UNIVERSITY OF HELSINKI

https://helda.helsinki.fi

A Canvas for Establishing Global Software Development Collaborations

Smirnova, Inna; Münch, Jürgen; Stupperich, Michael

Dregvaite, Giedre; Damasevicius, Robertas

2014

http://hdl.handle.net/10138/153354

Smirnova, I, Münch, J & Stupperich, M 2014, A Canvas for Establishing Global Software Development Collaborations. in G Dregvaite & R Damasevicius (eds), Information and Software Technologies : 20th International Conference, ICIST 2014, Druskininkai, Lithuania, October 9-10, 2014. Proceedings. Communications in Computer and Information Science, vol. 465, Springer International Publishing AG, Cham, pp. 73-93, International Conference on Information and Software Technologies, Druskininkai, Lithuania, 09/10/2014. https://doi.org/10.1007/978-3-319-11958-8_7

Downloaded from Helda, University of Helsinki institutional repository. https://helda.helsinki.fi
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.
Please cite the original version.

This is an author-generated version.

The final publication is available at link.springer.com

Bibliographic information:

Inna Smirnova, Jürgen Münch, Michael Stupperich. A Canvas for Establishing Global Software Development Collaborations. In Proceedings of the 20th International Conference on Information and Software Technologies (ICIST 2014), CCIS. Springer-Verlag, 2014.

A Canvas for Establishing Global Software Development Collaborations

Inna Smirnova[1], Jürgen Münch[1], and Michael Stupperich[2]

[1]Department of Computer Science
University of Helsinki
Helsinki, Finland
[2]Daimler Research & Development Ulm
Daimler
Ulm, Deutschland
inna.smirnova@helsinki.fi,juergen.muench@cs.helsinki.fi,michael.stupperich@daimler.com

Abstract. There is an increasing need and interest for organizations to collaborate with internal and external partners on a global scale for creating software-based products and services. Potential risks and different strategies need to be addressed when setting up such collaborations. Aspects such as cultural and social features, coordination, infrastructure, organizational change processes, or communication issues need to be considered. Although there are already experiences available with respect to setting up global collaborations, they mainly focus on specific areas. It is difficult for companies to quickly assess if they have considered all relevant aspects. An overall aid that guides companies in systematically setting up global collaborations is widely missing. In this paper we present a study based on the snowballing method as a systematic approach to literature review. Based on this literature review and inputs from industry we investigated what aspects and practices need to be considered when establishing global software development collaborations and how to prioritize them. Based on that we created activity roadmaps that aggregate existing experiences. Reported experiences were structured into nine main aspects each containing extracted successful practices for setting up global software development collaborations. As a result we came up with an initial version of a canvas that is proposed as guidance for companies for setting up global collaborations in the software development domain.

Keywords: Global software development, global collaborations, activity roadmaps

1 Introduction

Today's era of globalization already affected and is still in the process of influencing many fields. Particularly software development has a great impact [8]. Nowadays the phenomenon of Global Software Development (GSD) with process' distribution all over the world is seen as a normal way of doing things [8].

A major benefit that companies expect from global distribution and joint international collaborations is the access to low-cost resources, particularly a large remote labor pool with diverse expertise and working skills that could scale up development teams fast and might potentially lead to financial savings - *"Hourly onshore costs are typically three to four times higher than offshore rates"* [11]. Other reasons could include potential expectations on speeding up the time for product development, shortage of onsite resources, freeing up local resources for new projects, overall optimization of resources, accessing new huge markets, foreign know-how and technologies, and gaining valuable market competitiveness [12–17].

However, along with potential benefits GSD brings many new challenges, especially regarding communication, culture, coordination and project management areas [18]. For practitioners who do not yet have enough experience with setting up GSD collaborations it is necessary to be aware of the challenges and risks, different strategies and aspects along with existing practices and experiences. Such knowledge might help to reduce possible future negative effects leading to GSD projects' failures such as cost overruns, exceeding timeframes, low product quality, and overall decreased customer satisfaction [6, 8, 19]. As GSD collaborations suffer from geographical, temporal, socio-cultural distance, the organizations should adapt their current software development practices in order to benefit and gain competitive advantage in new Global Software Engineering conditions [20].

Although there are already many scientific papers and experiences available with respect to challenges, risks, mitigation advice, best actions and practices regarding establishing GSD collaborations, they are mainly focusing on specific areas such as examining trust or communication aspects [21–24]. Therefore the question how they can be used and integrated in an overall guide for setting up GSD collaborations still remains unclear. An overall holistic approach that synthesizes knowledge and guides companies in systematically setting up global collaborations for software-based products and services is widely missing [21].

To address the need for a holistic approach to setting up GSD collaborations, we performed a literature study and cooperated closely with a partner company from the industry side that operates in the automotive domain. Based on the results from the literature study and advice from the industrial partner we investigated what principal aspects and practices need to be considered when establishing global collaborations in the software development domain. Afterwards, we prioritized them and created activity roadmaps that aim at aggregating existing experiences that are relevant, credible and helpful for practitioners. Reported experiences were structured into nine main aspects. Each of them contains extracted practices for setting up GSD collaborations. Furthermore, we present the initial version of a worksheet, the so-called "Global canvas", that is proposed to be practically used as a guidance for companies intending to start global collaborations in the software development domain.

The original and new *contribution* of this study is the creation of a holistic "shopping list" of things to think of when establishing global collaborations

in the software development domain. The goal is to provide a worksheet that presents scientific results to industry in an effective way and helps to identify what needs to be considered when setting up global collaborations. The aim of this study is to come up with an initial proposal for such a holistic prescriptive worksheet that is driven and validated against the needs and requirements of our case company. A mature and detailed validation of the configuration of aspects and practices is out of the scope of this study and planned as future work.

The article is structured as follows. Section 2 gives an overview of existing research with a focus on establishing global collaborations. Section 3 explains the research method that was used for collecting data. Section 4 presents the results found through literature study and industry consultation. This section explains in detail what aspects and practices were discovered with respect to establishing GSD collaborations. The proposed worksheet "Global canvas" is described in section 5. Finally, section 6 discusses the conclusions, limitations of the study and the potential for future research.

