8th IEEE International Conference on Global Software Engineering (ICGSE 2013), Bari, Italy, August 26-29 2013 © IEEE 2013

A Preliminary User Evaluation of an Infrastructure to Support Activity-Based Computing in Global Software Development (ABC4GSD)

Paolo Tell
Software System Section
IT University of Copenhagen, Denmark
pate@itu.dk

Muhammad Ali Babar
IT University of Copenhagen, Denmark
Lancaster University, UK
maba@itu.dk

John Grundy
Centre for Computing and
Engineering Software Systems
Swinburne University of Technology
jgrundy@swin.edu.au

***Abstract*—Global Software Engineering (GSE) teams face challenges due to the need to replace traditional physical presence interactions and co-ordination with computer-mediated means. A vast majority of the available tool support for distributed collaboration is deep-seated in a desktop metaphor introduced in the '70s, and huge efforts are being devoted to overcome its known limitations. Over the last few years we have looked into the feasibility of providing an approach based on Activity-Based Computing (ABC) to address these issues in a novel way. We have developed a middleware and support tool, ABC4GSD, to enable collaborative distributed features in an application, while maintaining common interactions with the workstation that users are accustomed to. In this paper we present the results of a user evaluation we conducted on ABC4GSD using exemplar GSE scenarios. Participants' responses show positive and encouraging reception of the activity-based approach to supporting GSE.**

I. Introduction

The geographical distribution of software development teams across multiple sites has become a widely accepted software development paradigm, usually known as Global Software Engineering (GSE) [12]. As a result of the increasing popularity of GSE, more and more software development projects are being undertaken by following some kind of GSE paradigm. This promises several different benefits, such as availability of large pool of skilled developers, relatively low cost of development, and shorten time-to-market [8]. However, it also entails many unique challenges defined by geographical, cultural, linguistic, and temporal distances [22][8]. For example, an intensified degree of communication, coordination, and collaboration among team members is required and the provision of suitable tool support is essential. Communication, coordination, and collaboration—also called 3Cs—were initially identified for groupware technologies [11], and afterwards have been widely accepted as key features for many tools in Software Engineering (SE) [29].

Given the critical role and importance of appropriate and adequate tool support for GSE teams, many different solutions have been proposed by both academia and industry [17][37][29]. However, there are still research challenges related to bridging the gap between co-located and distributed environments. Previous research has focused on either enhancing particular development processes, e.g., inspection meetings [18] and requirements management [16], or incorporating information from different sources into monitoring applications, e.g., T3 [35] and FastDash [3]. We argue that to support GSE teams more comprehensively, these challenges need to be faced from a wider perspective, thus by providing "[...] a virtual space wherein all the stakeholders of a project—even if distributed by time or distance[, culture or language,]—may negotiate, brainstorm, discuss, share knowledge, and generally labor together to carry out some task, most often to create an executable deliverable and its supporting artifacts" [5]. Even if Booch and Brown [5] explicitly suggest that the web platform appears to be the most suitable environment for hosting Collaborative Desktop Environments (CDEs), different approaches are being trialled. Recently, Tamburry et al. [31] have also confirmed the importance of explicitly supporting social communities, and social structures. Awareness of the teams, of the tasks, and of the available skills are identified as critical in the context of GSE. In their study, they show a major need for task and artifact co-ordination in GSE, argument which we tackled in [34] by showing how artifact-oriented approached can be supported by our application of the Activity Theory (AT) [19].

Encouraged by its application in different domains, we are leveraging the Activity-Based Computing (ABC) [23] paradigm—grounded in AT—for providing a flexible and easily adaptable infrastructure for building and using activity-aware tools for GSE teams. This use of the ABC paradigm in the context of GSE has also been motivated by the successful use of Activity-Centric Computing (ACC) [38] and ABC [2] in different disciplines, such as the use of ABC for supporting mobile and distributed users in handling concurrent activities for intense collaboration in hospitals [2]. An ABC enabled system can:

- provide a unified abstraction mechanism able to aggregate the different digital and human resources required for a particular software development activity in a personalized manner;
- support multi-tasking among different activities, lessening metal overhead and reducing the burden of constantly dealing with work fragmentation;
- support collaboration, coordination, and the creation and preservation of awareness; and,
- facilitate, to a certain extent, application interoperability.

We have developed a prototype implementation of an ABC system for GSE, called *ABC4GSD*, which core features will be briefly described. We have carried out an evaluation of

our ABC4GSD platform for some exemplar GSE activities. We present the results of a user evaluation that we have conducted to assess its perceived usefulness, ease of use, and self-predicted future use, using an adapted version of the Technology Acceptance Model (TAM) proposed in [15]. We enriched the questionnaires used in our study to focus also on two critical aspects of GSE working environments, i.e., the aggregating capabilities of the system and its ability to support work fragmentation.

In the remainder of this paper, we will briefly describe in Section III the high-level client-server architecture that has been designed to support the introduction of ABC in GSE and the current user interface. Further, Section IV will list the research objectives of this paper, and Section V will introduce the user evaluation, which results are discussed in Section VI. Threats to the validity of the experiment are the topic of Section VII; and results are further discussed in Section VIII.

II. Background and Motivation

A. Tools for GSE

Diverse approaches have been used to provide tool support for GSE teams. However, a majority of them remain bound to a desktop-based single user paradigm, whose heritage can be traced to the '70s [27]. A huge effort has to be devoted to overcome the well known limitations of the currently used desktop metaphor, which was designed to support tasks performed by a single user. As such, some tool support for GSE has been developed in the form of plug-ins providing additional multi-person functionalities to improve existing single-user applications. There is an increasing focus on enhancing Integrated Development Environments (IDEs), such as Eclipse (e.g., IBM Jazz [13]) or Visual Studio (e.g., FASTDash [3]), to provide this kind of extension. One of the most successful examples of such tools in the context of distributed software engineering is Jazz [13], consisting of a set of plug-ins for Eclipse designed to provide status- and task- related awareness about team members. The most notable visual element is the Jazz band: a plug-in that enhances Eclipse by providing the list of people participating to the project inside the IDE. The whole system is supported by IBM Rational Jazz, which is a platform providing a binding service layer that allows information sharing throughout the linked components. Whilst, there have been several successful reports of integrating and using different GSE tools developed as plug-in, most of them have been designed to address one particular challenge being faced by GSE teams and are hardly customizable. Furthermore, even if these plug-in based tools may potentially be very effective, they are designed for specific software technologies and do not interoperate with other system.

