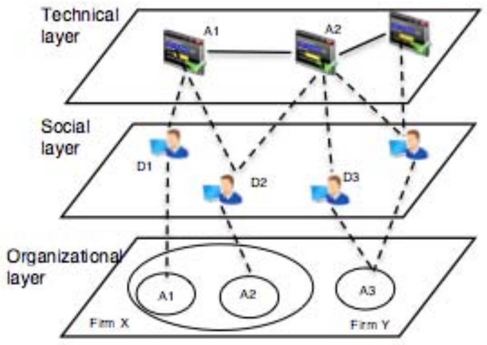

Figure 1: Organizational perspectives on socio-technical coordination (adapted from [12])

Our objective in this paper is to explore different attributes of organizational boundary, its consequence and remedies in the context of GSD. The first step of our study is to explore the attributes of organizational boundary and their outcomes on coordination among project team members. After identifying a list of organizational boundary attributes that cause coordination problems, we investigated the solutions that have been effectively adopted by practitioners. The two main research questions are:

- RQ1: What are attributes of the organizational boundary that introduce challenges on team coordination in GSD?
- RQ2: What are the coordination mechanisms that help to mitigate organizational boundary?

The extent of organizational boundary and its impact on team coordination can vary in different organizational relationships. Therefore, we examine and compare two projects with different organizational relationship, one leveraging company-internal resources in a different country (onshore outsourcing) and the other leveraging external third-party resources situated in the same country (offshore insourcing) [14].

The rest of the paper is organized as follows. In Section 2, we present an overview of related work to organizational boundaries, team coordination challenges and coordination mechanisms in GSD. In Section 3, we describe an empirical study design and context background of the cases. Results from the empirical studies are provided in Section 4. It is followed by a discussion in Section 5. Finally, Section 6 presents the conclusion with the summary of major findings.

II. Related Works

A. Organizational boundaries in GSD

While exploring challenges in GSD, SE researchers have found several issues with organizational boundaries, which can influence team coordination [13-20]. Divergence in terms of organization's objectives, policies and strategies has a visible impact on developers' work [13]. Espinosa et al. found that collaboration policy established by both side of a collaborated project affects the way work was coordinated [7].

Organizing teams across geographical and organizational boundary also plays a role in supporting team coordination. Hinds et al. found that informal hierarchical structure was associated with more smooth coordination on distributed teams [16]. Nagappan et al. investigated Vista projects and showed that measures of organizational structure have a significant relationship to software quality, and this relationship is stronger than ones of geographical or temporal factors [11].

The issue of a common process in software specification, implementation and maintenance is also nontrivial. Kommeren et al. reported experience with GSD in Philips and reported problems with lack of common process about acceptance procedures of mutual deliveries, escalation mechanisms, lack of clearly assigned responsibilities and lack of central control of integration [18]. Prikladnicki et al. showed that the software development process variation is more severe in offshore outsourcing projects than in internal offshoring projects [22].

Lastly, organizational boundary introduces not only variation in development processes, but also in development practices, such as coding and comment styles, testing approaches and adoption of development tools. The practices are likely informal and not defined by the overall process. Smite et al. found that coordination by standard is hindered by problems with disparities in work practices and limited feedback from remote locations [19]. Moe et al. also reported challenges with acquiring trust introduced by disparities in work practices [20]. Bhat et al. showed problems with conflicting RE approaches on disagreements in tool selection and communication issues [13].

It appears that the differences in the organization's policy, the way teams are organized, variation in development process and work practices are potential challenges for cross-team coordination. We conceptualize these attributes as Collaboration Policy, Team Organization, Engineering Process, and Development Practices and use this as an initial checklist to explore the organizational boundaries.

B. Team coordination challenges in GSD

In the scope of this study, we use the term "collaboration" to refer to the act of working with other to create something together in general. The term "coordination" refers to "activities required to maintain consistency within a work product or to manage dependencies within the workflow" [21]. Several types of dependencies have been identified in the context of Information System (IS) and Software Engineering (SE), such as [21, 22]: technical dependency (i.e., needed to integrate software parts seamlessly), temporal dependency (i.e., necessary to synchronize activities and adhere to project schedules), software process dependency (i.e., required to adhere to the established software processes and act as agreed with other team members), and resource dependency (i.e., a team needed business domain knowledge from the other team to accomplish a task). Insufficient management of these dependencies causes coordination problems. The challenges identified in previous literature are [3, 9, 20, 23-28]:

1. Large and complex dependency network: many stakeholders from different sites [3], which creates a larger dependency and communication network than in collocated projects;
2. Delay in communication and coordination: communication and information exchange take longer time in GSD than in collocated projects [3, 23];
3. Limited choice of coordination mechanism: distances among sites disable team coordination in face-to-face manner [9];
4. Difficulty on identifying coordination requirement: identification of people who are technically inter-dependent is difficult [24, 25];

5. Difficulty in scheduling task: the involvement of many sites across geographical locations and time zones makes scheduling common meetings difficult [26];
6. Lack of trust: it is harder to establish mutual trust among project stakeholders in GSD than in collocated projects [20, 27];
7. Lack of team awareness: GSD team members are often unaware of tasks and activities of other developers from remote sites [23, 28];
8. Misinterpretation: team members can misunderstand each other working in different environment and contextual information is not given sufficiently [9];

While some of these challenges can be accounted for a specific type of global distances, i.e. geographical distance lead to lack of face-to-face contact, most of the challenges with team coordination are often caused by many reasons and hard to identify the root cause [9]. In this study, we focused on coordination challenges created by organizational boundary attributes. To best of our knowledge, there is no empirical study that comparatively and systematically investigates the influence of different attributes of the organizational boundary on team coordination before.

