Activity Theory applied to Global Software Engineering: Theoretical Foundations and Implications for Tool Builders

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

Abstract—Although a plethora of tools is available for Global Software Engineering (GSE) teams, it is increasingly being realized that the majority of tools have several inherent limitations. We have proposed that activity based computing can be a promising alternative to build tools for GSE. However, significant effort is required to introduce a new paradigm; there is a need of sound theoretical foundation based on activity theory for addressing challenges faced by tools in GSE. This paper reports our effort aimed at building theoretical foundations for applying activity theory to GSE. We analyze and explain the fundamental concepts of activity theory, and how they can be applied by using examples of software architecture design and evaluation processes. We describe the kind of data model and architectural support required for applying activity theory in building tooling infrastructure for GSE and describe a proof of concept prototype.

I. INTRODUCTION

Global Software Engineering (GSE, also known as Global Software Development (GSD)) paradigm is characterized by geographical, temporal, cultural, and linguistic distances. These kinds of distances among GSE team members usually result in several kinds of challenges such as miscommunication caused by cultural and language issues, delays in decision making, and coordination overhead. Appropriate tool support is considered an important mechanism to enable GSE teams to mitigate the potential negative effects of these challenges of GSE projects. Hence, one of the key challenges of GSE researchers and practitioners is to provide GSE teams with appropriate tools to help them in effectively and efficiently perform their tasks, which are likely to require an increased degree of communication, coordination, and collaboration – commonly known as 3C dimensions [10] – among GSE team members due to the aforementioned distances. Given the critical role and importance of appropriate and adequate tool support for GSE teams, several dozens of solutions have been proposed by both academia and industry as reported in literature [18][34][29].

However, almost all of the available GSE tools have been designed and developed based on a file- and application-oriented desktop metaphor from the ’70s that was not aimed at providing inherent support for the kind of collaboration and coordination usually required by GSE teams. Limitations of the desktop paradigm have been well documented [15]; it has been argued and shown that such desktop metaphor does not provide inherent support for communication, coordination, and collaboration. Hence, it is being increasingly realized that there is a need of providing flexible collaborative environments; need which is even more perceived in contexts like the GSE one where, due to lack of physical presence, the 3C dimensions are fundamental characteristics that have to be facilitated by appropriate technological support [30]. Researchers have made alternative proposals for providing collaborative desktop environments. For example, Booch and Brown [7] defined it as “[...] a virtual space wherein all the stakeholders of a project – even if distributed by time or distance – 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.” Even if the authors explicitly suggest that the web platform appears to be the most suitable environment for hosting CDEs, different promising approaches are being experimented. An approach tested by industry and widely studied by researchers is the one based on activities. In these solutions, the fundamental metaphor provided to organize work is focused on the concept of activity rather than being strongly tightened to the ones of file and application.

Examples can be seen in 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. Following this concept, Yarosh and his colleagues from IBM [35] 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 introduced in [33], and the ABC Framework described in [4]. In these cases, the approach followed is the one initially introduced by Norman in the ’90s called Activity-Based Computing (ABC). In his book [25], he states that “[...] the basic idea is simple; make it possible to have all the material needed for an activity ready at hand, and

\(^1\)Gnome Shell has been released with Gnome version 3 in April 2011 (http://www.gnome.org/)
available with little or no mental overhead." Thus, the core concept of ABC is to provide an automatic, seamless, and non-intrusive support for the conceptual framework described in the activity theory.

Having realized the inherent limitations of the existing paradigm of building GSE tools and getting inspiration from the literature on the successful application of Activity Based Computing (ABC) for providing tools to support communication, coordination, and collaboration in disciplines such as healthcare, we decided to explore the viability of building tool support for GSE using the ABC paradigm [31]. Our initial effort brought some encouraging results; we successfully identified a set of significant high-level requirements for building an infrastructure that can enable GSE tools to leverage the ABC principles [30], and incrementally designed and built a Proof-of-Concept (PoC) prototype [31] using the conventional experimental computer science approach. However, we also realized that there was an urgent need of a theoretical framework to sustain the efforts for building tool support that can successfully introduce the ABC paradigm in GSE. Our realization of the need of a solid foundations for using the ABC in GSE was also reinforced by the questions and comments we received during the presentations of our initial work along these lines [30][31]. We also noticed that the activity theory has been successfully used for studying workplace practices and building tool support in other disciplines, such as Computer Supported Cooperative Work (CSCW), after building solid foundations appropriate to a particular discipline’s needs. Hence, we decided to undertake an effort to provide theoretical foundations for building tool support for GSE teams by applying the concepts and principles of the activity theory.