Hundreds of tools have been developed to support GSE teams [24][33]. However, it can be argued that there has been relatively little attention paid to tackle the problem with a wider perspective by enhancing coordination, collaboration, and communication within practitioners' teams. The majority of available tools focus on mitigating a specific issue by enhancing a particular software process (e.g., [7]), denying the benefits that could be gained by approaching GSE issues as an integrated system to support. Following this direction, researchers have looked into the feasibility of providing a better integration among the numerous tools utilized by GSE teams [14]. To tackle this objective, two different approaches have been followed: one operating at the model level, the other at the application level. The idea behind the former is to provide a common abstraction layer with which applications can interoperate by having a common understanding about the information handled. Examples of such approach include SoftDock [30], which allows developers to collaboratively model software components in UML by using UML eXchange Format (UXF) to exchange data between the clients in a CORBA architecture; and ADAMS [10], a system that improves project and artifact management by providing a fine-grained versioning system able to drastically enhance traceability. However, similarly to plug-in tools, these solutions provide support only to one particular aspect of the software ecosystem used in a GSE work environment and are not suitable for enhancing crucial characteristics of computer-mediated teamwork as a whole problem. In the case of application level integration, approaches try to support tools by providing a common way of interacting via an infrastructure or middleware. Through this method, the work environment can be enhanced as a whole providing an interaction mechanism to support different kind of enhancements. An approach, this, followed also by the solutions proposed by commercial vendors like IBM (i.e., Rational Suite), which are pushing towards comprehensive approaches ensnaring users in their platform packages. Two main architectural styles are used in these systems: peer-to-peer (P2P), often leveraging agent systems, and client-server. Examples of the P2P group can be found in [25] where an architecture is used to enhance collaboration in a distributed environment or in [21] where an environment able to provide applications integration is described. However, to justify the use of a P2P architecture, it is claimed that coping with GSE environments entails the presence of a large number of users, thus, a dense interaction that may find a bottleneck in a client-server design. Our belief is that this is rarely the case. In fact, one of the reasons behind the decision of starting a global or distributed collaboration is the need for more readily available skills [28], often because the initiator of the collaboration is a small- or medium- size organization. Moreover, the performances of the network infrastructures are constantly increasing and solutions deployed on cloud based infrastructures are able to minimize the issue presented as the main argument for choosing a P2P architectural style. Therefore, the tradeoff between risking bottlenecks with a client-server solution and loosing all the benefits of having a centralized control over the communication by adopting a P2P approach, is resolved in our design in favor of the client-server architectural style. An interesting example based on this approach is the middleware designed to enhance CASE tools providing awareness information presented in [20].

We assert that, from our experience [32][34], rather than enhancing a single application or process, there is a key need of an approach and infrastructure that can enable tools to adapt to the activities to be performed by GSE team members. By allowing existing and new tools to be activity-centric, such an infrastructure could help GSE practitioners to aggregate all the required human and digital resources for a particular software engineering activity such as designing a software in a collaborative arrangement; allow them to switch between activities and hand-over activities to other team members

geographically distant; allocate and reallocate resources for different activities; and be aware of other members' working status.

B. Approaches Based on Activity Theory

In activity-based approaches to work co-ordination, the fundamental metaphor provided to organize work is focused on the concept of the *activity*, rather than being strongly tied to the ones of file/document or application. Examples include the Gnome project, called *Gnome Shell*[1], where the concept of workspace is replaced by the one of activity bundling up digital artifacts and applications connected to them. Applying the constructs provided by Activity Theory, Yarosh et al. from IBM [38] have defined their work as Activity-Centric Computing (ACC): an activity theory loosely inspired approach designed to "[...] address work fragmentation by allowing users to structure their work around the computational construct of an Activity." Other notable contributions for tool support based on the activity theory include the desktop manager called Giornata [36], and the Context-Aware Activity Display (CAAD) [26]. In these examples, the approach followed is the one initially introduced by Norman in the '90s called Activity-Based Computing (ABC). In his book [23], he states that "[...] the basic idea is simple; make it possible to have all the material needed for an activity ready at hand, available with little or no mental overhead.". Thus, the core concept of ABC is to provide an automatic, seamless, and non-intrusive support for activities. Bardram has successfully demonstrated the applicability of the ABC paradigm for supporting different collaborative activities in hospitals [2]. He developed a framework that provides a robust replacement for the application-oriented computing paradigm by focusing more on the principles of roaming, sharing, and awareness. However, as we have shown in [34], the application of activity theory in software engineering differ from the one applied in hospitals; and, the core principles of the ABC paradigm [2] can provide a solid foundation for building an infrastructure that can help address many of the existing GSE related challenges.

III. ABC4GSD

Our ABC4GSD system is a middleware incorporating ABC concepts and is based on the client-server architectural style that supports event based co-ordination and communication. The main high-level components of its infrastructure are depicted in Fig. 1. The *Remote Server* is the element responsible for ensuring the persistency and the consistency of the data. It contains a data layer and is responsible for generating the events necessary to propagate the correct communication of the information to and from all client machines. The client architecture comprises two key elements: the main controller, represented by the *ABC4GSD Client* box, and the *ABC App Interface*. The ABC4GSD Client is the main component hosted on each client workstation. It acts as a broker between the remote server and the client applications by dispatching messages generated locally and directed to the remote server and vice versa. The ABC4GSD Client can be a stand-alone application running on the client host machine or an Eclipse plug-in. The purpose of the ABC App Interface is to provide

[1]Gnome Shell released with Gnome version 3 (http://www.gnome.org/).

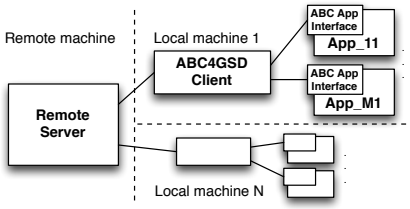

Fig. 1. High-level architecture components.

a common way to interact between the system and the applications. Applications have to adhere to this specific interface to be fully integrated in the system. This component provides functionalities to send and receive messages and subscribe or unsubscribe to events. It also defines a list of methods that have to be implemented to have both the ABC4GSD Client control some behavior of each application and the applications react to events. Such a mechanism allows the automatic integration of response behaviors as we will introduce in the next section.

All communication between the remote server and the ABC4GSD client as well as between the ABC4GSD client and the interfaces have been implemented using ZeroMQ[2]. This is a messaging library that allows the design a complex communication system through intuitive APIs also providing messaging patterns like publish-subscribe or request-reply.