C. *Effective coordination mechanisms*

Dependencies among tasks and task holders are handled by using coordination mechanisms, which are practices, methods, strategies or policies that are used to manage the dependencies among team members and tasks [21]. Previous studies suggests several effective coordination practices for inter-organizational software projects:

1. Synchronization of main milestones [29]: customer and vendor teams keep their own development processes and only main phases and milestones were synchronized;
2. Frequent deliveries [29]: deliveries are integrated and tested in a daily basic;
3. Establishment of peer-to-peer links [29]: contact points for manager-to-manager and developer-to-developer;
4. Early relationship building [29]: having involved personnel meet face-to-face in beginning phases of a project
5. Early architectural assignment [25]: assignment of task corresponding to architectural modularization;
6. Shared management and development tool [30]: having the same set of tools across sites;
7. Boundary spanner [31]: using an unofficial role of boundary spanners as a explicit coordination mechanism

Table 1: Data collection

Data collection	11/2011 – 1/2013	2/2012 – 5/2012
No. of interviews	7	3
Interview roles	PM (1), Team leader (2), Dev. (4)	PM (2), Dev. (1)
Interview across site	A1 (5), A2 (2)	B1 (3)
No. of observations	A1 (2)	-

This list is used as a basic for exploring and comparing effective coordination mechanism in our cases.

III. Research design

A. *Study design*

We investigated two GSD projects involving four different organizations. The reason for choosing these projects was that (1) the projects involved GSD and has faced with coordination challenges, (2) the projects represented different organizational relationship, and (3) the projects offered availability of rich data sources and multiple points of view. The unit of analysis is a development team, which matches with the objective of exploring inter-team coordination activities.

B. *Data collection and analysis*

We relied on data triangulation as defined by Yin [32]. Our data collection was based on:

- Interviews as a primary data source
- Documentation, and passive observation as a secondary data source

We conducted 10 semi-interviews (3 of them conducted via telephones) with different project roles, such as project manager, technical leader and developer. Interview guide consists of 4 main parts: (1) project and interviewee background, (2) general challenges when collaborating with others from distance, (3) specific problems related to organizational differences when working with other teams, (4) best practices when working with teams from other organizations. The interview guide was designed using the checklist from the literature review. Each interview lasted from 50 to 75 minutes. The interviews were recorded and then transcribed.

Two observation sections were performed in Case A, as given in Table 1. The first observation focused on a 4-hour work of a team leader, the later one focused on a 4-hour workshop between Team A1 and A2 (section III.C).

We collected 49 A4 pages of interview transcripts and 8 A4 pages of observation notes as a major data material. Other materials such as email exchanged, company website, product and project specification was collected for understanding case context. All transcripts were sent to interviewees for confirmation.

We analyzed the data in several steps. Firstly, we read all interview transcripts and relevant documents, and coded them according to open coding [33]. Each segment of text that expresses either organizational boundary, team coordination problem or effective coordination mechanisms were labeled with an appropriate code. Secondly, we compared segments from different interviews, but from the same case, that were assigned the same code (axial coding).

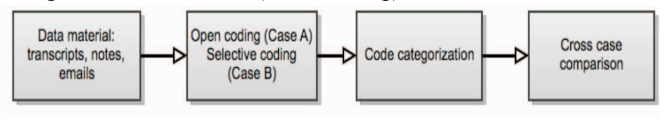
Data material: transcripts, notes, emails
Open coding (Case A) Selective coding (Case B)
Code categorization
Cross case comparison

Figure 2: Data analysis process

Table 2: Cases context

Attributes	Case A	Case B
Project type	Bespoken	Market driven
Global arrangement	Outsourcing	Insourcing
Project size	Medium	Large
Project age	5 years	5 years
Product	Ship management system	Search engine system
Involved organization	A customer team and two software vendors	A headquarter and two offshore sites
Team size	13	150
Inter team dependencies	Technical	Technical, temporal and process
Site locations	3 sites within Norway	3 sites in USA and Norway

In axial coding, indicators and characteristics for each concept were searched for, in order to define that concept [33]. Finally, the concepts were compared between two cases and the differences were discussed. The overall process is described in Figure 2.

C. Case contexts

This section describes a context of each project, including products, team structure, coordination practices and the need for inter-team coordination, summarized in Table 2.

Context in Case A:

The project studied in Case A is a ship management system. The project involves two external contractors, so-called Team A1 and Team A2. Team A1 locates in Trondheim, Norway and implements the web application systems. The internal coordination practices include face-to-face discussions, informal meeting, team foundation server (TFS) and adoption of a lightweight task tracking system. Task assignments are conducted based on early architectural division.