2 Related Work

There already exist many studies that analyze globally distributed software development projects, new challenges and risks compared to traditional co-located development, risk-mitigation advice, practices and experiences. Most studies focus on presenting an overview of challenges and problems which might occur as an impact of the distance that brings global orientation to software development, or examine in detail a specific aspect of GSD collaborations. In contrast, our research is aimed at a synthesis of relevant aspects with the purpose of creating a guide for companies that want to set up global collaborations and require a holistic view of the relevant aspects that need to be considered.

Nurdiani et al. performed a systematic literature review among GSD research literature that resulted in a checklist of 48 GSD challenges and 42 mitigation recommendations [6]. Another systematic literature review was done by Verner et al., who reported the risks of GSD collaboration with some mitigation recommendations structured into 12 areas starting from vendor selection and requirements engineering and finishing with coordination and control areas [7]. Šmite et al. conducted a systematic literature review of GSD experiences and came up with seven most commonly discussed practices that are aimed at overcoming GSD problems [8]. Mettovaara et al. performed interviews at Nokia and Philips and identified 10 common problems and 11 success factors based on the experiences in interorganizational collaborations in the two studied companies [25]. However, all those studies are risk- and problem-oriented in the first place. Our study takes these findings into account. However, instead of identifying relevant risks, we aim at providing a constructive guide that contains helpful practices and a sequence of activities for setting up global collaborations.

3 Context and Research Method

This study was performed in collaboration with the automotive OEM "Daimler AG" that served as case company for this study. The respective business unit of the company that was the contact point for this study is intending to set-up a long-term, multi-national distributed global collaboration for software-based products in the automotive domain.

The company identified a set of aspects (such as collaboration structure, product structure, communication, infrastructure) that were seen as highly important when setting up global collaborations. The company also provided a proposal for a sequence in which these aspects should be considered. These aspects and the information about the sequence were elicited from project leaders and reflect their experience from leading global projects in the business units "Daimler Trucks" and "Daimler Buses". The elicitation was done at Daimler via interviews and through company-internal workshops with the project leaders. The interviews and workshops were conducted by the one co-author of the paper who works at Daimler. In addition, members of the case company attended several ICGSE (International Conference on Global Software Engineering) conferences and input from these conferences implicitly influenced the selection of the aspects.

The aspects provided by the case company are used in this study as means for structuring the areas with practices for setting up global collaborations in the software development domain. This was the main rationale for selecting the aspects. We used existing systematic literature reviews (SLRs) to make small adjustments to the list of aspects provided by the case company, especially with respect to their definition and naming.

The research of this study was performed as backward snowballing [1, 2, 47]. Snowballing as a research method for data collection was chosen as an instance of a systematic approach to literature review that helps to collect all the necessary literature without performing a full systematic literature review. The chosen topic of interest was originally very broad, so that we decided not to perform a full SLR, but to choose a different systematic approach. Snowballing was found to be suitable for the exploratory research we aimed at.

Based on Webster and Watson [1] as well as Wohlin [47] the starting point for the backward snowballing research approach is the analysis of main contributions to the topic. We have decided to analyze four key SLRs related to setting up GSD collaborations as a starting point [5–8]. These SLRs aggregate already existing knowledge with respect to topics such as risks, mitigation solutions, and strategies. Therefore, they were seen as a suitable starting point. The next step in the backward snowballing method according to Webster and Watson as well as Wohlin is to "go backward" by reviewing the citations in the papers that serve as the starting point with the goal to identify topic-relevant studies that need to be considered [1, 2, 47]. We have performed a review of the bibliographic reference lists of the four SLRs and selected studies that fit to the aspects defined by the case company. This snowballing step was iteratively performed up to four times depending on the suitability of the results found.

In order to identify more topic relevant literature, we additionally reviewed scientific papers from major GSD conferences, i.e, from the International Conference on Global Software Engineering (ICGSE, 2006-2013), and from the International Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD, 2007-2010).

After the search for relevant literature, the found set of papers was examined with respect to content relevance. As a result, a collection of primary studies was defined as a literature pool for our study. The next step of the research study represented data extraction from the found literature, followed by further analysis regarding the identification of strategies and practices that need to be addressed in global software development projects. Following Whittemore and Knafl [3], the gathered strategies and practices were grouped together in an integrative way, and prioritized and described as a guidance framework that aims at supporting practitioners in setting up global software development collaborations.

As an initial validation, the results were frequently presented to the key stakeholders in the case company and reviewed. Feedback was used to revise the worksheet "Global canvas" and create the final version presented in the article.

4 Results from the Literature Study

In the following section we present the results found in the literature study. The results are structured into nine main aspects that are proposed to be addressed by the companies while setting up global collaborations in the software development domain. Each aspect contains extracted success practices and experiences.

4.1 Strategy

"Global Software Engineering becomes part of the everyday business", many software companies today recognize the need for globalization [5]. Organizations that have the intention of transferring software development work into a global context, need to understand how to start global software development, the reasons, first steps and actions for setting up global collaborations [5, 27]. Thus the first important aspect that we identified in our study is *Strategy* that is aimed to be the initial step and help software companies to answer such questions as - Why do we collaborate globally? What are the benefits that global collaborations might bring to the business? How do we do the collaborations, according to what model and where to? This step represents the whole high level framework of establishing global software development collaborations and therefore cannot be avoided by the companies engaged in performing such collaborations.

We identified five principal practices that need to be considered at the very initial stage of global collaborations.

First of all, the organization needs to see distinctly what the goals and potential benefits of collaboration are [17]. Based on the investigations of Forbath et al. [17] the main benefits and therefore goals of doing global collaboration for

software-based products can be classified into three areas. The primary driver is financial savings. Development costs might lower due to access to a large labor pool in countries with lower wages or access to many resources needed for development and easily available for use at the offshore destination. Another potential could be the access to foreign know-how, expertise, new technologies that might not be obtainable onshore. Furthermore, the proximity to new offshore markets could be a driver for customization and localization of products that might lead to new customers and bigger revenues.