All additional client elements are treated as independent applications, i.e., concrete external processes, for which, operative system calls are used to launch and terminate them. This design decision has been taken to maximize the decoupling of each component allowing the just in time (JIT) insertion or substitution of one element with another one. Further tackling components independence, artifacts are bound indirectly to applications by storing the 'need' that an application have to support to utilize the artifact. As an example, an application that needs to handle pdf artifacts would be represented by an identification string (e.g., 'pdfViewer'), which at every client-side can be flexibly bound to a specific application (e.g., Skim for workstations using MacOS, Foxit Reader for those running Windows systems, or an ad-hoc viewer implementing the ABC App Interface).

A. User Interface Overview

All components of the ABC4GSD Client application are Eclipse plug-ins deployed on an independent application built as an Eclipse RCP[3] application. Therefore, all plug-ins developed for the ABC4GSD Client application can be deployed directly within the Eclipse platform to enhance the main Eclipse application. However, our research effort is not aimed at providing specific tools for GSE nor at the enhancement of one of them. Rather, we want to explore how (i) integration and co-ordination mechanisms such as those provided by ABC4GSD can provide to SE tools a more flexible mechanism facilitating application interoperability, and (ii) by means of the ABC metaphor, whether we can provide better support for work fragmentation and help users in aggregating human and digital

[2]http://www.zeromq.org

[3]Eclipse RCP. The Eclipse Rich Client Platform is a plug-ins subset of the entire Eclipse platform that allows the development of independent application.

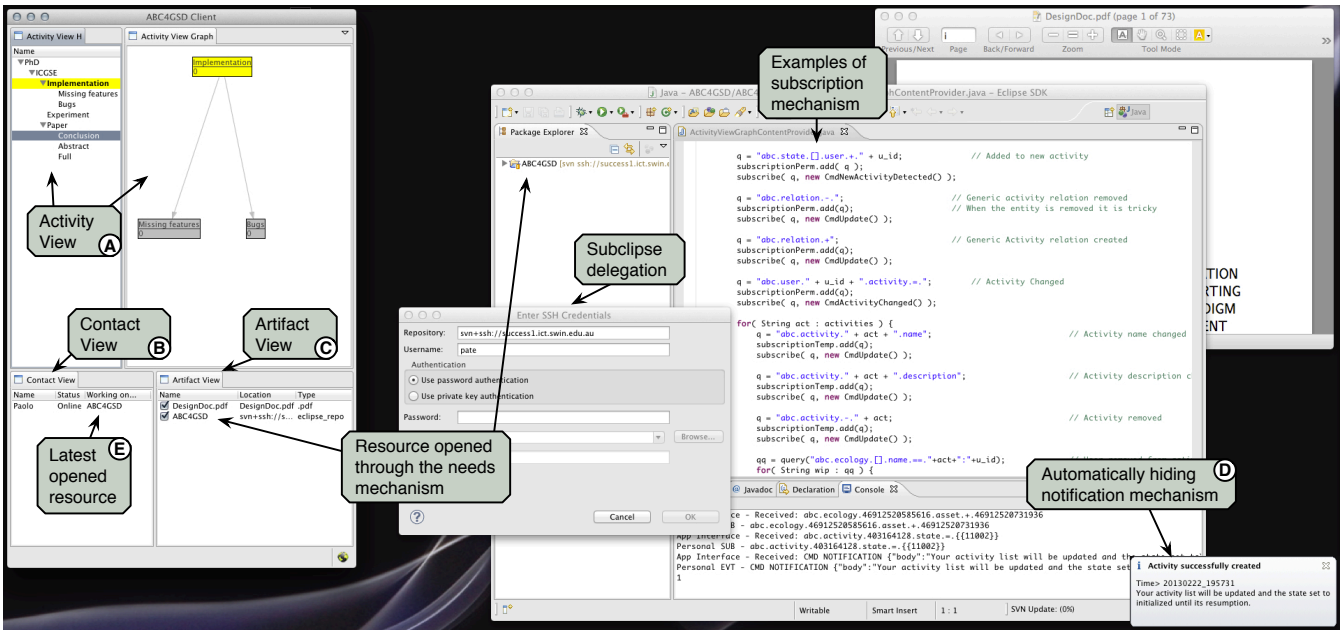

Fig. 2. Overview of the ABC4GSD user interface.

resources in a GSE setting. In the following subsections we provide an overview of each client-side component included in the ABC4GSD user interface.

Activity View (Fig. 2-A): this is the component responsible for displaying the list of activities in which a user is participating. The left hand side presents the user with a hierarchical view of his/her activities. Through a double click, activities can be resumed. This causes the population of the graph view (top right quadrant), providing a detailed view of the resumed activity. In both views, a contextual menu is available to access commonly used functionalities (e.g., modification, removal, cloning). The depth of the graph presented in the graph view has been limited to three levels, which, besides avoiding a cluttered overview, maps the activity hierarchy described by Leont'ev (i.e., activity, action, operation).

Contact View (Fig. 2-B): this plug-in is designed to show the people participating in the activity currently resumed, it provides information about their status and the artifact each of the users is working on (Fig. 2-E). At this stage, if a user is connected but working on a different activity, both the activity and the artifact are revealed.

Artifact View (Fig. 2-C): is used to list all digital artifacts and general resources used inside an activity. It shows the name, location, and type for each of them. A checkbox is also provided to control the automatic loading of the artifact during the resumption of the activity.

Notifications (Fig. 2-D): provided with the ABC application interface, a common notification mechanism is shared by all plug-ins. This component is able to visualize ad-hoc temporary notifications. In the current version of the system it has been used only to notify few events, e.g., the creation of a new activity in which the user is involved (Fig. 2-D), but could easily be extended to provide a more comprehensive awareness support.

ABC-Enabled Eclipse: to perform this experiment the Eclipse IDE has been enhanced by means of a plug-in designed to act as middle man between the ABC4GSD system and Subclipse[4]. Information about the currently loaded activity, the user, and the artifact are sent to the plug-in upon resumption. This allows the plug-in to retrieve the remaining information querying directly the server and programmatically instructing Subclipse to checkout or update the linked subversion repository. The automation currently supported has been intentionally limited to the checkout and update of the repository, which is executed upon the loading of the resource, thus leaving the commit process completely controlled by the user.

Most applications, in order to benefit and participate to the middleware, need to implement the interfaces described in the previous section to obtain a tight interaction (e.g., ABC enabled Eclipse). However, to obtain a more versatile system, not limited only to applications implementing with the interfaces, we decided to support a more shallow coupling as well. In such a case, the control of their behavior—the suspension and resumption life-cycle imposed by the ABC paradigm—is directly controlled through operating system calls invoked by the ABC4GSD Client (e.g., using Skim for viewing pdfs).