External communication with customers is mainly done by phone and email. A few times a year, the Team A1 team leader visits external stakeholders (customer team and Team A2). Team A2 locates in Stavanger, Norway and implements the desktop application. The development methodology is Scrum. External collaboration is done via email, and frequency of email exchange varies according to the needs of the project. The communication between Team A2 and external stakeholders is based on phone, face-to-face meeting, email and TFS.

Team A1 and A2 are responsible for the two different parts of the systems, but sharing a common data access layer and database. When someone develops or modifies a feature that needs to update the database, the change had to be informed to the other team for updating affected features. There was little dependency between two teams in the beginning of the projects. At the time this study was conducted, both teams were performing a joint effort on refactoring overlap code base and updating common data model. For developing the new feature, Team A1 had a resource dependency on Team A2 as a key member of Team A2 know much about business logic of the whole system. Both teams had a technical dependency on each other as they developed their part in parallel and need to synchronize these parts and keep the data model consistent.

Context in Case B:

The project in Case B is a search engine system as part of a large IS management system. The search engine system has many API interfaces, which are used by many higher-level systems. The part of the project that we investigated only relate to a team in the headquarter office in USA and two development centers in Norway, namely Team B1, B2 and B3, correspondingly.

Team B1 locates in the headquarter office of Cooperation X in USA. X is a large software production organization with established software development processes and many years of experiences in GSD. Common set of collaboration practices are used among Team B1, B2 and B3, which are daily (virtual) meeting, teleconferencing meeting, email and TFS. A few times a year, there is an exchange of team members between USA and Norway. Team B2 and B3 were initially two branches of a Norwegian company, developing a standalone search product. After acquired by an US company, Team B2 and B3 became a part of development centers in Norway. In addition to common collaboration practices for the whole cooperation, Team B2 and B3 remains local collaboration practices, such as communication via Git, informal chat and frequent visit between Team B2 and B3.

Collaboration between B1, B2 and B3 is a basic need for developing the main system. There is high amount of technical, temporal and process dependency among these teams. Each team develops some modules that will be integrated later in the process. The synchronization of tasks is crucial to achieve overall project milestones and quality of the whole search system.

IV. FINDINGS

This section presents attributes of organizational boundary that contribute to inter-team coordination problems. Table 3 presents the organizational boundary attributes and its consequence on coordination and development activities. Besides, we also describe how the coordination mechanisms were used to close this gap. Each aspect is illustrated with quotes from the interviews.

A. ***Findings on Case A:***

A.1 Collaboration policy

In this case, collaboration policy refers to an established governance policy by customer and competition attitude of each team towards other organizations. Firstly, **customer governance policy** did not support inter-team coordination on tightly inter-dependent tasks, which contributed to problems of lack of *inter-team awareness, technical debt* and *source code replication* between two teams.

Table 3: Organizational boundary attributes

Organizational boundary attributes	Coordination problems	Development problems	A	B
Collaboration policy				
Governance policy on team collaboration	Lack of inter-team awareness	Technical debt, code duplication	X	
Competition attitude	Limit of context sharing,	Limit of knowledge sharing	X	
Different expectation on task responsibility	Extra coordination effort	Effort on conflict resolution		X
Team organization				
Mismatch in code submission structures	Difficulty on communication		X	
Different branching models	Extra coordination effort			X
Engineering process				
Scrum vs. Iterative development	Lack of in-depth coordination	Long knowledge transfer process	X	X
Different integration and release frequency	Communication delay	Difficulty in schedule meeting	X	
Focus on quality vs. productivity	Difficulty on task coordination	Difficulty on task implementation		X
Development practice				
Different coding and comment styles		More effort on shared code	X	
Difference on automation testing approaches	Mistrust on code quality		X	X
Various usage of collaboration infrastructure	Unnecessary coordination effort		X	X

Respondents from both organizations and in both roles, developer and manager, were aware of a tight control mechanism over collaboration between two teams in the beginning of the project. A developer from Team A2 stated that until last year, inter-team communication is mainly done via customer. Any necessary technical discussion was controlled by the customer team. Another developer from Team A1 also said: "*... the product owners didn't wish us to talk to [Team A2] without the presence of the … This was based on the kind of suspicions that we, as two consultant companies, would make stuff up to make some money*."

Both teams had worked almost independently until the coordination needs had extrapolated to what the customer team was not able to manage. As each team worked on their own codebase, they unexpectedly created *redundant code* on data access layers and some utility functions. The overlap source code became larger and larger overtime, resulting in a technical debt, which needs to be resolved before starting a new highly joint development tasks.

Later on, the governance policy on collaboration between vendor teams had been loosed up, partly because of the long-term relationship between customer and each team had been established. Both teams could have meetings or discussions without customer representative. However, we still observed a low level of awareness across organizational boundary. Both team did not know much about how the other team's status and work progress. Two interviewed developers from Team A1 were not aware of status of Team A2 and two other interviewed developers from Team A2 were also not aware of status of Team A1: "*Researcher: Do you talk to Team A2 about progress and status of the work? DevA1: … not that much as we should. We are very autonomous in this aspect. But we do talk a lot with [Customer names] about progress, not much to [Team A2] …*"