The paper reports the initial outcomes of our effort aimed at systematically and critically analyzing the activity theory and its applications in other disciplines with the intention of gaining deep understanding of the theoretical foundations and principles of the activity theory, demonstrating its suitability for describing activities in GSE processes, and building GSE tools. However, another important goal of this paper is to motivate the GSE community to gear their efforts towards providing solid theoretical foundations for supporting GSE processes and tools. Based on our observations from other disciplines, where the activity theory has been used for tool support, we are convinced that there will be a significant amount of community based effort required to provide sound theoretical foundations for incorporating the ABC paradigm in building the next generation of GSE tools. We assert that the activity theory can provide an interesting set of research challenges for GSE researchers and practitioners, and especially tool builders.

The main contribution of this paper to the global software engineering research and practice is threefold:

- we systematically and critically analyze the theoretical concepts and models underpinning the activity theory for assessing their viability to provide alternative computational constructs for building the next generation of GSE tools;
- we provide an elaborative discussion and detailed argumentation to demonstrate how the conceptual models from the activity theory can be applied for structuring GSE activities and guiding the GSE tool building efforts by using examples of two processes from the software architecture discipline; and,
- we present and discuss different aspects of a data model, a high level architecture, and a proof of concept implementation of an infrastructure based on the activity theory theoretical constructs; we assert that these three elements from our work can have interesting implications for GSE tool builders.

II. BACKGROUND AND MOTIVATION

GSE has become an established software development paradigm. An increasing number of companies are either getting their software developed by geographically distributed teams or relocating their software development centers at different geographical locations to gain several perceived benefits such as access to a larger number of skilled personnel at relatively low cost, or reduced development time as a result of ‘follow the sun’ strategy [12]. Adequate tool support is essential to successfully support GSE teams [18]. The Software Engineering (SE) community has developed several dozens of tools for supporting distributed development teams, and many GSE projects have adopted tools which were developed and used by the Computer Supported Cooperative Work (CSCW) and Open Source Software (OSS) communities (e.g., [18], [8], [34], and [29]). Researchers have reported extensive surveys of tools for GSE teams [34]. For example, an overview of commercial and open source technologies and tools for distributed teams has been reported in [26]; Lanubile et al. have reported an overview of collaboration tools for global software engineering in [18]; Sarma et al. [27] have reviewed coordination technologies and proposed a framework to categorize them; and, a classification driven by the software life-cycle has also been proposed in [22]. Furthermore, some researchers have reported systematic literature reviews, e.g., modeling tools have been analyzed in [8], and, the authors of [29], have systematically identified and reported awareness support for GSE. A significant number of publications describe tools developed to enhance particular development processes (e.g., inspection meetings in [19]; requirement management in [17]) or to incorporate information from different sources into monitoring applications (e.g., T3 presented in [32]; FastDash described in [6]).

However, one of the main challenges that has been identified in the context of GSE is the lack of integration [14]. Throughout the software life-cycle many diverse tools are used and the improvement of one of them can hardly have a broad enough impact to truly lessen the GSE related issues. Encouraged by its application in different domains, we propose the use of the activity theory as a gluing framework to provide a powerful tool to both structure and describe activities in GSE processes, and support and facilitate tools integration.
The activity theory has been used as a conceptual framework for modeling human activities. The work of Barab et al. [3], is just an example of these attempts in which the activity theory has been used to analyze systematic tensions (i.e., theory versus practice and teacher-centered instruction versus student-directed learning) in a course for providing insights on how to improve teaching and support learning educators. Another example closer to software engineering can be found in [9]; de Souza and Redmiles demonstrated the application of models provided by the theory to study collaborative work of a large-scale software development team.