Finally, to show the efficacy of the event system, the business logic of all plug-ins described is implemented through the subscription mechanism provided by the middleware. All the updates are obtained by subscribing and automatically responding to the events received with a small additional coding effort. A code snipped is visible in Fig. 2 within the Eclipse editor—the code is the implementation of the behavior of the Graph View in the top right quadrant of the ABC4GSD Client.

[4] subclipse.tigris.org

IV. Research Objective

Our primary goal in this study was to investigate the use of the ABC4GSD system regarding the viability of introducing an approach based on the concept of activity in the context of GSE. To do this we designed a user evaluation driven by scenarios (as described in [6]) investigating the perceived usefulness, perceived ease of use, and self reported future use in relation to an exemplar GSE environment augmented by ABC4GSD. Further, we included two additional variables related to the aggregating capabilities of ABC4GSD and its ability to help handling work fragmentation. The research framework chosen to address these matters has been an adapted version of the technology acceptance model (TAM) [15], extended by two additional sets of questions addressing aggregating capabilities and work fragmentation. Additionally, the opportunity was also leveraged to gather feedbacks and collect opinions about ABC4GSD. The key underlying research questions (RQ) that were addressed by our study are:

- **RQ1:** *How is the approach based on activities supported by ABC4GSD perceived by the users?* Do users perceive ABC4GSD as useful for the exemplar GSE activities, not useful, or are they indifferent.
- **RQ2:** *Is the approach based on activities supported by ABC4GSD easy to use?* Do users perceive ABC4GSD as easy to use, difficult to use, or are they indifferent.
- **RQ3:** *Would users adopt the approach based on activities supported by ABC4GSD if it was widely available?* Would users adopt our ABC4GSD and/or an activity-based approach to GSE work co-ordination.
- **RQ4:** *Is the abstraction conveyed by the concept of activity properly supported by ABC4GSD for organizing human and digital assets?* Is having the structures provided by ABC4GSD facilitating the coordination and division of labor mechanisms necessary to be productive when working in a distributed GSE team.
- **RQ5:** *Does the activity life-cycle help in handling work fragmentation?* By using ABC4GSD is handling work fragmentation easier, harder, or does there seem to be no difference in how work fragmentation is handled.

In the following sections we describe the design of our study, results obtained from twelve participants, key findings from this study, and key directions for future research that we have identified.

TABLE I. Questionnaires.

Perceived Usefulness	
PUsf.1	Using a system based on activities like ABC4GSD in my job would enable me to accomplish tasks more quickly.
PUsf.2	Using a system based on activities like ABC4GSD would improve my job performance.
PUsf.3	Using a system based on activities like ABC4GSD in my job would increase my productivity.
PUsf.4	Using a system based on activities like ABC4GSD would enhance my effectiveness on the job.
PUsf.5	Using a system based on activities like ABC4GSD would make it easier to do my job.
PUsf.6	I would find a system based on activities like ABC4GSD useful in my job.
Perceived Ease of Use	
PEoU.1	Learning to operate ABC4GSD would be easy for me.
PEoU.2	I would find it easy to get ABC4GSD to do what I want it to do.
PEoU.3	My interaction with ABC4GSD would be clear and understandable.
PEoU.4	I would find ABC4GSD to be flexible to interact with.
PEoU.5	It would be easy for me to become skillful at using ABC4GSD.
PEoU.6	I would find ABC4GSD easy to use.
Self-Predicted Future Use	
SPFU.1	Assuming ABC4GSD would be available on my job, I predict that I will use it on a regular basis in the future.
SPFU.2	I would prefer using ABC4GSD to one dependent on multiple applications for organizing my tasks within a collaborative environment.
Aggregation Capabilities	
AC.1	Unifying *my* work practices through ABC4GSD is easy.
AC.2	Organizing work and consolidating information around the concept of activity is useful.
AC.3	Sharing a document with a colleague is easy.
AC.4	Using ABC4GSD makes the understanding of common outcomes within a team easy.
AC.5	ABC4GSD allows me to organize work and consolidate information around the concept of activity.
AC.6	Organizing a team through ABC4GSD is easy.
AC.7	Having a common way to organize work through ABC4GSD is useful.
Work Fragmentation	
WF.1	Being aware of events happening that pertain my work through ABC4GSD is simple.
WF.2	When using ABC4GSD, I am aware of who is doing what.
WF.3	When using ABC4GSD, I am aware of who is available.
WF.4	When using ABC4GSD, I am able to understand who is responsible for what.
WF.5	When using ABC4GSD, I am able to be aware of the people I am working with in each task and their status.
WF.6	Handling interruptions is simpler when using ABC4GSD.
WF.7	Having my activities organized by ABC4GSD reduces the effort required to handle multiple tasks.

V. User Evaluation

A. Theoretical Model

The Technology Acceptance Model (TAM) [9] aims at assessing user beliefs about the usefulness and ease of use of a technology by means of a questionnaire focused on two variables: perceived usefulness and perceived ease of use. According to Davis [9], perceived usefulness is defined as "the degree to which a person believes that using a particular system would enhance his or her job performance"; whereas, perceived ease of use as "the degree to which a person believes that using a particular system would be free of effort". The measuring model that was used in this study is the extended version of the TAM model that was proposed in [15] and has been adopted by previous studies (e.g., [1]). According to [15], "since both usefulness and ease of use are correlated to self-predicted future usage, they can be considered determinants of tool acceptance behaviors". For each of the three variables the related questionnaires were adapted to focus on ABC4GSD. Moreover, as introduced in the previous section, we added two sets of questions related to the aggregating capabilities of ABC4GSD and its ability to help handling work fragmentation. The questions related to each of the investigated variables are reported in Table I. Similarly to what has been used in [9] and [15], each question was measured with a seven-point Likert scale[5], which allowed us to capture both positive and negative evaluation as well as a neutral one to match the null hypotheses.

[5]Likert scale parameters: (1) extremely likely, (2) quite likely, (3) slightly likely, (4) neither, (5) slightly unlikely, (6) quite unlikely, (7) extremely unlikely.

TABLE II. DEMOGRAPHIC SHEET.

D.1	Name			
D.2	Age			
D.3	Job title			
D.4	Gender			
D.5	Computer literacy	Novice	Average	Expert
D.6	Exp. in working as part of a team	<1 year	1-3 yrs	>3 yrs
D.7	Exp. in working with collab tools	<1 year	1-3 yrs	>3 yrs
D.8	Exp. in software engineering	<1 year	1-3 yrs	>3 yrs

TABLE III. OPEN-ENDED QUESTIONNAIRE.

	Comments and Feedbacks
CaF.1	What did you like about ABC4GSD?
CaF.2	What did you dislike about ABC4GSD?
CaF.3	What would you like to see improved in a future version of ABC4GSD?
CaF.4	What would you like to see included in a future version of ABC4GSD?
CaF.5	What was intuitive about the system?
CaF.6	What was confusing about the system?
CaF.7	Please make any additional comments below.