Secondly, the **competition attitude** to each other also created a barrier for team coordination. Each team belong to a software vendor with similar capability in the market, who are directly competitors in getting contracts from the customer. The better relationship and perceived credits from the customer could lead to more contracts for each team. The attitude of competition was recognized from both Team A1 and Team A2. A technical leader from Team A1 said that: *"Regarding to the organizational dimension, it is not a problem with legal issues. But they do collaborate as competitors, so we need to keep the competition capability"*. One developer from Team A2 also stated that: "*…so we are in the competition with [Team A1]. Maybe [Team A1] want to do it too so we need to discuss and make argument about why should we do it instead. This is absolutely one part of the competition. But we also want to look good compared to [Team A1]. And we know they want to do the same …*"

Although this attitude was positively perceived by the customer since it motivated both teams to perform better and achieve higher level of customer satisfaction, the competition attitude limited *contextual information sharing*, *knowledge sharing* and the *teamness* between two teams. As recognized from developer's perspective, there was a "comparison mindset" towards the other team, in which knowledge domain and customer recognition were competitive advantages: "*We enjoy working in a way that satisfy customer because we know we gain something compared to [Team A2] and the other way around for [Team A2]*". By reading emails exchanged between Team A1 and Team A2, we observed that little information related to customer's status and project information were exchanged. Most of the time, both teams discussed about technical contents.

Besides, competition attitude hindered knowledge transfer between two teams. Team A2 adopted a strategy to discuss about business logic of a joint developed features just-enough and just-in-time. One developer from Team A2 said: "*[Team A1] relies on us 100% because they do not have business domain knowledge. If some of us is on vacation, it is kind of not having control in this case and high effect on others..."*

Moreover, the feeling of working as a big team was not observed from both teams. Across different interview transcripts, the used pronouns were "we" versus "they" when talking about team collaboration. The customers' name was mentioned from interviewee from both sides, but there is no private name of person from other team found in the transcripts: "... *Usually you don't ask other developers from another company, which you don't know him well. You don't ask him first even though you probably know that he is the one who knows the answer. Firstly you tried yourself and then you ask someone locally*."

A.2 Team organization

Team organization refers to challenges related to differences in team structure creating on team collaboration. We explored which coordination problems arise when a team in a flat structure (Team A1) working with a team in a hierarchical structure (Team A2). As architecture of the system is clearly modular from the beginning, both teams have quite clear division of technical task and responsibility. However, the **mismatch in code submission structures** introduced *difficulty on team communication*. In Team A1, team leader did most of the contact and receiving tasks. A task was broken down and given to the developers but they are not allowed to check in directly to a common VCS system. The team leader spent significant amount of time on delegating and integrating tasks in the team. He also performed some tests before checking in. In Team A2, developers had more autonomy on checking the code into the common VCS system. Therefore, for team A1, it was easy to find the relevant person from Team A2 who submitted a piece of code or a bug report. However, developers from Team A2 had to wait for information forwarding back and forth from the team leader in Team A1. As mentioned by a developer from Team A2: "*... if it does not compile you can send an email (to the team leader of Team A1). I would see what the check in comment is. I will do it before I contact him ...*".

A.3. Engineering process

Engineering process refers to the challenges of team coordination created by the **variation on development processes**. Lack of frequent and in-depth coordination during task integration led to *complexity and extra efforts on task integration*.

Team A1 and Team A2 had different approaches on capturing requirements, planning tasks, implementation and integration of the tasks. Team A2 adopted SCRUM with many small sprints while Team A1 used ad hoc and informal agile development, without splitting tasks into sprints and continuously receiving and developing tasks. The different development approaches hindered in-depth discussion on technical tasks at team level, as one developer said: "*...we work in such different ways so we don't share work processes. We communicated in a bit higher level than what we are doing now. As we see thing from Scrum, we have more defined tasks for near future and backlog items, but [Team A1] they could give feedback on how they are doing continuously*". It also limited *establishment of frequent discussion* among developers across sites. Coordination was driven on deadline basic rather than daily basic: "*... I am not very sure about [Team A1]'s deadlines. But they have kind of implicit deadline by that most of their system need to be ready for us ...*".

The difference on development **iteration duration** made teams harder to find common milestones for task integration, as mentioned by the team leader of Team A1: "*... If you work for one branch for half of a year, there is quite a lot of code to merge. [Team A2] is ready to merge every 2 months. ... [Team A2] is ready to launch every month but it costs too much to ensure the quality...*" Both teams had different set of internal deadlines regarding to the project. Inter-team coordination needs increased when Team A1 need to deliver a function that related to Team A2 and vice versa. However, at that time, Team A2 might not give much resource for cross-team synchronization activities, which lead to some issues as communication delay and difficulty on meeting scheduling.

A.4 Development practices

Variation on testing and coding approaches between two teams introduced *mistrust* and *extra coordination effort.* The difference on **code and comment style** was mentioned by one developer in Team A2. The developer experienced some difficulty on understanding code from Team A1 due to insufficient comments, lengthy and complicated code files. He suggested that: "*If all developers don't comment or clean over it you experience kind of low quality. We have established better mutual understanding about what good code is because we shared a lot of high-level source code. I am sure if we do a lot of high level code we would be able to come to common understanding of how it should be done*".

The difference on **testing approaches** also seemed to create a *mismatch on expectation of the quality* of the integrated codes. Automated testing reached 100% coverage in Team A1. Team A2 had only done automated testing partially. This introduced some topics for discussion between the teams when a bug with cross-boundary impact appears. The awareness of variation in testing approaches between two sites seemed to increase confidence of Team A1 when discussing on quality of code from each site.