Next, after understanding the reasons for setting up global collaborations, the organization has to choose an appropriate collaboration model that suits a specific company context and goals. Based on the inputs from the industrial partner, we have identified and focused on three possible scenarios for launching GSD collaboration named Offshore outsourcing, Offshore insourcing and Innovative offshoring [5, 7, 19, 29]. Offshore outsourcing refers to consuming of resources and development services from an external 3rd party that is located in a different country, often representing client-subcontractor relationships [5]. Offshore insourcing means consuming of internal organizational resources that are located in a foreign country [5, 19, 29]. The company establishes, for instance, a foreign branch in a different country, in order to customize products for a dedicated market. The model of Innovative offshoring refers to consuming of innovative R&D services from the offshore partner company that is situated abroad. Those innovative services could, for instance, aim at improving the headquarter's product [5, 19, 29].

The following recommended steps that organizations should follow at the beginning of setting up global collaborations are the investigation of the foreign legal system regarding contract and IP laws; the selection of a suitable vendor with desired expertise, efficient capabilities and sufficient technological infrastructure; and the planning of financial budget for collaboration including possible risks and therefore hidden costs [7]. One of the main motivations for performing GSD collaborations by the company could be cost savings, therefore financial planning should be considered early on [7].

4.2 Collaboration Structure

In order to maximize the potential positive effects that global software development collaborations might promise and minimize possible negative risks, it is necessary to choose the right way for establishing collaborations, to determine the most advantageous form of dividing task distribution, responsibilities, peer-to-peer connections between involved sites. Our defined aspect *Collaboration structure* is dedicated to these questions of global collaborations. This aspect is aimed at determining the approach of development task allocation between locations based on collaboration goals, at creating roles and responsibilities along with the way of distributing them; at defining an organizational structure and peer-to-peer connections between sites [16, 20, 31–33, 46]. Clear understanding of work division, roles and responsibilities might help to decrease coordination and project management efforts when actual global software development takes

place. Therefore it is important to address this aspect already at the planning stage of setting up global collaborations.

Regarding the aspect of *Collaboration structure* we identified two main practices which need to be considered. Those are to define the approach to distributed process breakdown and task allocation and to determine and specify the organizational structure and peer-to-peer links between sites.

Based on Šmite's case study in a Latvian software company [16, 26], Nissen's case study report from an inter-organizational cooperation in telecommunications domain [31], and the research of Faiz et al. [32] we have identified three models of process-based task distribution between sites which are suitable for collaboration scenarios described in the aspect *Strategy*. The first model was considered as a typical outsourcing model where most of the intellectual work stays onsite and only actual software development tasks are transferred offshore. In this model, requirements creation, system analysis, design are done onsite, while coding and testing phases can be performed jointly with work division, for example, by modules [16, 31–33]. The main challenges of this kind of task distribution consist of troublesome system requirements clarification, system integration and bug fixing along with coordination and control efforts. The second model is considered to be more suitable for the Offshore insourcing collaboration model where, in contrast to the first model, requirements creation, system analysis, design are performed as joint activity between collaboration partners [16, 31–33]. Such task distribution might suit product customization goals and help to create better common understanding and social ties between locations. However, this model requires good domain business knowledge from the vendor site that might be effortful to create. The last third model was considered to be used for the Innovative offshoring scenario that represents close collaboration between locations where the offshore partner performs most of the intellectual and implementation work such as R&D, requirements creation, system analysis and design, actual implementation [16, 31–33]. This model might be based on a prototyping strategy that refers to creation of innovative prototypes by the offshore site. Later the developed prototype can be used onsite as a base for building new functionality or a complete product on top.

Another practice that was aimed to be a part of the *Collaboration structure* aspect is to specify organizational structure and peer-to-peer links between collaboration sites. The main aim of the organizational structure is to clearly list roles, responsibilities and to draw communication channels between locations involved in collaboration at management, project and team levels. The examples on what roles and sites' connections the global collaboration can include were presented in the studies of Braun [34] and Faiz et al. [32]. Defined and documented organizational structure is aimed to achieve easier coordination and control, to make the information flow more transparent and traceable. It helps to make communication between parties less challenging and reach project-related common understanding faster, which eventually might reduce some efforts and investments needed for setting up distributed collaboration [24, 32–34].

4.3 Product Structure

The development process breakdown and the following task distribution, which were considered to be identified in the aspect of *Collaboration strategy*, affect the definition of the product architecture. The aspect *Product structure* addresses how the product architecture could be adapted for global software development compared to co-located development, the product ownership boundaries between locations, and how modifications to the product part at one location can affect work at other locations. Therefore it has an impact on work division between GSD teams at different locations. Clearly identified work division and product ownership boundaries between collaboration sites are expected to improve communication, reduce project coordination efforts, and help to avoid rework and duplications [7, 20, 26, 35, 46]. Thus the aspect *Product structure* is an important step for companies and needs to be already considered at the preparation and planning stages of global collaborations. It is strongly connected with the *Collaboration structure* aspect, especially with detailed definition of organizational structure, roles and responsibilities.

We have distinguished the following practices that need to be considered by the organizations. Those are to determine the product architecture, to specify product ownership between locations and to define product-based work distribution among GSD teams at different locations.

Depending on the collaboration scenarios we have identified in the aspect *Strategy*, the possible way for a *Product structure* definition might differ greatly. Referring to the Offshore outsourcing model, the strategy with one core product and full onsite ownership where the offshore partner is responsible for allocated tasks in the product development lifecycle is considered to be most suitable [36, 41, 45]. In contrast, the model of Offshore insourcing can be based on a product line architecture with variants which are built on top of a core product. The ownership of the core product can be kept onsite, while the offshore site might be responsible for the full development of one of the variants from the product line [9]. Such a variant can be customized and market-specific. In this model the offshore site has a great responsibility for the whole product variant, and thereby global collaboration might have a form of peer-to-peer partnership [26, 36]. The Innovative offshoring scenario is intended for the development of new innovative products or prototypes by the offshore site with different possibilities of product ownership boundaries depending on the initial collaboration goals and model.