In her book [24], Nardi, with the contribution of other authors of profound works like Kuutti, Kaptelinin, and others, proposed and analyzed the feasibility and benefits of applying activity theory to Human-Computer Interaction (HCI). The activity theory evolved in such a way that it became not only an analytical conceptual framework used to describe human activities but also a source of inspiration providing insights on how to design systems supporting computer-mediated interactions with a new perspective. The concept of activity as a unit of analysis therefore, when applied to technologies, became a computational construct able to provide a metaphor more suitable for supporting collaborative work compared to the old-dated legacy from the '70s. However, its adoption differed greatly. Lotus Activities, for example, studied by Yarosh et al. [35], has been designed for providing an ACC system in which activities were used for their aggregating properties were able to support work fragmentation. Whereas, designed to support medical work in hospitals, which is characterized by mobility and collaboration, the ABC framework represent an example of ABC system. In [5], Bardram describes how the system was found very effective in supporting physicians by providing a robust replacement to the application-oriented computing paradigm. He states that one of the main success factors should be ascribed to the way collaboration was supported; in fact, through the ABC paradigm, collaboration was not anymore a characteristic that had to be added by means of external applications rather a basic feature provided upfront by the system itself.

In this work we will show, the activity theory can not only be used to thoroughly study collaborative work (e.g., [9]) but also, using the original concepts described by Leont’ev [20], to both structure activities a priori and design an abstract model for being the foundation on top of which technological support can be build. Our purpose is therefore twofold: to raise awareness of activity theory within the GSE community, and to show how it can provide a solid theoretical foundation supporting tool building efforts. Furthermore, we will present an infrastructure grounded in the activity theory, which can support application interoperability and high-level requirements, which needs to be addressed to support the need of replacing the absence of physical presence with computer-mediated means [30]. Achievement that would, among other benefits, release tool builders from constantly re-introducing these features that are clearly not supported by the current desktop metaphor.

### III. Activity Theory

Conceptualized by Vygotsky and further heavily improved by Leont’ev [20], the activity theory became quickly popular after Engeström [11] brought it out from the Soviet Union. One of Engeström’s main contributions is what he called activity system: a schematic representations of the theory detailed by Leont’ev (Figure 1).

![Activity System](image)

In this model, a human activity is described by relating the subject (individual or group), equipped by an instrument (cognitive or material), to an object: the target of the activity, which, after realization is transformed into an outcome. One of Leont’ev’s main addition to Vygotsky’s unit of analysis (i.e., object-oriented action mediated by cultural tools) is concretized by the addition to the theory of the social relation of the subject inside a community. An addition this, which exposes the model to the need of relating the community to both the subject and the object. This is realized by rules and division of labor. The former, identifying those “logical (and mathematical) system of laws, rules and regulations” [20] that govern the subject inside a community. The latter, engendering the natural separation and distribution of task inside a community necessary to achieve the object. See hunting example [21].

Further, Leont’ev contributed to the theory by including to the hierarchical structure a third level: the operation, which are automatically performed activities driven by conditions. Therefore, as summarized by Engeström [11], Leont’ev’s hierarchical model of human activity is composed of three level: activity, action or process, and operation. Each of which, see Figure 2,

![Hierarchical Structure](image)

...driven by a different need, namely: motive, goal or purpose, and condition. This description slightly differs from Engeström work [11], but appears to be closer to the original description in which a goal is also referred to as a conscious purpose.
and an action as a process. See hunting [21] and driving [20] example.

IV. APPLYING THE ACTIVITY THEORY FRAMEWORK FOR STRUCTURING GSE ACTIVITIES

Based on the conceptual model described in the previous section, we demonstrate how the activity theory can be used to describe different activities and processes of developing software in the context of GSE. For the piece of work described in this section, we identified published articles describing, detailing, and decomposing software engineering activities. Then, those activities were mapped onto concrete instances of the activity theory conceptual model. In particular, after merging and relating together activities identified in three articles (i.e., [13], [1], and [2]) by means of the hierarchical structure, we have detailed the activity systems for each of the presented granularity levels. Even though the boundaries chosen to frame the activities might seem ad-hoc, we assert that the use of the tools provided by the activity theory is consistent with the original theory, useful, and intuitively meaningful. We believe that the granularity of the analysis, which results in a specific instantiation of either the hierarchy or the activity system, implicitly reveals the lens that an observer is using to describe the specific human activity.

This application of the activity theory immediately unveils its usefulness for improving coordination – constituent of the 3C model [10]. First of all, a reader will easily be able to relate the hereafter presented content with Mintzberg’s six coordination mechanisms [23] (i.e., mutual adjustment, direct supervision, and standardization of work processes, of outputs, of skills, and of norms) by carefully considering the flexibility of the hierarchical structure of activities (Figure 2); and the role of the concepts of community, division of labor, and rules of the activity system (Figure 1). Secondly, equally important but not present in Mintzberg’s organizational configurations framework as irrelevant for his analysis, this application of the activity theory evenly supports the coordination of the single individual (see [35]).