Tool adoption refers to how **configuration management and development tools** were used in each team. Shared set of development infrastructures without a common set of adoption practices led to *unnecessary coordination effort*. Although both of teams shared the same configuration management tool, the way they used these tools were quite different. In Team A2, the functionalities of TFS were fully adopted, from recording requirements and modification requests from customers, storing source code, implementation and tracking and bug fixing. In Team A1, TFS was not actually adopted for communicating with customer and Team A2, as mentioned by a developer from Team A1 reported that: *"TFS have a lot of features for these kinds of things. We tried it out and we are kind of in another self-organizing way. We found it is too much bureaucracy for our way of working..."*

The only function of TFS shared by both teams was source code control. Therefore, some features of TFS that is beneficial for inter-team coordination, such as "gated check-in", which helps to enforce a clear contract between developers and avoid manual check of problems with logical source code structure, data type, variable naming, variable scope and error handling, were not used. A technical leader of Team A1 highlighted that the common process and practices are important for working over distances: "*If you establish guideline for coding, you have a lot of tools you can achieve the quality you want in the project. And it should not depend on whether you seat on other country or city or whatever."*

A.5 Effective coordination practices

We also observed an adoption of effective coordination mechanisms, such as establishment of personal relationship among developers, early and clear architectural assignment and boundary spanner.

Informal communication among developers was gradually introduced as personal relationship between developers was built. One developer from Team A1 said, "*We actually share everything and work even more together. Whenever we (internal people) talk together, before we met [Team A2], we always think they are so creepy… But now it has changed quite a lot by meeting them and seeing them just like us. It is a big gain for us*." Besides, architectural assignment helped both teams limited the area need to be coordinated. The contract-based division of work was based on the overall architecture of the whole system. Therefore, it reduced the number of common parts between two teams.

The project leader in Team A1 was considered as an unofficial boundary spanner and also an effective coordination mechanism. For an internal team member, boundary spanners were seen as a closest position to external stakeholders and have influence on them. A developer in Team A1 said, *"I believe that the project owner trust us deeply. They trust so much that letting [Project leader Team 1] as an assumption part of technical insight. If [Team 1] exploded and disappear the customer will have problem. I don't think [Team 1] is easy to replace at this time"*. For a customer, boundary spanners were seen as a source of information and also a target for giving feedback and negotiating tasks.

*B. **Findings on Case B***

B.1 Collaboration policy

In case B, collaboration policy refers to **different expectation on task responsibility** on borderline tasks. The team's divergent expectation led to *extra effort on team coordination and conflict solving*. Team B2 and B3 pursued a relatively high level of autonomy in collaborating with other development centers. The divergence between Team B2 and other teams was caused by task dependency and unclear assignment of responsibility at team level. As mentioned by a program manager in Team B2, each team often had a special competition on trying to not to take more tasks: "*We had so much to do that we tried to push the tasks in the borderline between team. We tried to push it to other team.*"

The given reason was that Team B2 wanted to focus on their modules and not to spread through many parts of the system. The coordination occurred in these borderline components and conflicts also appeared here. The program manager from team B2 mentioned that: "*When there is a problem with the exchanged component, the other team can work around or we can do to fix the issues... At the moment we have more problems with who is going to do what*." This question was rather a strategic question that is resolved at management level: *"… normally, it (the conflicted tasks) is a negotiation between teams by ourselves. But if we don't have the agreement, we need someone to make the prioritization."*

As mentioned by the program manager in Team B2, most of his work time spent on coordinating inter-team activities, in which, solving the conflict related to tasks, work process and temporal dependency are the most time consuming. It seemed that a clear responsibility and ownership of a component could facilitate team coordination and helped to avoid such conflicts.

B.2 Team organization

There was no significant problem related to communication across team in Case B. The communication occurred much more frequent on management level rather than it did in developer level. At the time of this study, Team B1 and B2 were putting a lot effort on enhancing the code commit structure. In general, the branching system was organized by the organizational structure. However, allowing the development of branches for each developer led to a complex branch system. When the integration was not frequently performed, the different branching models across sites might be problematic. A developer from Team B2 experienced extra coordination effort when merging branches from Team B1. The habits on working with Git facilitated the development of shared understanding among developers between Team B1 and B2. However the technical integration with a team outside Norway was more difficult. The program manager from Team B2 mentioned the problem: "*So when the developers check in (to the main branch), should they be free to check in any time or what kind of testing we need to do before checking in? We are adjusting this process.*"

B.3 Engineering process

Recalling challenges with incompatible development processes, **difference on development process** caused an unexpected *long duration of knowledge transfer.* When acquired by X and started collaborating with Team B1, Team B2 and B3 had a difficult time on working with the US site. Although the acquisition process had gone through a process compatibility assessment to support the integration of the new development center to the cooperation, there were a lot of challenges in the beginning.

Firstly, developers from Team B2 and B3 had to adapt to the new introduced engineering process, tools and practices. A developer from Team B2 mentioned: *"… when we were acquired by [Cooperation B] is a big difficulty for us to* *understand what they are talking about, what is the terminology, what is [Work package name] is. [Work package name] is not the way we are working before. We did have things in Git and exchanged like connection to Git, not sending the [work package name] …*" As mentioned by the program manager from Team B2, some parts of the old development process still remains due to the given autonomy of the development center.