B. Method

Given the rather novel high-level activity life-cycle interaction introduced by the ABC4GSD system, we opted to expose study participants to the system through a scenario-based approach. This method entails the design of scenarios based on realistic settings exposing participants to complex situations, which would otherwise be hard to observe [6]. Fig. 3, provides an overview of the experimental procedure. The experiment was organized into three sessions as follows.

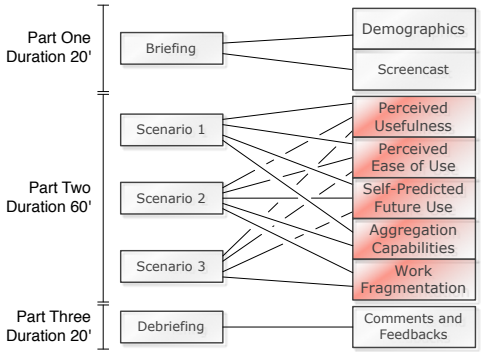

Fig. 3. Experiment breakdown.

In the first session, after signing the informed consent[6] and filling out a basic demographic sheet (Table II), the participant was introduced to ABC4GSD through a 15 minute screencast. In this briefing session, a recorded video was chosen over a presentation to limit bias and ensure repeatability. The video provided an introductory tutorial to ABC4GSD. In particular, it introduced the key concepts and the functionalities needed to perform the scenarios, e.g., how to create and edit an activity, how to log in the system, how to link human and digital assets to an activity, and how to resume and suspend an activity.

[6]This experiment was approved by the Swinburne University of Technology Human Research Ethics Committee - SUHREC Project 2013-003.

In the second part, the participant was asked to perform the three scenarios detailed below adopting a think-aloud approach. Scenarios were treated as independent experiments, and the execution sequence was assigned according to a latin square randomization. After completing each scenario the participant was administered the related questionnaire. For each scenario, the participant was given a description of his/her task introducing the simulated context, additional material was also provided if needed (e.g., a fictitious organizational chart needed to be aware of the available human resources), and the required digital material placed on the desktop of the workstation used. A set of scripts was used to instrument ABC4GSD so that the participant could experience interaction with remote team members[7].

In the third part, the participant was administered a final questionnaire comprising open-ended questions aimed at gathering feedback and comments both positive and negative on the system (Table III).

Self-reported quantitative data is thus the main data collected during the evaluation obtained through questionnaires; qualitative data were gathered through open-ended questions during the final debriefing. Furthermore, additional data was recorded. Interactions with the workstation were captured through a screencast during each scenario (including audio); notes were taken by the main investigator; and system event logs were recorded by capturing timestamps of participant's interactions with the system while performing the scenarios.

C. Scenarios

We summarise the three scenarios we used for our study. We chose these as exemplar activities that are common in a range of GSE teams and that often lack appropriate coordination support. Considering also the several challenges identified by Tamburri et at. regarding GSE social structures [31], we selected some activities that required a variety of tasks to be carried out, artifacts to be used or modified, and participants to carry out sub-activities handed over by remote team members.

Organization of a new project: the participants were asked to play the role of a GSE project manager in the process of organizing his/her team for a new distributed project regarding detailed analysis of three candidate software architectures for a complex system. The participants had to decide independently the granularity of the activities to be created, the people to assign to each activity, the description to provide, and whether or not to assign particular artifacts from a set of available ones. The participants were provided with a hard copy of the organizational chart describing the human resources available, and a possible (but by no means the only) solution for the division into subtasks. Moreover, a fictitious digital document for the requirements as well as three software architecture candidate solutions were present on the workstation local desktop. This scenario was designed to force the participant to experience the aggregating capabilities of ABC4GSD by requesting the detailed creation of several activities with several available team members, some distributed and some local. Moreover,

[7]This pilot evaluation did not comprise scenarios including several real remote users. This will be the focus of the next rounds of our experiments in which ABC4GSD will be evaluated against the distances characterizing GSE.

by performing this scenario, the participants were required to organize a team using the metaphor of activity.

Fixing a bug: the participants were asked to act as a developer. While working on an implementation activity, another activity involving him/her was automatically created. The activity description mentioned an urgent bug in a specified software artifact that needed to be fixed as soon as possible. To successfully complete this task, the participant needed to realize the message received through the notification system and act accordingly. Not realizing the message was considered a failure in the scenario. As in [6], the consideration regarding the level of awareness of the participant adhered to the following scheme:

- participants were evaluated fully aware when they had spontaneously noticed the notification;
- they were evaluated partially aware if they noticed the notification after being prompted by the experimenter;
- they were considered unaware in all remaining cases.

The rational behind the use of this scenario was connected to the ability of ABC4GSD to handle work fragmentation, to keep users aware of events happening in their team, and to delegate activities. Furthermore, the participants were able to experience the integration of the Eclipse IDE.

Preparing for a presentation: the participants were asked to act as a software architect on a GSE project. They were asked to prepare for a meeting focusing on the evaluation of the architecture of a system. At his/her office, he/she had to prepare a presentation named "Architecture evaluation" for driving the discussion during the presentation using his/her personal machine. After quitting the system, he/she were requested to simulate the visit to the company. Once in the meeting room, using the local machine, he/she was asked to resume the activity as left once suspended and deliver the presentation. In addition, the project manager required a copy of the presentation for future use. Therefore, the participant was asked to create an activity to which only the presentation and the project manager had to be linked to. With this scenario we focused on the activity roaming principle of ABC and the ability of the system to share digital artifacts associated with activities among distributed GSE team members.

D. Participants and Setup

For study participant recruitment, neither gender nor age distinction was applied. In total, 12 participants were recruited for this experiment (mean age 33.3) from ICT staff and students of Swinburne University of Technology. A compulsory requirement for their selection was related to their expertise in SE and/or GSE. Therefore, recruited participants were selected among PhD students or Professors heavily involved in software engineering areas (D.8). Table IV, shows the reported experience level in relation to the different experience variables.

It can be seen that even though only slightly more than half of the participants have worked with collaborative software (D7) for more than one year, both the technical background (D5) and the level of experience in working as part of a team (D6) range from medium (i.e., one to three years) to high (i.e., more than three years).

TABLE IV. PARTICIPANT EXPERIENCE.

	Low	Medium	High
D.5	0	42	58
D.6	0	33	67
D.7	33	17	50
D.8	0	0	100

The experimental setup comprised a laptop connected to an external monitor equipped with mouse and keyboard. This setup allowed the main investigator to instrument the system while the participant was filling out questionnaires. Finally the server side of the infrastructure was hosted within the Swinburne University of Technology premises.