Hofmeister et al. have proposed a general model of software architecture design by merging five industrial models [13]. The new model of software design process provides three high-level activities: architecture analysis, architecture synthesis, and architecture evaluation. In a similar piece of work, Ali Babar and Gorton have proposed a general model of software architecture evaluation consisting of five commonly found activities in four well known architecture evaluation methods [1]. Later on, Ali Babar [2] showed how the five activities of the software architecture evaluation process general model could be broken down to outline concrete sub-tasks, major participants involved, and main groupware tools required for each of the software architecture evaluation process activities in the context of GSE. We use the software architecture design process general model [13] and the groupware-supported software architecture evaluation process for GSE teams [2] to demonstrate how the activity theory framework can be applied to structure activities of a software engineering process of geographically distributed software development teams.

Starting from the general model of software architecture design [13], we describe how both the activity system and the hierarchy model can be applied to map a software architecture design activity when instantiated to describe a concrete example. Analyzing the situation from the perspective of a hypothetical company that has been asked by stakeholders to design a software architecture, Figure 3 shows how the activity can be framed. The subject involved in this activity is thus composed of all the entities of the company (such as project manager, evaluation manager, and architect) involved in the design – the objective. The design can be achieved by focusing the collective knowledge possessed by the community – the object. Such transformation can only be achieved by appropriately applying and shaping the instrument: the framework proposed in [13]. Here we can see the importance of the community, which by providing capabilities, experience, and competences, can support the overall activity by providing knowledge. Hence, stakeholders are part of the community together with the company staff, which, in one way or another, following a proper division of labor, may contribute to the realization of the motive. Finally, interactions between subject and community are described as driven by rules imposed by the context (such as company or ethical rules). Once properly

![Fig. 3. Instantiation of activity system model to describe an application of the software architecture design framework.](image)

![Fig. 4. Instantiation of hierarchy model to describe an application of the software design framework.](image)
activities identified. A reader may argue that these activities cannot be considered operations. However, using the lens of a specific subject involved, e.g., the project manager, we can see how these can, in fact, be considered automatically executed by the sub team of the company to whom such work has been delegated.

Now we can utilize the lens of an evaluation manager. Figure 5 clearly depicts his/her driving motive: evaluate a candidate solution. Hence, at this stage, the focus is narrowed down and the community only considers a subset of the original community in which all the company staff was involved together with stakeholders. Being a theoretical exercise without real data, it is nontrivial to appropriately identify the involved entities. However, we can assume that it will be a smaller, not necessarily properly contained, subset of the general entities presented in Figure 3. Hence, the object of the evaluation manager’s activity is a candidate architecture, which, by means of the evaluation framework proposed in [2], can be transformed into the objective – the validated architecture. In the previous analysis, this objective represented the goal of an evaluation action, part of the design activity; however, we can now elevate what previously was a goal, using the lens of the evaluation manager, to a motive thus the driving force of his/her activity (Figure 6). This said and keeping in mind the previous argumentation, we can safely continue our decomposition following the sub-tasks suggested in the framework proposed in [2]. Therefore, the action of preparing and managing the results has been further decomposed into its constituting operations of artifact interpretation, report preparation, and result presentation.

The aforementioned decomposition can be further applied to a point at which the activity is performed through actions facilitated by technologies (Figure 7). In such case, the experi-

ence of the person acting could allow these actions to be either considered action or operations (see driving example [20]).

However, it is recommended to avoid further decompositions as in contrast with the activity theory purpose of being a philosophical framework for studying human practices as development processes.

V. BUILDING TOOLS FOR GSE BY APPLYING THE ACTIVITY THEORY FRAMEWORK

In this section, we describe how the activity theory conceptual models can be leveraged for building the next generation of GSE tools. Our application of the activity theory for building GSE tools also has implicit and explicit indication of the potential ramifications for the GSE tool builders (i.e., researchers, practitioners, and tool vendors). First of all, we briefly present and discuss the current approaches that are based on the activity theory to set the context. The brief discussion about the current approaches to building tools using the concepts from the activity theory is expected to provide inspiration and stimulate ideas about building activity enabled tools for GSE. Later on, we presents a data model that supports the framework introduced in Section III by incrementally relating different elements of the data model structure to the models presented in Figure 1 and 2. Furthermore, we also present and elaborate a high-level architecture, which is providing an overall roadmap to our effort to build an infrastructure based on the concepts and principles described in the previous sections. However, we believe that the presented data model and architecture are general enough to be leveraged by other tool builders interested in exploring the ideas of incorporating the activity theory in their tool building efforts.