Secondly, Team B2 and B3 experienced that the **different focus on quality and productivity** led to *difficulty on coordinating technical tasks*. Team B1, and the whole cooperation had a high focus on quality of software product, since their work serves for global market with a large amount of customers. Team B2 and B3 were used to work with a small set of local customer, hence focusing more on finishing on time and customization. The different focus led to different development and testing approaches between the two teams. The program manager from Team B2 said: "*The new focus on quality was very difficult… We had to spend more resources to have better quality in this case. Earlier we had a shorter last LOC developed to the release date. Now we have a longer time and an established test process. For [B2, B3's old company] just when the last line compiled we send the package to the customer and see their feedback*."

B.4 Development practices

Issues with different coding style, naming convention and error handling were automatically checked by tools before integrating the check-in source code into a main branch. Therefore, most of variation tool adoption and practices were mitigated in this case.

Variations on **test automation** among team B1, B2 and B3 introduced *the mistrust between teams.* The automation testing coverage rates also varied among teams. A typical development team was encouraged to increase the coverage rate up to 80%. A developer from Team B3 complained about the quality of code from other teams: "*… we also work with a team in Sri Lanka. There is a huge difference in quality between the two, but quality levels will converge next year. They will implement more automation test …*". While Team B1 focused on developing automated tests, Team B2 did not consider high coverage of test automation as an indication of high code quality. As Team B1 and B2 worked relatively close to each other, this issue had been extensively discussed among program managers.

B.5 Effective coordination practices

We observed the effective adoption of many coordination mechanisms, given in Table 4. All teams followed the same global development process, which consists frequent deliveries, synchronization of milestones and a consistent set of collaboration tools. On the collaboration approaches, all program managers didn't use desk phone but use a unified communication platform. A developer in Team B2 mentioned that: "*… for daily meeting it is like people seat in their own office place with their headset. I have my colleague who seats behind me. Very often we participate in the same meeting back-to-back and talk to other locations. I would say it is life safety for us to do collaboration with other locations.*" These approaches helped to form a similar organizational culture and common process between the teams.

Relationship building was established by having staff exchange among locations. The program manager from Team B2 mentioned that they have some short-term visit every year. A developer from Team B2 said, "*Researcher: do you have some informal talks and social activities with other teams? DevB2: when I was in Oslo, I spent two evenings with dinners. We had a Christmas party. A lot of us are friends outside work"*

A board of program managers acted as official boundary spanners that handle cross-team collaboration issues. The program manager from B2 mentioned that 50 – 80% of his work time was spent on coordinating inter-team activities, including communication of task status and schedule, conflict resolution and management of dependencies among locations. The specialization of the board as a coordination mechanism reduces the workload on communication in developer level.

V. DISCUSSION

We have described, in Section IV, the results from data collected on two cases that faced challenges in coordinating technical tasks. We now discuss the cases in light of our RQs.

RQ1: What are attributes of the organizational boundary that introduce challenges on team coordination in GSD?

On the collaboration policy, we observed a direct impact of high-level governance policy on coordination of technical tasks. Developers often collaborate in a multiple dimensional context, where not only technical dependency matters, but also resource and strategic dependencies does. In case A, customer's collaboration policy enlarges the organizational distance between two firms. In case B, high-level management policy clashed on autonomy of each development center. The misalignment of expectation and commitment among teams about responsibility of their partners were not clear due to the large organizational distances. In both case, the collaboration policy inhibited synchronization activities. This is a well-known problem and a common advice is to address who-do-what question in collaborating developers from other locations [15, 23, 28]. In our cases, the alignment of collaboration policy with coordination needs is seems to be equally important and need to be addressed at first.

On the team organization, alignment of communication structure and code submission structure is a challenge in both cases. This observation is surprising as team organization issues still appear in a small project with relatively little coordination needs. It seems that working on branches and committing code in a shared repository require a large amount of coordination. Previous literature often highlights the importance of supporting communication between developers who work on the same tasks [4, 11, 15]. Our case reveals that in a fine-grained level, the communication structure among developers should be aligned with their code commit structure.

On the engineering process, we found that differences on development methodology, such as sprint iterations and team's prioritized focus hinder frequent team communication and task synchronization. Even in the same organization, the coordination issues created by procedure differences should be expected. Paasivaara et al. reported that synchronization of milestones could help to coordinate two different development processes [29]. However, our observation suggests that there should be at least some common parts of processes. Depending on the coordination needs, this procedural commonality should allow synchronization activities across teams without extra resources and efforts.

On the development practices, we confirmed that the variation on coding and testing practices introduce the negative perception on the work of others and hinders the establishment of common understanding [19, 20]. Differences on testing and coding practices can introduce difficulty on establishing common technical understanding. In our case, we observed coordination problems in a homogeneous development and collaboration infrastructure. In case A, some features in TFS could reduce the needs of communication if they were adopted in both sites. In case B, synchronization of source code in both Git and TFS introduced some amount of unnecessary extra works. The coordination problem is more severe in case A as the organizational boundary is larger than that in Case B. This aligns with findings by De Souza and his colleagues, which highlights the importance of adoption of collaboration tool across geographical boundaries [15].