VI. RESULTS

Table V provides an overview of the quantitative results of the questionnaires. A discussion of the results including insights from the open-ended questions administered in the debriefing session is provided in the 'Discussion' section.

Perceived Usefulness: in general, ABC4GSD received very good results from the evaluation. If on the one hand, the system was perceived appropriate for performing task more quickly (Table V-(a), PUsf.1), in an easier way (Table V-(a), PUsf.5), and, overall, was perceived as potentially useful (Table V-(a), PUsf.6); on the other hand, participants responses are less positive when related to improved performance (Table V-(a), PUsf.2), productivity (Table V-(a), PUsf.3), and effectiveness (Table V-(a), PUsf.4).

Perceived Ease of Use: all participate managed to appropriate the system with more or less confidence in a very brief period of time through the video tutorial provided in the briefing session. Even though we anticipated that on average scores would have been worse due to the lack of attention paid to usability aspects of the user interface, the participants still found ABC4GSD easy to learn (Table V-(b), PEoU.1, PEoU.5, and PEoU.6) and rewarded the intuitiveness of the ABC paradigm.

Self-Predicted Future Use: also the self-predicted future use variable questions scored well. Participants reported a good likelihood of adopting our ABC4GSD for supporting their future work (Table V-(c), SPFU.1), and expressed an interest in using a system based on activity over the currently available ones (Table V-(c), SPFU.2)

Aggregation Capabilities: the concept of activity has been confirmed to be a powerful metaphor to provide a useful (Table V-(d), AC.7) aggregating tool (Table V-(d), AC.5). Being designed to support collaboration, not surprisingly, the easiness of sharing documents has been confirmed by the evaluation participants (Table V-(d), AC.3). However, some aspect of the system like the easiness of organizing a team (Table V-(d), AC.6) or the understanding of the common goals (Table V-(d), AC.4) scored slightly less. This will be analyzed in Section VIII and our initial hypothesis strongly related such results with the lack of attention given to usability during the design of the user interface.

TABLE V. QUESTIONNAIRES RESULTS. SCORES ARE ON A 7-POINT LIKERT SCALE FROM 1 (EXTREMELY LIKELY) TO 7 (EXTREMELY UNLIKELY). NOTE: ($\bar{x}$) AVERAGE; ($\tilde{x}$) MEDIAN; (σ) STD DEV.

	$\bar{x}$	$\tilde{x}$	σ
PUsf.1	1.97	2.00	0.93
PUsf.2	2.17	2.00	0.87
PUsf.3	2.19	2.00	0.84
PUsf.4	2.14	2.00	0.89
PUsf.5	1.94	2.00	1.03
PUsf.6	1.81	2.00	0.84

(a) Perceived usefulness.

	$\bar{x}$	$\tilde{x}$	σ
PEoU.1	2.06	2.00	1.33
PEoU.2	2.33	2.00	1.47
PEoU.3	2.19	2.00	1.24
PEoU.4	2.47	2.00	1.44
PEoU.5	1.67	1.00	0.91
PEoU.6	1.81	1.50	1.02

(b) Perceived ease of use.

	$\bar{x}$	$\tilde{x}$	σ
SPFU.1	2.17	2.00	1.17
SPFU.2	1.89	2.00	0.97

(c) Self-predicted future use.

	$\bar{x}$	$\tilde{x}$	σ
AC.1	2.13	2.00	0.78
AC.2	2.04	2.00	1.06
AC.3	1.58	1.00	0.81
AC.4	2.13	2.00	1.05
AC.5	1.79	2.00	0.64
AC.6	1.92	2.00	0.95
AC.7	1.79	2.00	0.82

(d) Aggregation capabilities.

	$\bar{x}$	$\tilde{x}$	σ
WF.1	1.92	2.00	0.81
WF.2	1.96	2.00	0.84
WF.3	1.58	1.00	1.04
WF.4	2.04	2.00	0.93
WF.5	1.67	1.00	0.90
WF.6	2.50	2.00	1.15
WF.7	2.17	2.00	0.99

(e) Work fragmentation.

Work Fragmentation: in general, participants found the system able to support work fragmentation. Responses regarding the awareness of available team members (Table V-(e), WF.3) and team composition (Table V-(e), WF.5) scored very well. However, it needs to be highlighted that even if participants found the ABC4GSD system able to release the mental overhead related to the handling of multiple tasks (Table V-(e), WF.7), they scored less its ability to handle work fragmentation (Table V-(e), WF.6). Reasons for this result will be further discussed in Section VIII.

VII. THREATS TO VALIDITY

Participants affiliation: participants in our experiment were recruited from members of the SUCCESS research centre[8]. While all participants satisfy the selection constraints we set, it does potentially represent a bias, even though it is a common practice for pilot studies such as ours.

Learning effect: the three scenarios that we constructed for our experiment have been treated as independent experiments for the purposes of this study and our analysis. To avoid bias in the results related to the execution sequence, and thus to mitigate the impact of the learning effect, scenarios were randomly assigned using latin square (with 12 participants we performed two instances of each permutation of the 3 scenarios).

Process validity: one threat that had to be carefully considered is related to the realism of the GSE scenarios that we designed for our experiment. To assure that the scenarios would be representative of real GSE team situations they were designed and agreed by the authors following an iterative process. This was intended to guarantee that the participant would be exposed to the key challenges investigated in this research, i.e., aggregation capabilities and work fragmentation, while at the same time build an educated opinion of ABC4GSD. Moreover, these scenarios were designed to mimic three completely different situations representative of a GSE environment, i.e., organization of a distributed GSE team project, preparation for a presentation and sharing resources, and implementation work.

Results validity: being a pilot experiment, the number of participants was limited. Even though no statistical significance or scalable or generalizable results were sought by this study, we analyzed the reliability of the evaluation by looking at the internal consistency of all the investigated variables using the Cronbach's alpha coefficient [4]. As can be seen from Table VI, even considering the relatively small number of participants, the α coefficient is good for all the variables but one. This allows us to have confidence that the results, far from being scalable or generalizable, are yet indeed reliable giving us more confidence when confirming our research hypotheses. The threshold for acceptable results is 0.7 [4]. In our case, apart from the self-predicted future use variable (calculated as less than 0.5, i.e., unacceptable), the remaining variables show a good internal consistency (i.e., greater or equal than 0.8).

TABLE VI. CRONBACH'S ALPHA FOR THE INVESTIGATED VARIABLES.