Considering the approach to detailed software system architecture and following development work distribution between locations, our main suggestions are to use modular architecture and decoupling that allow having well-defined software work packages which can be distributed between different locations based on available resources and expertise. Salger [35] demonstrates an example structure of the software work package based on the experience at Capgemini sd&m. The software work package might consist of the following parts: *Software requirement specifications* describing use cases, user interface, domain objects, and specifications of functional test cases; *Design artefacts* including an external technical view on a software module and internal high level design view;

Project management artefacts containing a list of work units described in earlier parts, schedules, budget, definition of quality objectives and work acceptance criteria [35]. Such a well-defined work package structure promises to ease work transfer between locations and to achieve low dependencies between locations during actual implementation work [7, 8, 12, 14, 20, 26, 36–41]. This way it might be possible to reduce communication and coordination needs between different locations. However, system integration could become a troublesome bottleneck.

4.4 Coordination

Global software development brings geographical distance and cultural diversity into the software development process compared to co-located development. Thus globally distributed software development teams need to be effectively managed and controlled in order to complete software projects successfully, to be inside a financial and technological budget, and to use available resources and capabilities beneficially for the collaboration goals [20]. Therefore competent coordination, communication and control procedures should be attentively planned and later performed by companies on an everyday basis. Coordination can be seen as work integration in a way that each involved unit contributes to the completion of the overall task [12, 40]. Coordination procedures describe how collaboration sites communicate between each other in order to complete commonly defined tasks and to achieve collaboration goals [12, 40]. The *Coordination* aspect represents the set of activities that aim at managing dependencies within the global software development project workflow, so the work can be completed faster and more effectively. According to a study by Nguyen-Duc and Cruzes [39] such dependencies in a GSD context might include technical views such as system integration, configuration change management; temporal issues such as synchronization of schedules, deliveries between sites; software development process organization; resource distribution such as infrastructure, budget, or development tools. This aspect promises to be very important for setting up global collaborations, because coordination and project management efforts might become a cause of project hidden costs. Therefore coordination challenges need to be minimized by the organizations starting from a planning phase. For instance, differences in organizational policies, lack of common processes, variation of coding and testing standards between collaboration sites might affect coordination and project management efforts, and also might lead to insufficient end product quality and additional costs [39].

Within the aspect of *Coordination* we have specified two main practices that need to be addressed by organizations, i.e., Project management and Project control.

Project management aims at planning and organizing software development project-related activities in such a way that they lead to successful work completion. Such activities might include creating shared synchronized understanding of main milestones between collaboration sites, concrete tasks to perform, deliveries schedules, project budget constraints, peer-to-peer contact links between

collaboration sites, and managing the whole project execution [39]. Project management can be seen as a mechanism that integrates software, human, and economic relations in order to use existing technology, resources, time, capabilities in the most productive and effective way [43]. Project control procedures refer to the process of monitoring work status and ensuring that the work process goes in the right direction according to the planned budget, timeframes and quality expectations [12]. Among project control procedures, a formal reporting structure concerning updates, changes and escalation path can be seen to play an important role for achieving visibility of software project status and work progress, for detecting project bottlenecks and work conflict situations and reacting to them early on [7, 10, 20, 39].

Different collaboration scenarios might have different coordination mechanisms working better in particular situations. Based on a case study by Hossain et al. in an Australian-Malaysian cooperation [44] we have identified different possible ways of performing coordination processes and discovered how they can suit different global collaboration models. For instance, the Offshore outsourcing model might require a high degree of defined standard policies, direct supervision and centralized project organization for the offshore team from the headquarter company. For the Offshore insourcing and the Innovative offshoring scenarios the software development work might be managed better with a high degree of mutual adjustment when collaboration is based on building trust and social relationships between sites, thus, many software project activities and decisions are often performed jointly [7, 9, 12, 20, 23–26, 29, 33, 35–40, 43, 44].

4.5 Development Process

As soon as the collaboration model, the process breakdown, the product structure, the task distribution, the coordination and the control mechanisms are identified, a solid foundation for defining and/or customizing a development process is laid. The aspect *Development process* aims at defining the model for software development activities between the collaboration sites. Based on experience from the authors it is recommended to mainly define the processes at the interfaces between the collaborating sites and not to aim at unification of all processes at all sites, especially when the sites belong to different organizations. Typically the specific local characteristics at each site make it hard to prescribe unique internal processes at all sites. Defining a software development process model also helps to clarify roles and responsibilities, the level of independency between sites, and the product quality expectations. Frictions such as role confusions can be avoided. Moreover, a development process can affect coordination and communication efforts, infrastructure needs, change management mechanisms and system integration efforts. In the Offshore outsourcing scenario or the Innovative offshoring scenario the collaboration sites might keep their own development processes if these processes are already established and well-working [20, 24]. However, it is essential to synchronize project milestones and schedules for important software product deliveries in order to achieve project transparency, continuous frequent integrity and early feedback on the quality of

developed software [20, 24]. In the case of the Offshore insourcing scenario that has peer-to-peer close partnership orientation, it could be suitable to establish standardized guidelines for a common software development process and tools between sites. This promises to create a joint corporate level of work standards [7, 37, 43].

4.6 Communication

Communication can be seen as the exchange of information that helps to reach a common shared understanding between remote sites, including information and knowledge sharing [12, 40]. The aspect *Communication* addresses all kinds of communication activities between the different development sites. As global software development collaborations are to a large degree human-based, communication becomes crucial and needs to be considered early on, i.e., starting from the planning and negotiating phases of the collaboration till its full establishment and maintenance. Numerous studies based on industrial project investigations report the importance of communication in the GSD context and usually come to the conclusion that it is the number-one problem. Studies focusing on communication include the study by Mettovaara et al. in Nokia and Philips [25], the case study by Leszak and Meier on embedded product development in the telecommunications domain in Alcatel-Lucent between Germany and China [36], the study by Paasivaara and Lassenius based on interviews in 8 global software projects distributed across Europe, North America and Asia [24], and the study by Oshri et al. at LeCroy (Switzerland and USA), SAP (India and Germany), and Baan (India and The Netherlands) [30].

Geographical distances between teams often cause difficulties with using traditional communication paths such as face-to-face meetings and informal communication. Remote sites often need to rely on asynchronous ways of communication (tools such as E-mails, chats, blogs) or phone/video conferences that bring certain risks like misunderstandings, delays, unnecessary work, reduced trust, or absence of team spirit and partnership feeling. Those challenges might result in additional project costs, customer dissatisfaction, and barriers to maintaining long-time global collaborations. Thus companies need to consider and plan communicational strategy early on in the first stages of collaborations.