***RQ2: What are the coordination mechanisms that help to migtigate organizational boundary*?**

In this section, we discuss the effectiveness of some common mentioned effective coordination practices in previous studies [25, 29, 30, 31]. In Table 4, the coordination mechanism is either effectively used (marked by +), or not effectively adopted (marked by -) or not used (marked by O).

Synchronization of main milestones were observed in both cases. In case A, the duration between two continuous common milestones was too long to sufficiently exchange information and intermediates. Attempts to coordinate in a higher frequency were hindered by different deadlines and priorities of each team. Therefore, the effectiveness of this mechanism depends much on planning common milestones among teams without affecting each team 's priorities.

Table 4: Team coordination mechanism

Team coordination mechanism	A	B
Establishment of peer to peer links	O	O
Frequent deliveries	O	+
Synchronization of milestones	-	+
Shared management and development tool	-	+
Early relationship building	+	+
Early architectural assignment	+	+
Boundary spanner	+	+

Frequent deliveries were successfully adopted in Case B, as developers in all teams worked on daily build. This practice is very important for avoiding divergence on codebase and accumulation of technical debts. However, it was not applicable in Case A since it would require a global development processes for both partners.

Establishment of peer-to-peer links: we didn't notice the establishment of developer-to-developer links. The combination of manager-to-manager and manager-to-developer contact is a common pattern in both cases. Inter-team awareness and the feeling of teamness can be improved by facilitating developer-to-developer communication.

Early relationship building: our data showed that knowing other team member in person is especially important for coordination to succeed. Early face-to-face meetings and some informal talks should be done to facilitate later electronic communication. However, establishment of social relationship is initially limited by the control policy in case A.

Early architectural assignment: in both cases, task breakdown structure was created to support team organization structures. The source code branch and repository structure is balanced between team structures and product structures. For complex projects like Case B, this is compulsory to reduce complexity and inter-dependency between teams and tasks.

Shared management and development tools: were established in both cases. However, the full advantages of this are disabled in case A, due to the variety of development practices. Hence, we suggested that common tools set should be accompanied with common usage practices in order to fully utilize the advanced development environments.

Boundary spanners played an important role in supporting team coordination in both cases. The team leader (in case A) and the program manager (in case B) were the ones who know about business domain, technical insight of the product and established a stable relationship with the customer teams. Therefore, they no only facilitate team communication, but also to transfer knowledge and to contact customers.

VI. CONCLUSION

Distances do not need to be global to matter, and not all problems in GSD are caused by physical distances. From two exploratory case studies, we observed several team challenges with coordinating across organizational boundaries and a set of effective coordination practices.

To response to "What are attributes of the organizational boundary that introduce challenges on team coordination in GSD?", we found that collaboration policy, organization structure, engineering process, development practice and tool adoption negatively influence technical coordination among project stakeholders. There is variation on specific attributes of these organizational boundaries and the extent of their influence on team coordination, depending on the coordination needs, project size, and type of organizational relationship. For practitioners, our study showed that firstly, project managers need to be aware of the level of development process variation among organizations for the project coordination between organizations to be successful. Secondly, when determining high-level of collaboration policy, the indirect influence on team coordination at technical level should be considered. Lastly, after obtaining shared development environments, a guideline for shared standard and tool adoption practices should be established.

To response to “What are the coordination mechanisms that help to mitigate organizational boundary?”, we confirmed the effectiveness of practices, such as early relationship building, early architectural assignment, and knowledge broker in both cases. The large organizational boundary inhibits the effectiveness of synchronization activities, frequent deliveries, peer-to-peer contact, and shared management and development tools. For practitioners, our study confirmed that relying on a single coordination mechanism is not sufficient to support team collaboration beyond organizational boundary. The formal processes, standards and guideline should be established among organizations to facilitate the transparent collaboration process in both management and operation levels. Most importantly, the role of boundary spanners in supporting team coordination need to be recognized and supported.

In future works, we will continue to explore the other aspects of organizational boundary in GSD. Further studies in this direction can focus on identification of some measures of organizational distance and quantitatively investigate their relationship to team coordination.

References

[1] P. J. Ågerfalk, B. Fitzgerald, H. Holmström Olsson, and E. Ó Conchúir, "Benefits of global software development: The known and unknown", vol. 5007 LNCS, pp. 1-9, 2008.

[2] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination", Workshop on the Future of Software Engineering (FoSE), Washington, DC, USA, pp. 188-198, 2007.

[3] S. Nidumolu, "The effect of coordination and uncertainty on software project performance: Residual performance risk as an intervening variable", *Information Systems Research*, vol. 6, pp. 191-219, 1995.

[4] R. E. Kraut, L. A. Streeter, “Coordination in Software Development”, *Communaction of the ACM*, vol. 38(3), pp. 69-8, 1995.

[5] J. D. Herbsleb, A. Mockus, “An Empirical Study of Speed and Communication in Globally Distributed Software Development”, *Transaction on Software Engineering,* vol. 29(6), pp. 481-494, 2003.

[6] M. Cataldo, J. D. Herbsleb, “Factors leading to integration failures in global feature-oriented development: an empirical analysis”, ICSE 2011, Honolulu, Hawaii, USA, pp. 161-170, 2011.