Yarosh et al. [35] have stated: “[w]e have shown that ACC systems have the potential to support knowledge workers by providing a shared Activity constructs for coordinating work units, producing deliverable outcomes, gathering information, and sharing knowledge”. Indeed, Lotus Activities is able to provide support for structuring the natural work fragmentation. Even if comforting, we are interested in researching the possibility of pushing further by leveraging the generated information also for facilitating users’ context switches, which have already been identified as main source of frustration among practitioners [28]. As envisioned by Norman [25] and later shown by Bardram [4], the computational unit of the activity can be used to reduce mental overhead by providing a
mechanism to automatically support users in their alternation of activities. Bardram’s ABC framework is driven, in particular, by two principles that enable the support for activities ‘life-cycle’: suspension and resumption. The interaction with the system supported by the ABC framework is not anymore driven by the execution of either applications or files, rather by the resumption, and consequential suspension, of activities.

One challenge can be how to decide what should be considered an activity in the context of GSE, and how such activities can be identified. Both of these questions can be answered by using the description of an activity given by Yarosh et al. in [35]: “[a]s a unit of analysis, an activity is defined as a coordinated set of actions by people towards a common objective, mediated by tools and subject to situational constraints (e.g. [16]). But in work settings, knowledge workers have objectives at different levels of granularity, some objectives being in service of other objectives (e.g. creating a demo as part of preparing for a review as part of managing a project). […] We have identified some common patterns of use for Lotus Activities, but users often structured their instances of these patterns in different ways. This presents a challenge to designers trying to provide guidance for defining Activities.”

As we described in Section IV, these statements respect the activity theory. Different subjects perceive actions (or activities) in different ways; they have different goals or motives. When this occur, as we have shown in Figure 6, the action is elevated to the level of activities (or, when dealing with activities, a higher activity). There are multiple levels of granularity in which activities are perceived depending on the subject. Furthermore, the main idea behind ABC, as introduced by Norman [25], is to release users from mental overhead; therefore, a system should be able to automatically integrate the concept of activity, rather than overloading users with the unnecessary concern of strictly defining activities in terms of actions and operations. This represent our first step in supporting the claim that a strict application of the separation formalized in the activity theory becomes less useful when applied to the process of building tools based on it. Moreover, contrary to Bardram’s approach for supporting healthcare environments [4], Yarosh et al. identified a need for higher flexibility in the definition and support of activities. Indeed, in hospitals, there is no harm in providing a completely shared view of activities (i.e., WYSIWIS paradigm); however, in software engineering, this cannot be applied. Artifacts (e.g., source code files) are constantly evolving, and hardly all participants need to be aware of every modification and want to have resources presented in a standardized way. These observations and deliberations led us to a different design decision emphasizing the freedom, in terms of visualization, which needs to be granted to software engineering practitioners. They need to be able to independently manage applications and handle artifacts without necessarily having these being part of a common representation of activities shared by its participants. If not engaged in a concrete synchronous collaboration, users require activities representation to be personalized thus fitting their needs in a flexible manner.

A. Supporting Data Model

One of the early steps in any effort aimed at providing tool support is usually the design and assessment of a suitable conceptual data model. In this section, we present and discuss the conceptual data model aimed at supporting the activity theory conceptual constructs and their relationships suitable for building GSE tools. We believe that a system based on the activity theory, to be effective and quickly appropriated, must not deprive its users from the freedom and ease of use suitable to their current habits of using tools developed using the file- and application-based paradigm. The data model supporting the activity theory, tailoring these preliminary analysis of the context, is depicted in the UML diagram presented in Figure 8 and detailed in the following.

As previously argued, the only element of the hierarchy explicitly considered is the activity. In the case of actions, these would be presented as subelements of the activity they are part of; in the case that such action needs to be analyzed, these would become the main activity potentially defined by its own actions. Operations are handled in a similar way as they are either directly supported by tools or ‘automatically’ accomplished by someone different from the subject of the activity. This design decision is closer to the Norman’s fundamental idea of releasing users from mental overhead [25], and, at the same time, supports the need to be flexible for, as empirically identified in [35], activities in software engineering are hard to define and utilized by users for different purposes.