With the aspect *Communication* we have identified five practices that aim at building a successful communication strategy when establishing GSD collaborations.

Communication protocol aims at identifying who is supposed to communicate with whom within the company such as, communication channels, interface points among collaboration teams and team members, sufficient frequency of communication, information exchange paths, official corporate language [24, 42]. A detailed description of a communication protocol based on the organizational structure should be documented and distributed among team members at all collaboration sites. It is expected to create awareness of team members from whom they will get work inputs and to whom they need to distribute work output re-

sults. This way the software development project might gain more transparency and traceability [24, 42].

Team awareness channels aim at making collaboration team members become more familiar with remote colleagues and their skills, their expertise and availability, as well as their project activities and work status. It is expected that teamness and trust between remote collaboration sites help to achieve project visibility and to reduce delays for finding the right person to contact in case of some questions [38]. Team awareness can be supported, for instance, by organizational charts, project websites, or shared calendars [23, 24].

Social relationships between collaboration sites represent the result of all communication activities and efforts. Relationship building is a long process and therefore needs to be seen as a constant activity when doing global collaborations. Face-to-face meetings are a highly efficient way for building social interrelations between collaboration sites, thereby frequent visits and staff exchanges between sites are necessary. Even though face-to-face visits might cause additional investment and time, they need to be present especially in the first phases of global collaborations [7, 12, 14, 23–25, 33, 36, 40, 44].

Rich *communication tools* aim at supporting all the above-mentioned communication practices. Collaboration sites are often located at remote places, so tools often provide the only way for software development teams to get connected. Thus a variety of different communication tools such as web meetings, phones, e-mails and mailing lists, chats, file transfer tools, groupware and shared services tools should be provided by organizations [28, 37, 42].

All communication activities are expected to make global collaborations more peer-to-peer partnership-oriented. Thus it helps to build up a *common knowledge base* between collaboration sites. This promises to create the "organizational memory", shared collective understanding of the domain knowledge, technology and business needs. It accumulates the experience of a collaboration in a specific organizational context and might help collaboration sites to learn and improve their way of working together [28, 37, 42].

4.7 Social Aspects

Global distribution of software development implies that individuals are usually not only geographically dispersed but also culturally. Thus the process of socialization and cultural integration is important when setting up global collaborations. With *Social aspects* we refer to the process through which team members gain the knowledge on behavioral and communication norms, attitudes, cultural and social patterns of each other in order to work together in cooperation [30]. The process of socialization and getting to know the partners is expected to create a mutual vision on the collaboration and specific project goals, to create the understanding of remote partners' way of working and behavior, and to make global collaboration function successfully and beneficially for all the sites. *"When there is a win-win situation the motivation is usually high and the chances of success get better"* [25]. Socio-cultural distance might bring many challenges and negative effects into the collaboration process such

as difficulties and inability of sites to communicate, unawareness of remote colleagues' qualification, unwillingness to exchange information, conflicts of tasks interpretation and unsuccessful end results. These challenges might have a great negative impact on the collaboration process between sites and the quality of the end product. Therefore, organizations need to consider social aspects and stimulate socio-cultural integrity between collaboration sites [7].

Within *Social aspects* we have specified two categories that need to be considered - *Trust* and *Cultural understanding*.

Trust is considered to be one of the keys for establishing effective, productive, reliable, and longitude collaborative social relationships between teams in global software development contexts [22, 23, 40]. Trust can be defined as the willingness of individuals to cooperate with others based on the belief that partners are reliable, competent and will do actions which are beneficial for the cooperation rather than for individual purposes [40]. *"Trust is a pre-requisite for globally distributed software development"* [40]. Trust promises to create the ability of remote collaboration sites to work together, and to build up the feeling of teamness. Trust stimulates the willingness of sites to communicate and work towards the completion of shared project goals - not "we and you" relations but "us" [22, 29]. Lack of trust might lead to a situation of non-cooperation, social conflicts, absence of information exchange, overall decrease in productivity and end product quality, and eventually to job dissatisfaction among employees [29]. Thus a lot of efforts are needed to be done by organizations in order to build trust between collaboration sites. Such efforts, for instance, are face-to-face visits, frequent remote communication via a rich variety of tools, staff exchanges, socio-cultural trainings, social activities. Trust needs to be built and maintained through the whole partnership history from the first collaboration stages till its end [11, 22, 25, 29, 33, 40].

Cultural understanding represents shared norms and beliefs which are historically situated and followed by people belonging to a concrete society [4]. In the context of global collaborations socio-cultural diversity among sites might be interpreted as a facilitator for promoting creativity, innovativeness, and knowledge sharing. However, at the same time cultural diversity might become a barrier for communication and effective coordination. Cultures differentiate especially with respect to the sense of time, social hierarchy, power distance, and preferable communication styles. All these distinctions affect the norms for organizational and working culture. Therefore culture-specific understanding and training should be addressed by organizations in order to create mutual awareness and avoid conflicts and misinterpretations [4, 25, 30, 40, 42].

4.8 Infrastructure

Infrastructure refers here to all tools, platforms, and other technical means that support technical, organizational, and managerial activities in the context of distributed software development, maintenance, and operation. The term infrastructure subsumes here, for instance, tool support for coordination and communication, IDEs, and quality assurance tools. Although the infrastructure al-

ready plays an important role in co-located development, the global distribution of development tasks imposes additional and new requirements that should be considered early on. It is necessary for organizations to identify infrastructure-related requirements, to analyze the existing infrastructure, and to invest into the infrastructure in order to reach the stated requirements. In addition, there is a need to analyze how the existing infrastructure at different sites can be modified so that it fits to a new distributed setting. One essential requirement for the infrastructure is the compatibility between sites. For instance, collaboration sites should have equal internet connections, bandwidths, and communication facilities. Compatibility is important, for instance, for configuration management environments, for development tools, and for coordination support. Coordination tools and communication tools promise to help mitigate communication risks that are due to temporal, geographical, and cultural distances. A rich set of groupware tools is expected to help reduce the impact of distance in global software development, to increase the frequency and ease the communication between sites, to lessen coordination efforts, and to provide equal accessibility to all project-related artefacts. A compatible infrastructure at all collaboration locations is highly important for conducting the distributed development process effectively and efficiently [7, 28].