[7] D. Šmite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical evidence in global software engineering: A systematic review", *Empirical Software Engineering*, vol. 15, pp. 91-118, 2010.

[8] J. A. Espinosa, W. DeLone, and G. Lee, "Global boundaries, task processes and IS project success: A field study", *Information Technology and People*, vol. 19, pp. 345-370, 2006.

[9] A. Nguyen Duc, D. S. Cruzes, R. Conradi, “Dispersion, coordination and performance in global software teams: a systematic review”, ESEM 2012, Lund, Sweden, pp. 129-138, 2012

[10] B. Curtis, H. Krasner,and N. Iscoe, “A Field Study of the Software Design Process for Large Systems,” *Communications of the ACM*, vol 31, no. 11, pp. 1268-1287, 1988

[11] N. Nagappan, B. Murphy, V. R. Basili, “The influence of organizational structure on software quality: an empirical case study”, ICSE 2008, Leipzig, Germany, pp. 521-530, 2008.

[12] D. Damian, I. Kwan, “Requirements-Driven Collaboration: Leveraging the Invisible Relationships between Requirements and People”, *Collaborative Software Engineering*, Springer Berlin Heidelberg: pp. 57-76, 2010.

[13] J. M. Bhat, M. Gupta, G. N. Murthy, “Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing”, *IEEE Software*, vol. 23(5), pp. 38-44, 2006.

[14] D. Šmite, C. Wohlin, Z. Galviņa, R. Prikladnicki “An Empirically Based Terminology and Taxonomy for Global Software Engineering”, *Journal of Empirical Software Engineering*, 2012

[15] C. R. B. de Souza, J. M. Reis Costa, M. Cataldo, “Analyzing the scalability of coordination requirements of a distributed software project”, *Journal of the Brazilian Computer Society* vol. 18(3), pp. 201-211, 2012.

[16] P. J. Hinds, C. McGrath, “Structures that work: social structure, work structure and coordination ease in geographically distributed teams”, CSCW 2006, Banff, Alberta, Canada, pp. 343-352, 2006.

[17] R. Prikladnicki, J. L. N. Audy, “Managing Global Software Engineering: A Comparative Analysis of Offshore Outsourcing and the Internal Offshoring of Software Development” *IS Management,* vol. 29(3), pp. 216-232, 2012.

[18] R. Kommeren, P. Parviainen, “Philips experiences in global distributed software development”, *Empirical Software Engineering*, vol. 12(6), pp. 647-660, 2007.

[19] D. Šmite, N. B. Moe, and R. Torkar, "Pitfalls in remote team coordination: Lessons learned from a case study", LNCS, vol. 5089, pp. 345-359, 2008.

[20] N. B. Moe, D. Šmite, “Understanding a lack of trust in Global Software Teams: a multiple-case study”, *Software Process: Improvement and Practice* vol. 13(3), pp. 217-231, 2008.

[21] T. W. Malone, K. Crowston, “What is Coordination Theory and How Can It Help Design Cooperative Work Systems?”, CSCW1990, Los Angeles, USA, pp. 357-370, 1990.

[22] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb, "Team knowledge and coordination in geographically distributed software development", *Journal of Management Information Systems*, vol. 24, pp. 135-169, 2007

[23] D. Damian, S. Marczak, and I. Kwan, “Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centered Social Networks”, Conference on Requirements Engineering, New Delhi, India, pp. 59-68, 2007.

[24] D. E. Damian, D. Zowghi, “RE challenges in multi-site software development organisations”, *Requirement Engineering* vol. 8(3), pp. 149-160, 2003.

[25] M. Cataldo, M. Bass, J. D. Herbsleb, L. Bass, “On Coordination Mechanisms in Global Software Development”, ICGSE 2007, Munich, Germany, pp. 71-80, 2007.

[26] J. A. Espinosa, J. N. Cummings, and C. Pickering, "Time Separation, Coordination, and Performance in Technical Teams", *IEEE Transactions on Engineering Management*, vol 59, pp. 91-103, 2011.

[27] E. Salas, D. E. Sims, C. S. Burke, “Is there a ‘‘big five’’ in teamwork?”, *Small Group Research* vol. 36(5), pp. 555–599, 2005.

[28] D. Damian, L. Izquierdo, J. Singer, I. Kwan, “Awareness in the Wild: Why Communication Breakdowns Occur”, ICGSE 2007, Munich, Germany, pp. 81-90, 2007.

[29] M. Paasivaara and C. Lassenius, "Collaboration practices in global inter-organizational software development projects", *Software Process Improvement and Practice*, vol. 8, pp. 183-199, 2003.

[30] J. Whitehead, I. Mistrík, “Collaborative Software Engineering: Concepts and Techniques”, *Collaborative Software Engineering,* Springer Berlin Heidelberg, pp. 1-30, 2010.

[31] B. Curtis, H. Krasner and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, Vol. 31, No. 11, 1988, pp. 1268-1287.

[32] R. K. Yin, *Case Study Research: Design and Methods,* 3rd edition, vol. 5. Sage Publications: Thousand Oaks, CA, 2003.

[33] A. Strauss, J. Corbin, *Basics of Qualitative Research Techniques and Procedures for Developing Grounded Theory*. 2nd edition, 1998