Variable	α
Perceived Usefulness	0.82
Ease of Use	0.81
Self-predicted future use	0.43
Aggregation Capabilities	0.82
Work Fragmentation	0.82

VIII. DISCUSSION

The purpose of the ABC4GSD system is to smoothly introduce the activity-based computing paradigm in to GSE team environments to provide a tool able to tackle the challenges of distributed collaboration. Among these challenges, we concentrated in this study on investigating the ability of ABC4GSD to support aggregation of human and digital resources through the activity metaphor as well as the ability of the system to handle interruptions by means of the activity life-cycle mechanism of suspension-resumption. In this section we will discuss our findings from the pilot study of ABC4GSD and outline key areas for future research we are planning based on these results.

Overall, the experiment has indicated promising results about our overall approach. In general, almost all participants scored ABC4GSD and its activity-based co-ordination and co-operation metaphor positively for each question. In particular, the concept of activity was well received and participants quickly understood the benefits the system brings in in terms of facilitating collaboration, delegation, and resource sharing; the activity life-cycle mechanism and awareness have been also appreciated and one participant in particular reported: *"I liked the ability to resume tasks [...], this makes the whole activities concept portable"*; *"I liked the fact that I can follow what I have left before and what others are doing at the same time"*.

[8] http://www.swinburne.edu.au/ict/success/

However, several critical aspects have emerged, especially from our analysis of the open-ended questionnaires. These have helped us to provide a justification to some of the scores given. We identified two major factors that have had a stronger impact on participant perception: usability aspects and effectiveness of the notification system.

Some encouraging results were reported in relation to the perceived ease of use of ABC4GSD. However, we observed from our video analysis that most participants initially struggled to perform some of the tasks related to the scenarios. One of them reported the impression of being partially constrained in the process of creating an activity and aggregating the required resources (mentioned verbatim in the open question answer). Another notable result not visible from the aggregated statistics is that users on average improved their scoring regarding the ease of use of the system throughout the scenarios. This demonstrates how, even given the novelty of the activity metaphor, participants managed to quickly use the system and were able to achieve their goals. On the other hand, almost all participants gave suggestions and comments regarding possible improvements in terms of usability. Most we plan to address in our next implementation iteration. As an example, we will revise the currently-used common interface design by adding buttons directly on the main interface to allow a more intuitive access to the main functionalities. We also want to provide a smoother mechanism to aggregate digital resources (e.g., drag-and-drop capabilities) from the desktop and other open applications.

Furthermore, it became clear that participants following a specific order in the scenarios were less 'shocked' by the new interaction mechanism than those that started the experiment with the second scenario, i.e., *Fixing a bug*. This is certainly related to the amount of novel interactions presented to the users at once compared to having a gradual introduction to the system. The tutorial, in fact, turned out to be less effective then what was foreseen in terms of explaining to participants the available functionalities, opposed to the more hands-on approach provided by carrying out the scenarios. Other researchers have lessened this issue by guiding the participants through a training session in which they ask them to mimic the experimenter's operations directly by using the system. When we repeat our study with more participants we may try this approach as well.

Regarding the ability to handle interruptions, the scenario of *Fixing a bug* was meant to test the ability of the system to make the user aware of a situation directly concerning their work, i.e., the creation of an urgent activity involving the user. The results from this interruption scenario was less encouraging than the overall ones. In fact, only 50% of the participants were considered fully aware of the event by noticing the notification pop up and acting accordingly by either consciously delaying it or promptly resuming the new activity. Three of the participants' attention was captured by the new item in the activity list rather than the notification message, and one of them noticed the change of status of the user Paolo that created the activity triggering his interest in looking for the cause of such event. Thus, 33% of the participants were classified as partially aware of the situation. And 17% (2 out of 12) did not notice any changes in the user interface and therefore were classified as unaware. Such a result unveils a serious issue in the awareness mechanism chosen for the system. A post analysis of the captured screen videos showed that during the second scenario most participants were fully concentrated in solving the problem related to the main activity thus following the instructions received. This is a situation that is very close to a real working environment in which practitioners try to avoid distractions in order to be focused on their respective tasks. The notification pop up was found to be not sufficiently effective in communicating crucial information to the user. Even though some users suggested that the problem could be related to the fact that such pop ups were automatically fading out, we believe that more work needs to be dedicated to finding a more effective awareness mechanism, endorsing one of the key findings of the survey on GSE organization presented in [31].

Moreover, it is interesting to note how throughout the scenarios, Yarosh's statement about the diverse appropriation of the activity concept observed in different users was confirmed in our experiment. In their work, Yarosh et al. [38] described the different patterns of usage they identified in their case study and explained how the lightweight idea of an activity can become a powerful tool. This is done by not constraining the end user to specific workflows and instead providing a flexible mechanism that can be utilized to fulfill the diverse needs of practitioners. In GSE, this flexibility can be leveraged to a further level that will be one of the foci of our next experimentation round. Culture and language represent two of the distances identified the most by research in the GSE context together with time and location. Organizations, for instance, might have a very hierarchical structure opposed to others with an almost flat one. Once put in the position of collaborating, these structurally diverse realities will find a strong clash with regards to their organizational culture. Usually such a situation would be led by aligning one of the two parties to the other. However, a different solution could look at the possibility of allowing the two entities to organize their work according to their culture and support solely the provision of a bridge between the two representations. A system like ABC4GSD can provide support to these kind of differences in the culture by allowing different representations of the same common activity. Such a scenario, as long as the consistency in terms of the common motive is granted, would allow differences, for instance the example related to organizational culture just exposed, to coexist.

Finally, the current prototype lacks communication features, which will be tackled in our next implementation iteration. Nonetheless, two participants overcame this limitation in the second scenario (i.e., *Fixing a bug*) by using the text document provided to notify Paolo (i.e., the creator of the activity requiring urgent attention) that the bug had been fixed. Even though confirming the shortage, the initiative taken by the two participants confirmed the ability of ABC4GSD to provide a flexible tool.

IX. Summary

The ABC4GSD system is a middleware technology that aims at transparently introducing the principles of ABC in regular operative systems by making use of the theoretical constructs provided by activity theory in order to lessen known coordination, collaboration, communication, and awareness

issues proper of computer-mediated cooperation, which are exacerbated in the context of GSE. ABC4GSD has been designed to support Norman's vision of releasing users from mental overheads when interacting with the computer. After performing the user evaluation herein reported, we are confident in stating that the direction taken can and does provide a concrete support for GSE users, which constantly face challenges ascribable to cultural, linguistic, temporal, and geographical distances.