4.9 Organizational Change Process

When organizations start setting up global software development collaborations, there is clear evidence that a sufficient amount of time is needed in order to gain desired efficiency [27]. At first, challenges such as communication, coordination, trust building, awareness of partners and integration of working procedures imply significant reductions of the overall efficiency. Reasons for the decrease in the work efficiency in the first collaboration stages are usually the time necessary for building a compatible infrastructure, establishing the necessary communication ways, providing domain, technology and cultural training, as well as building social relationships and teamness. After the first stages of a global collaboration, there is typically a period of time when partners learn to know each other and better understand the ways of working together. In this phase, the software development efficiency is usually recovering gradually. After this phase, global collaborations might exploit scaling effects with respect to efficiency that go beyond the efficiency of co-located development [9, 27, 41]. Gaining these scaling effects requires the establishment of systematic process improvement procedures.

The accumulated working history with respect to the transfer from co-located software development into a GSD working style gives a lot of insights and thus should later be examined by organizations for potential improvements [9, 26, 27]. New ideas for process changes and improvement actions should be discussed and analyzed jointly by the collaboration sites on a regular basis during the whole period of the collaboration. The improvement of the distributed collaboration can follow different process improvement approaches such as the continuous or the model-based improvement. However, there is a lack of improvement approaches

and experience that are focused on global collaborations. Therefore, we recommend to deploy a problem-oriented, continuous improvement approach. The continuous process improvement aims at reaching a high level of standardization of the overall global software development process that might lead to improved end product quality and customer satisfaction [19].

5 Canvas

While setting up global software development collaborations different phases can be distinguished. Each collaboration phase can be characterized by a specific set of activities that need to be performed by the organizations. We have distinguished four main phases that organizations face when setting up global collaborations. Based on defined collaboration phases and aspects that need to be addressed by organizations, we have structured them as activity roadmaps that can be adjusted for specific organizational contexts. The initial sequence of activities was provided by the case company and refined at a joint workshop of Daimler and the University of Helsinki. The final order of activities was created mainly based on experiences reported by project leaders from the case company and results from the literature study. Some relations between activities also have an underlying inner logic. The proposed activity roadmaps are aimed to be a guidance and reminder for organizations about activities that need to be performed when setting up global collaborations and moving from a local to a globally distributed working mode. For practical industry use, we propose an initial version of a visualized structure of activity roadmaps that we named "Global canvas" (Fig.1). The proposed activity roadmaps for organizations intending to establish global software development collaborations are described as follows.

Phase 1. **Initiate**: In this phase an organization intends to transfer co-located software development into a global context as one of its business strategies. Thus the organization should investigate the potential benefits of transition into a GSD environment, what models of global collaboration exist and what model will suit the specific organizational context. Moreover, the organization makes its first decisions on the partnership type and selects collaboration partners. Thus, the proposed sequence of activities to be addressed by the organization at the initiation phase might look as follows:

a) Identify needs and goals for doing a global software development collaboration. Analyze carefully what benefits and outcomes are expected of the global collaboration.
b) Choose a global collaboration model that is suited for the specific organizational context and the business needs.
c) Investigate the foreign legal system(s) concerning IP and contract laws.
d) Choose appropriate partner(s)/vendor(s) with sufficient infrastructure, capabilities and expertise needed for the chosen collaboration model.
e) Define a budget plan for doing global software development projects. Include possible hidden costs such as communication tools or face-to-face visits.

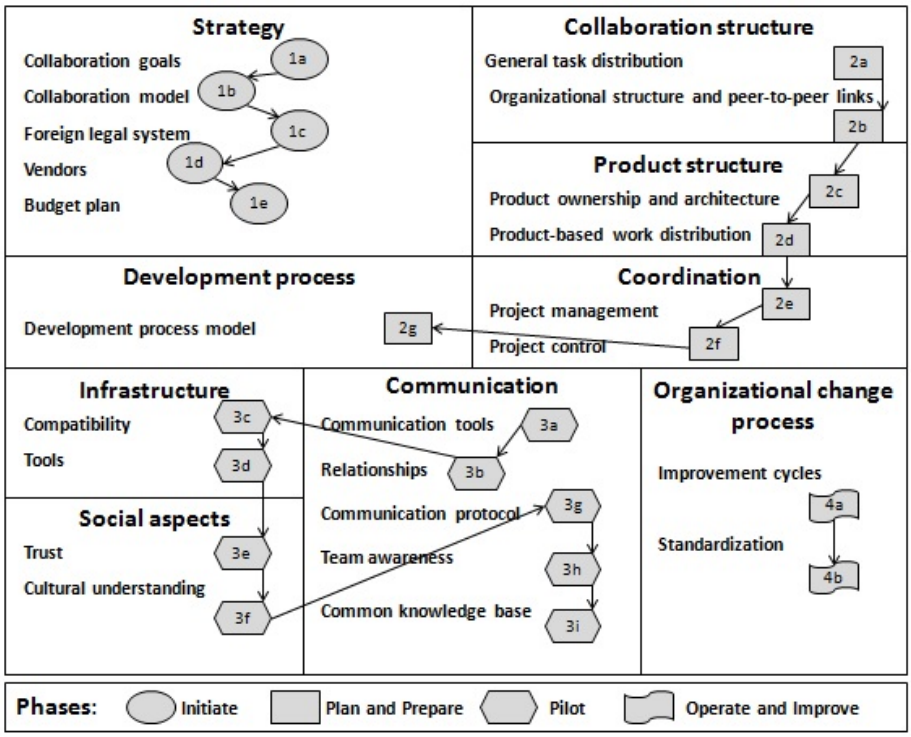


Fig. 1. Global canvas

Phase 2. **Plan and Prepare**: This phase aims at building all the conditions needed for global collaboration to start functioning. An organization defines a product structure, a work distribution, and responsibilities between collaboration sites based on available resources. The model for work coordination and the development process organization is chosen. In this phase, the organization still keeps ongoing product development mainly onsite. However, at the same time, the organization pilots first practices of setting up global software development collaboration. The potential set of activities that need to be done by companies at the preparation phase is described as follows:

a) Identify a development process breakdown and subsequent task distribution.
b) Define and document the organizational structure including specific roles, responsibilities and peer-to-peer contact channels.
c) Identify an architecture and the product ownership between sites (based on the collaboration model defined earlier and the process breakdown).
d) Define a product-based task distribution between sites based on available resources and capabilities.

e) Define the coordination mechanisms between collaboration sites. Choose an appropriate project management model that suits the agreed partnership model.
f) Specify project control procedures for monitoring work progress and detecting problems as early as possible. For instance, the data on channels for status reporting should be assigned, documented and distributed to team members.
g) Choose a model according to which the software development process will be working.