The central element of the model is the Ecology entity that, by indirectly linking activities to users, keeps track of all assets associated to an activity by each user. From the diagram we can see how every instance of the Ecology entity links one activity to one of its participants and to all the assets used (i.e., applications and artifacts). Through the Ecology entity, the necessary flexibility previously described is obtained. The Activity entity captures the core concept of the activity theory and contains information shared by all participants (e.g.,

---

motive of the activity by means of its description). To support the flexible structured nature of activities previously detailed, activities relations are facilitated by the Relationship element. Participants are modeled through the User entity by collecting all details needed to identify a physical person interacting with the system. An association class between the User and the Activity entities is provided to capture all information necessary to specify, for example, the role or the status of each participant in each activity. Asset is the superclass mapping digital artifacts and applications common information. Properties and unique information regarding assets connected to users are encoded in the Property association class, further supporting the flexibility initially discussed. The Artifact entity collects all digital artifacts used in activities (e.g., UML models or source code) as well as links to resources like web-sites or repositories by means of URIs. Whereas, the Application element, stores the ‘need’ that applications have to support. To give 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 or Foxit Reader for those running Windows systems). Therefore, every ‘need’ is mapped at the client-side via a configuration file. Finally, the proxy pattern has been implicitly introduced to allow the same asset (i.e., both artifacts and applications) to be linked to different instances of the Activity and User entities, through instances of the Ecology one, yet maintaining unique properties. Application and Artifact elements are thus connected to the Ecology through the Asset entity that acts as a wrapper hiding the concrete artifact/application from the properties linked to each of its instances.

**TABLE I**  
**MAPPING OF THE DATA MODEL AGAINST THE ACTIVITY SYSTEM.**

<table>
<thead>
<tr>
<th>Activity system</th>
<th>Data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>User</td>
</tr>
<tr>
<td>Object</td>
<td>Ecology</td>
</tr>
<tr>
<td>Outcome</td>
<td>Activity description</td>
</tr>
<tr>
<td>Tool</td>
<td>Application</td>
</tr>
<tr>
<td>Community</td>
<td>Users part of the Ecology</td>
</tr>
<tr>
<td>Rules</td>
<td>-</td>
</tr>
<tr>
<td>Division of labor</td>
<td>Relationship</td>
</tr>
</tbody>
</table>

We can see in Table I, how the elements of the activity system find a representative in the data model. In the following these will briefly detail. The subject is trivially described by the user entity. By collecting all information related to an activity, the ecology element represent the object of the activity; in fact, to obtain an outcome, a user can make use of all the resources shared and aggregated by the ecology. The outcome is presented by the description contained in the activity; the instrument, if not cognitive, is represented by the application element. The community is composed by all users participating to an activity. Further, the division of labor is obtained by the relationships. Finally, the rules are not considered as they are part of the real context identifying the behavior of the subject inside the community.

**B. Supporting High-Level Architecture**

The infrastructure designed to support this data model is based on a client/server architectural style. The main components of the infrastructure are depicted in Figure 9 and hereafter described. The Remote Server is the element responsible for ensuring the persistency and the consistency of the data, it contains the persistence layers and is liable for generating the necessary events to ensure the correct communication of the information throughout the connected clients. The client architecture comprises essentially two elements: the main controller represented by the Activity Manager box and the ABC App Interface. The Activity Manager is the main component in each client workstation; it is a middleware that 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 purpose of the ABC App Interface is to provide a common way to interact between the system and the applications; applications have to adhere to this specific interface to participate in the system. This component, besides providing the functionalities to send/receive messages and subscribe/unsubscribe to events, defines a list of ad-hoc methods that have to be implemented to have both the Activity Manager control some behaviors of each application, and the applications react to events. Such design allows the automatic integration of response mechanisms. Finally, it has to be noted that all additional client elements are treated as independent application, i.e., concrete external processes. 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 any another one allowing users to use applications normally used by GSE practitioners.

This infrastructure is able, without changing the current desktop metaphor, to introduce the concept of the activity theory.