The evaluation has shown that common problematics proper of computer-mediated collaboration like sharing resources with remote colleagues and handling interruptions while working can be effectively supported by a metaphor grounded in activity theory. Research however needs to be done on how to quickly introduce this novel metaphor in a regular operative system as the evalution has shown that users tend to apply their existing knowledge based on the file and application metaphor. Feedbacks collected unveiled many usability concerns that might have tampered with a faster appropriation of ABC4GSD, which will be tackled in the next implementation iteration. Besides including communication functionalities, in our future research we will focus more specifically on investigating the system capability of concurrently supporting diverse representations of the same activity, a problem this, which so far is often mitigated by the adoption of a middleman in charge of liaising with teams distant in terms of culture and language.

ACKNOWLEDGMENT

The authors would like to thank all the participants to the experiment. We also thank the IT support team of Swinburne University of Technology for helping in deploying the infrastructure.

REFERENCES

[1] M. A. Babar, D. Winkler, and S. Biffi. Evaluating the Usefulness and Ease of Use of a Groupware Tool for the Software Architecture Evaluation Process. In *Empirical Software Engineering and Measurement. First International Symposium on*, 2007.

[2] J. E. Bardram. Activity-based computing for medical work in hospitals. *ACM Trans. on Computer-Human Interaction (TOCHI)*, 2009.

[3] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson. FASTDash: a visual dashboard for fostering awareness in software teams. *Proc. of the SIGCHI conference on Human Factors in computing systems*, 2007.

[4] J. M. Bland and D. G. Altman. Statistics notes: Cronbach's alpha. 1997.

[5] G. Booch and A. Brown. Collaborative development environments. *Advances in Computers*, 2003.

[6] G. Convertino, D. C. Neale, L. Hobby, J. M. Carroll, and M. B. Rosson. A laboratory method for studying activity awareness. In *Proceedings of the third Nordic conference on Human-computer interaction*, 2004.

[7] D. Damian, F. Lanubile, and T. Mallardo. On the need for mixed media in distributed requirements negotiations. *IEEE Trans. on Sw. Eng.*, 2008.

[8] D. Damian and D. Moitra. Guest Editors' Introduction: Global Software Development: How Far Have We Come? *IEEE Software*, 2006.

[9] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.*, 1989.

[10] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Fine-grained management of software artefacts: the ADAMS system. *Software: Practice and Experience*, 2010.

[11] C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 1991.

[12] J. Herbsleb. Global software engineering: The future of socio-technical coordination. *Future of Software Engineering*, 2007.

[13] S. Hupfer, L.-T. Cheng, S. Ross, and J. Patterson. Introducing collaboration into an application development environment. *Proceedings of the ACM conference on Computer Supported Cooperative Work*, 2004.

[14] M. Jiménez, M. Piattini, and A. Vizcaíno. Challenges and improvements in distributed software development: a systematic review. *Advances in Software Engineering*, 2009.

[15] O. Laitenberger and H. M. Dreyer. Evaluating the usefulness and the ease of use of a Web-based inspection data collection tool. In *Software Metrics Symposium. Metrics. Proceedings. Fifth International*, 1998.

[16] M. Lang and J. Duggan. A Tool to Support Collaborative Software Requirements Management. *Requirements Engineering*, 2001.

[17] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno. Collaboration Tools for Global Software Engineering. *IEEE Software*, 2010.

[18] F. Lanubile, T. Mallardo, and F. Calefato. Tool support for geographically dispersed inspection teams. *Software Process: Improvement and Practice*, 2003.

[19] A. N. Leont'ev. *Activity, Consciousness, and Personality*. Prentice-Hall, 1978.

[20] M. Mangan, M. Borges, and C. Werner. A middleware to increase awareness in distributed software development workspaces. *Proc. WebMedia and LA-Web.*, 2004.

[21] P. Mukherjee, A. Kovacevic, M. Benz, and A. Schürr. Towards a Peer-to-Peer Based Global Software Development Environment. In *Software Engineering*, 2008.

[22] J. Noll, S. Beecham, and I. Richardson. Global software development and collaboration: barriers and solutions. *ACM Inroads*, 2010.

[23] D. Norman. *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. 1999.

[24] J. Portillo-Rodriguez, A. Vizcaíno, M. Piattini, and S. Beecham. Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology*, 2012.

[25] M. Purvis, M. Purvis, and B. Savarimuthu. Facilitating collaboration in a distributed software development environment using P2P architecture. *Int. Workshop on Agents and P2P Computing (AP2PC '06)*, 2008.

[26] T. Rattenbury and J. Canny. CAAD: an automatic task support system. In *SIGCHI conf. on Human factors in computing systems*, 2007.

[27] K. Schmidt. The critical role of workplace studies in CSCW. 1999.

[28] B. Sengupta, S. Chandra, and V. Sinha. A research agenda for distributed software development. *Proceedings of the International Conference on Software Engineering*, 2006.

[29] I. Steinmacher, A. Chaves, and M. Gerosa. Awareness Support in Global Software Development: A Systematic Review Based on the 3C Collaboration Model. In *Collaboration and Technology*. 2010.

[30] J. Suzuki and Y. Yamamoto. Leveraging distributed software development. *Computer*, 1999.

[31] D. A. Tamburri, P. Lago, H. V. Vliet, and E. Di Nitto. On the Nature of GSE Organizational Social Structures: An Empirical Study. In *Int. Conf. on Global Software Engineering (ICGSE)*, 2012.

[32] P. Tell and M. A. Babar. Requirements for an infrastructure to support Activity-Based Computing in Global Software Development. In *Int. Conf. on Global Software Engineering Workshop (ICGSEW)*, 2011.

[33] P. Tell and M. A. Babar. A Systematic Mapping Study of Tools for Distributed Software Development Teams. Technical Report TR-2012-161, Oct. 2012.

[34] P. Tell and M. A. Babar. Activity Theory Applied to Global Software Engineering: Theoretical Foundations and Implications for Tool Builders. In *Int. Conf. on Global Software Engineering (ICGSE)*, 2012.

[35] V. Trapa and S. Rao. T3 - tool for monitoring agile development. *Agile Conference*, 2006.

[36] S. Voida, E. D. Mynatt, and W. K. Edwards. Re-framing the desktop interface around the activities of knowledge work. *Proc. of the annual ACM symp. on User Interface Software and Technology (UIST)*, 2008.

[37] J. Whitehead. Collaboration in software engineering: A roadmap. *Future of Software Engineering (FOSE '07)*, 2007.

[38] S. Yarosh, T. Matthews, T. P. Moran, and B. Smith. What Is an Activity? Appropriating an Activity-Centric System. In *Proc. of the IFIP TC 13 Int. Conf. on Human-Computer Interaction: Part II (INTERACT)*, 2009.