Phase 3. **Pilot**: This phase focuses on systematic testing of practices. This phase aims at detecting the problems of collaboration - if some things do not work at first, they can be changed early on. The proposed activities to be performed are described as follows:

a) Provide a rich variety of communication tools in order to stimulate communication between sites and to avoid misunderstandings.
b) Start gradual building of social relationships between sites (e.g., organize face-to-face visits, joint social activities, staff exchanges).
c) Ensure that the remote partner(s) has sufficient infrastructure needed for software development projects. Provide compatibility of internet connections, bandwidths, communication facilities (for instance, video conference rooms) between sites.
d) Introduce groupware tools that are aimed to ease the collaboration process between sites.
e) Consider socio-cultural aspects between partners. Start building trust between sites early on.
f) Ensure the teams' awareness of cultural differences and perceptions that might occur in collaboration between partners belonging to distinct societies.
g) Establish rules for a communication protocol. Identify who should communicate with whom and how often. Make team members understand that communication is an important part of everyday work.
h) Ensure team awareness channels. Team members need to be aware of remote colleagues' contact details, expertise, roles and responsibilities, work schedules. Ensure that the teams are aware of the project status.
i) Accumulate the experience based on the working history between sites, and create a collective shared knowledge base - the "organizational memory".

Phase 4. **Operate and Improve**: The overall operation of a global software development process is ongoing. Partners accumulate working history, learn, propose and handle process changes and improvements. This phase aims at a seamless operation and a continuous improvement of the collaboration. The set of activities at this stage is suggested as follows.

a) Analyze the working history, discuss potential process changes and improvements.

b) Improve the process continuously and thereby aim at achieving a high level of process standardization.

The proposed prioritization of activities in the different phases is not a strict order but meant as guidance for practical use. The order of activities can be customized based on specific organizational context and needs.

6 Conclusions

In this article we investigated and aggregated the aspects and main practices that need to be addressed by companies when establishing global software development collaborations. Furthermore, necessary activities were grouped into collaboration phases and structured in a form of activity roadmaps that can be used by industry as guidance for setting up global projects. The initial version of a "Global canvas" presents the visualization of activity roadmaps. The canvas provides a holistic view on setting up global collaborations, aggregates all the main necessary aspects and presents the activities as feasible roadmaps.

However, the sequence of activities proposed in our canvas is not mandatory and based on assumptions, literature findings, and industry inputs. The presented aspects are derived from the case company and might differ in other contexts. Therefore, the general applicability is limited and more experience is needed to better understand context-specific customization needs. As global software development is gaining a growing interest and many companies in the domain search for new business opportunities in a transition from co-located development into the global environment, a practice-oriented worksheet that guides decision making such as the canvas promises high potential. Besides using the canvas for guidance, it could also be used for other purposes. Another use case could be, for instance, using the canvas as an assessment scheme. We are planning to further evolve the canvas based on findings from applications in industry. In addition, we are planning to systematically analyze the dependencies between different practices and strategies as well as the suitability of the canvas for other purposes than guidance.

References

1. J. Webster and R. T. Watson, Analyzing the past to prepare for the future: Writing a literature review. MIS quarterly, 26(2), 2002.
2. S. Betz, S. Fricker, A. Moss, W. Afzal, M. Svahnberg, C. Wohlin and T. Gorschek, An Evolutionary Perspective on Socio-Technical Congruence: The Rubber Band Effect. In Replication in Empirical Software Engineering Research (RESER) Workshop, 2013, pp. 15-24.
3. R. Whittemore and K. Knafl, The integrative review: updated methodology. Journal of advanced nursing, 52(5), 2005, pp. 546-553.
4. H. Huang and E. M. Trauth, Cultural Influences on Temporal Separation and Coordination in Globally Distributed Software Development. In ICI, 2008.