**VI. PROOF OF CONCEPT**

Having observed the practical viability of leveraging ABC for GSD teams through a trivial proof of concept [31], and having introduced more solid theoretical foundation, we decided
to provide a proof of concept focusing on a concrete situation characterized by a scenario based on the activities described in Section IV. In this paper, we will use the infrastructure to simulate a scenario in which the authors are engaged in the final activity of evaluating the architecture of an hypothetical project: the presentation of the results. Following the concrete scenario used.

Ali, member of the evaluation team, is in his office still working on improving the radar diagram for the presentation, while Paolo, evaluation manager and responsible for the presentation, is rehearsing the slides before moving to the meeting in which he will deliver it.

For the purpose of this proof of concept, the setup of the environment included: a git repository, deployed on the remote server to allow artifacts versioning and storage; applications bound to the infrastructure through the need mechanism, not implementing the interfaces; and, the Client Master application (Figure 10-F), providing a background demon used as entry point to the infrastructure by users, and in the following detailed.

- ABC4GSD Client (Figure 10-A). Developed as an Eclipse RCP application, it is the graphical interface for the Client Master. It exposes administrative functionalities, allows the connection and disconnection from the remote server, and includes some debugging features. It is the only necessary component in all client workstations as it builds the client-side middleware infrastructure allowing other applications to attach to the middleware through itself. One of its views is the one giving the user a clear overview of its situation. This view is composed by the plug-ins described hereafter.

- Activity View (Figure 10-B). The component responsible for displaying the list of the activities a user is participating in, their description, and functionality not yet implemented, the number of online participants.

- Contact View (Figure 10-C). This plug-in is designed to shows the people participating in the activity currently resumed, provides information about the status of users, and, in case, the artifact each of the users is working on.

- Artifact View (Figure 10-D). Used to list all digital artifacts used inside an activity. Provides information about their type and location. A checkbox is also provided to allow users to explicitly decide which artifacts should be launched automatically when resuming the activity.

- Notifications. Shared by all other components, the notification system is able to visualize ad-hoc temporary notifications (Figure 10-E).

Artifacts launched by the middleware follow the suspension/resumption life-cycle imposed by the ABC paradigm described in [30]. Therefore, upon user’s selection of an activities, all applications executed to visualize artifacts are terminated and new artifacts visualized. However, if the same artifact is present in both activities, the one suspended and the one resumed, no operations are performed. This has been done to prevent the considerable time that would take resource-demanding applications like Keynote to close and reopen; thus, improving performance and user’s experience.

With the help of above mentioned scenario, we will describe how the whole activity is facilitated by leveraging this infrastructure. Once Paolo’s system is started, the ABC4GSD Client is displayed allowing the connection to the remote server. The login of an authorized user triggers all plug-ins that start reacting upon a successful connection is performed. The Activity View plug-in requests all the needed information related to activities in which Paolo is participating. After one of the activities is selected, the specific activity is resumed after suspending the one that was – in case – previously running. The suspend procedure is performed by sending an event to all the running application to let them store the information needed for future resumptions on the remote server before closing. The resume procedure is orchestrated by the Master Client after receiving the information about the activity to be resumed. It is performed by executing and initializing all the resources associated with the activity.

The initialization of them is obtained by checking which need should be selected and thus which application should be executed. If integrating the provided interfaces a specific method (i.e., resumeOperation) is called to fetch all data needed for the application to be resumed (such as position and size) and to perform the required subscription to the desired events. As previously mentioned, the applications linked for this scenario are not implementing the interfaces; however, they are executed and requested to handle the specific resource.

Therefore, after the activity Presentation is selected (Figure 10-B), all plug-ins contributing to the user interface are initialized and populated with consistent data. Moreover, all resources selected to be automatically executed are launched (i.e., Notes, Radar.pdf, and Evaluation01...key). This is achieved by checking-out the remote repository for modifications to ensure that the resources loaded are up to date.

Furthermore, all activity participants are enlisted in the Contact View from which Paolo can easily be aware of Ali’s activities. In this example, he is informed that Ali is working on the Radar.pdf resource, which is also part of the Prepare reports activity. Such information is fundamental to tackle the gap between co-located and distributed environments. In fact, Paolo might be wondering if there are some issue with the diagram also used in the presentation and start a conversation with Ali to understand if the presentation needs to be modified to incorporate some last-minute changes. Such flow of information is trivially available through the infrastructure described. The system, instead of constraining tool builders, allows them to make the use of the shared information they believe more appropriate without restricting them in any way. Once the meeting for the presentation arrives, Paolo reaches the meeting room, powers the workstation present in the room and logs into the system to present the work; his last activity is resumed thus loading all related resources, and each

---

\(^1\)http://git-scm.com/

\(^2\)RCP: Rich Client Platform
application adjusted to fit in the new environment.