5. D. Šmite, C. Wohlin, Z. Galvina and R. Prikladnicki, An empirically based terminology and taxonomy for global software engineering. Empirical Software Engineering, 2012, pp. 1-49.
6. I. Nurdiani, R. Jabangwe, D. Šmite and D. Damian, Risk identification and risk mitigation instruments for global software development: Systematic review and survey results. In Global Software Engineering Workshop (ICGSEW), 2011, pp. 36-41.
7. J.M. Verner, O.P. Brereton, B.A. Kitchenham, M. Turner, M. Niazi. Risks and risk mitigation in global software development: A tertiary study. Information and Software Technology 56 , 2014, pp. 5478, in press.
8. D. Šmite, C. Wohlin, T. Gorschek and R. Feldt, Empirical evidence in global software engineering: a systematic review. Empirical Software Engineering, 15(1), 2010, pp. 91-118.
9. A. Bhadauria, S. Bhattacharjee, C. B. Anandkumar and S. Puthiyonnan, Sustaining High Performance in an Offshore Team in Globally Distributed Development: A Success Story. In Global Software Engineering (ICGSE), 2013, pp. 120-123.
10. D. Bhadade, A Guide to Escalation in Project Management, February 27, 2013, unpublished.
11. J. W. Rottman, Successfully outsourcing embedded software development. Computer, 39(1), 2006, pp. 55-61.
12. P. J. Agerfalk, B. Fitzgerald, H. Holmstrm, B. Lings, B. Lundell and E. O. Conchuir. A framework for considering opportunities and threats in distributed software development. In International Workshop on Distributed Software Development, 2005, pp. 47-61.
13. W. Kobitzsch, D. Rombach and R. L. Feldmann, Outsourcing in India. Software, IEEE, 18(2), 2001, pp. 78-86.
14. B. Lings, B. Lundell, P. J. Agerfalk and B. Fitzgerald, A reference model for successful Distributed Development of Software Systems. In Global Software Engineering, ICGSE 2007, pp. 130-139.
15. D. Šmite, C. Wohlin, A. Aurum, R. Jabangwe and E. Numminen, Offshore insourcing in software development: Structuring the decision-making process. Journal of systems and software 86, 2013, pp. 1054-1067.
16. D. Šmite, Global software development projects in one of the biggest companies in Latvia: is geographical distribution a problem?. Software Process: Improvement and Practice, 11(1), 2006, pp. 61-76.
17. T. Forbath, P. Brooks and A. Dass, Beyond cost reduction: Using collaboration to increase innovation in global software development projects. In Global Software Engineering, ICGSE 2008, pp. 205-209.
18. J. D. Herbsleb, D. J. Paulish and M. Bass, Global software development at siemens: experience from nine projects. In Software Engineering, ICSE 2005, pp. 524-533.
19. R. Prikladnicki, J. L. N. Audy, D. Damian and T. C. de Oliveira, Distributed Software Development: Practices and challenges in different business strategies of offshoring and onshoring. In Global Software Engineering, ICGSE 2007, pp. 262-274.
20. I. Richardson, V. Casey, F. McCaffery, J. Burton and S. Beecham, A process framework for global software engineering teams. Information and Software Technology, 54(11), 2012, pp. 1175-1191.
21. S. Beecham, P. OLeary, I. Richardson, S. Baker and J. Noll, Who are we doing Global Software Engineering research for?. In Global Software Engineering (ICGSE), 2013, pp. 41-50.
22. A. Piri, T. Niinimäki and C. Lassenius, Fear and distrust in global software engineering projects. Journal of Software: Evolution and Process, 24(2), 2012, pp. 185-205.

23. J. Pyysiäinen, Building trust in global inter-organizational software development projects: problems and practices. In International Workshop on Global Software Development, 2003, pp. 69-74.
24. M. Paasivaara and C. Lassenius, Collaboration practices in global interorganizational software development projects. Software Process: Improvement and Practice, 8(4), 2003, pp. 183-199.
25. V. Mettovaara, M. T. Siponen and J. A. Lehto, Collaboration in Software Development: Lesson Learned from Two Large Multinational Organizations. In PACIS, 2006.
26. D. Šmite, A case study: coordination practices in global software development. In Product Focused Software Process Improvement, 2005, pp. 234-244.
27. D. Šmite and C. Wohlin, Lessons learned from transferring software products to India. Journal of software: Evolution and process, 24(6), 2012, pp. 605-623.
28. M. R. Thissen, J. M. Page, M. C. Bharathi and T. L. Austin, Communication tools for distributed software development teams. In Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, 2007, pp. 28-35.
29. N. B. Moe and D. mite, Understanding a lack of trust in Global Software Teams: a multiplecase study. Software Process: Improvement and Practice, 13(3), 2008, pp. 217-231.
30. I. Oshri, J. Kotlarsky and L. P. Willcocks, Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects. The Journal of Strategic Information Systems, 16(1), 2007, pp. 25-49.
31. H. W. Nissen, "Designing the inter-organizational software engineering cooperation: an experience report."2004, pp. 24-27.
32. M. F. Faiz, U. Qadri and S. R. Ayyubi, Offshore software development models. In Information and Emerging Technologies, ICIET 2007, pp. 1-6.
33. J. Cusick and A. Prasad, A practical management and engineering approach to offshore collaboration. Software, IEEE, 23(5), 2006, pp. 20-29.
34. A. Braun, A framework to enable offshore outsourcing. In Global Software Engineering, ICGSE 2007, pp. 125-129.
35. F. Salger, On the use of handover checkpoints to manage the global software development process. In On the Move to Meaningful Internet Systems: OTM 2009 Workshops, pp. 267-276.
36. M. Leszak and M. Meier, Successful Global Development of a Large-scale Embedded Telecommunications Product. In Global Software Engineering, ICGSE 2007, pp. 23-32.
37. F. Q. Silva, R. Prikladnicki, A. C. C. Frana, C. V. Monteiro, C. Costa and R. Rocha, An evidence-based model of distributed software development project management: results from a systematic mapping study. Journal of Software: Evolution and Process, 24(6), 2012, pp. 625-642.
38. K. T. Chang, and K. Ehrlich, Out of sight but not out of mind?: Informal networks, communication and media use in global software teams. In Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, 2007, pp. 86-97.
39. A. Nguyen-Duc and D. S. Cruzes, Coordination of Software Development Teams across Organizational Boundary–An Exploratory Study. In Global Software Engineering (ICGSE), 2013, pp. 216-225.
40. G. Hofner and V. S. Mani, TAPER: A generic framework for establishing an offshore development center. In Global Software Engineering, ICGSE 2007, pp. 162-172.

41. A. Mockus and D. M. Weiss, Globalization by chunking: a quantitative approach. Software, IEEE, 18(2), 2001, pp. 30-37.
42. S. Deshpande and I. Richardson, Management at the Outsourcing Destination-Global Software Development in India. In Global Software Engineering, ICGSE 2009, pp. 217-225.
43. V. Casey, Virtual software team project management. Journal of the Brazilian Computer Society, 16(2), 2010, pp. 83-96.
44. E. Hossain, M. A. Babar and J. Verner, How Can Agile Practices Minimize Global Software Development Co-ordination Risks?. In Software Process Improvement, 2009, pp. 81-92.
45. J. Hyysalo, P. Parviainen and M. Tihinen, Collaborative embedded systems development: survey of state of the practice. In Engineering of Computer Based Systems, 13th Annual IEEE International Symposium and Workshop, 2006, pp. 1-9.
46. A. Lamersdorf, J. Münch, D. Rombach, Towards a Multi-criteria Development Distribution Model: An Analysis of Existing Task Distribution Approaches. In Global Software Engineering (ICGSE), 2008, pp. 109-118.
47. C. Wohlin, Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. EASE14 18th international conference on Evaluation and assessment in software engineering, 2014, pp. 321-330.