A reader of this paper may also like to note down that the business logic of all plug-ins described was implemented through the subscription mechanism provided by the middleware. All the updates were obtained by subscribing and automatically responding to the events received with a small additional coding effort.

VII. CONCLUSION

It is important to provide appropriate tool support for bridging the cultural, linguistic, geographical, and temporal distances that characterize GSE teams. Several dozens of tools have been developed with the claim of providing effective and efficient support for GSE teams. However, most of these tools have been developed based on a file- and application-oriented desktop metaphor that was not conceived to provide inherent support to fundamental characteristics of tasks in GSE teams like collaboration, coordination, and communication. In an attempt to find an alternative paradigm for building the next generation of GSE tools, we have started exploring the viability of applying the concepts and models from the activity theory. Our initial effort in this regards has highlighted the need of having a solid theoretical foundations for furthering the work on providing appropriate tool support for GSE teams by leveraging the activity theory as has been done in disciplines like CSCW and healthcare.

In this paper, we have reported a systematic and detailed analysis of the activity theory and its models; we have described their viability for providing the theoretical underpinning for structuring and describing GSE activities and processes at different levels of abstractions, and guiding the development of tools that are activity aware. We have done this by providing theoretical argumentation and practical examples of how the activity theory can be applied not only for analyzing human activities in GSE, but also for modeling and structuring activities a priori – still in accordance with the theory presented by Leontev. To help others understanding how the activity theory can be applied to study and structure SE processes in general, and GSE processes in particular, we have used frameworks reported in the literature incrementally describing the decomposition of the software architecture design activity into its constituent sub-elements detailing at every step the related activity system and hierarchical structure. We have emphasized the importance of understanding the lens used by a subject, which allows describing activities at different levels of granularity.

To support the efforts aimed at building appropriate tool support, by leveraging the concepts and models from the activity theory, we have designed a data model and a high level architecture. We have successfully shown how different elements of the data model and their relationships can be mapped onto the activity theory. The reported high level architecture has been designed to build an infrastructure able to provide support for collaboration and coordination intensive tasks. Such an infrastructure is expected to enable existing and future GSE tools to leverage the activity theory if the tool builders desired so. Based on the reported data model and architecture, we have been incrementally designing and developing a novel ABC based infrastructure aimed at supporting applications integration to address the limitations of tools currently used by GSE practitioners. We have provided the details about the proof of concept prototype implementation of the infrastructure. Our initial experimentation with the reported proof of concept has shown very encouraging results about the practical feasibility of applying the activity theory as
supporting theory for building GSE tools.

In this paper, we have reported our effort aimed at providing theoretical foundations and highlighting its practical implications for applying the activity theory to build GSE tools. We assert that this work makes several important contributions, which have the potential of providing interesting set of research challenges to researchers and practitioners interested in providing tool support for GSE teams. It reports, first of its kind, a detailed theoretical investigation into the potentials of leveraging the models of the activity theory for structuring GSE activities. It motivates and describes a data model and an architecture for building an infrastructure based on the activity theory able to facilitate applications integration; we assert that the data model and the architecture can not only guide our efforts for building an infrastructure for supporting the activity based paradigm in GSE, but can also provide interesting inspiration to other tool builders. It demonstrates the practicality of leveraging the concepts and models underpinning the data model and the high level architecture of our approach by reporting a supporting proof of concept prototype, which we have been incrementally designing and implementing. However, there are several interesting and important challenges that need to be explored and addressed to fully leverage the activity theory for building GSE tools. Hence, one of the goals of this paper is to garner the support from GSE researchers and practitioners for furthering this line of research; thus, for building solid theoretical foundations to support both GSE processes and tools based on concepts and models from the activity theory. As far as our research in this direction is concerned, we intend to carry out a few pieces of work. We intend to gain and share a deeper understanding for providing solid theoretical argumentation to thoroughly describe why the activity theory can be applied in the way shown in this paper. We intend to further refine the architectural design, and enhance the infrastructure to make several improvements before starting its trial in academic as well as industrial setting of GSE.

**REFERENCES**


