Social Team Awareness

Delfina Malandrino, Ilaria Manno, Alberto Negro, Andrea Petta, Vittorio Scarano, Luigi Serra
ISISLab, Dipartimento di Informatica, Università di Salerno
Via Giovanni Paolo II 132, I-84084, Fisciano (SA), Italy
Email: {delmal, manno, alberto, vitsca}@dia.unisa.it, {andrpet, luigser}@gmail.com

***Abstract*—Software that is meant to support collaboration is mostly developed "ad hoc", placing some additional overhead to users, that are required to integrate the common work practices, realized with the traditional software applications, with the new collaborative features offered by the new application. It has been argued that this is likely to inject lack of motivation on users, jeopardizing the positive effects of collaboration in workplace, since the time dedicated to collaboration is perceived as wasted.**

In this paper we present a generic mechanism to provide team awareness through the integration between a social platform and a work environment. The integration mechanism is, indeed, generic and the work environment potentially can be any kind of application usually adopted by team members. We illustrate the mechanism through the design and implementation of SocSVN, a proof-of-concept example in the scenario of collaboration support in software development. SocSVN integrates a social platform (Elgg, a well known open source social networking engine) with SVN, a source code versioning system widely used in software development.

We also abstract the mechanism provided and show how it is easily generalizable to other software, providing a list of the requirements and the amount of work to be integrated in the architecture.

***Index Terms*—team awareness; CSCW; software development; social platform;**

I. Introduction

So far, the Computer Supported Collaborative Work field has been widely explored and has produced many software systems oriented towards several aspects of collaboration: providing team awareness, supporting discussions, decision making, cooperative work, collaborative learning. However, the adoption of such systems has been hindered by several obstacles; one of the most relevant is the context switch between the phases of individual work and collaborative practices due to the lack of integration between the collaborative applications and the applications usually adopted in the work flow; another important limiting factor in the spreading of collaborative software is the lack of motivation by users: often, the users that more actively participates in collaborative practices are not those who benefit of relative results and, therefore, the time dedicated to collaboration is perceived as wasted [1], [2].

Nevertheless, the interest in computer supported collaboration has not decreased, as witnessed by a multitude of Web-based tools supporting collaboration and social interactions. Notable and not exhaustive examples are Google Apps, Smartsheet, SlideRocket, Wikipedia, Delicious, Facebook; a wide list of collaborative software is provided by Wikipedia[1]. Some of them, as the social networks, tagging systems, wikis, are well known and have a consolidated success in the consumer context. Their success has suggested the idea of using them to support collaboration in work setting [3]. Often, indeed, this approach has not been successful due to poor adoption by users [2], [4], [5]. In our opinion, these approaches have not solved the initial issues of CSS: '*disparity between those who do the work and those who get the benefit*' and the missing support for the context switch between personal and group work [1]. From several studies another interesting point has emerged: the success of Web 2.0 tools is based on the participation of a critical mass which is difficult to involve in a work context [4], [5], [6].

In spite of that, the idea of supporting collaboration through Web-based tools offers a wide range of possibilities to interconnect many different tools the users are familiar with and therefore this approach could be more easily accepted. Moreover it could reduce the friction between work and collaboration phases through the integration of the collaborative functions within the applications usually adopted by workers. This idea is not completely new and several attempts have been conducted with tools that are not web-based: Xia et al. introduced collaboration features into MS Words [7], Manno et al. introduced collaboration features into XMind (a tool to creates mind maps) [8] and several studies introduce collaborative features directly into a software development environment [9], [10], [11]. These studies present the same founding idea: the collaborative features are introduced *into* the work application and therefore, in some way, the scope of the collaborative features is application-specific and does not provide broad team awareness about the work goals, tasks and members activities beyond the current task. Our idea is to integrate work applications *within* Web-based collaboration features, so that users can collaborate through several work applications and achieve a richer team awareness.

Indeed, we do not forget that a web based tool is not *magic* and does not guarantee by itself a large adoption [3]: it does not solve the issues related to users' participation due to skepticism, work-collaboration friction, lack of motivation and of the critical mass to be reached (in case of Web-based tools). Therefore, the adoption of collaborative features should be stimulated and scaffolded. In literature, several studies propose rewarding mechanisms to stimulate users' participation [12],

[1] http://en.wikipedia.org/wiki/List_of_collaborative_software

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254087

[4], [13]. Even if these approaches produce a larger adoption, they often stress the competition among team members and should be fine-tuned to avoid negative phenomena like rat-races or bullying [4], [13]. Our idea to scaffold the adoption of such tools is to provide users with a system which can lead them beyond the initial slope by offering an automated mechanism able to create contents in a collaborative environment starting from certain working activities.

In this paper we present a generic mechanism to provide team awareness through the integration between a social platform and a work environment. The integration mechanism is, indeed, generic and the work environment potentially can be any kind of application usually adopted by team members.

The proof-of-concept of our mechanism is based in the scenario of collaboration support in software development. This context appears particularly relevant, since the developer teams often are world-wide distributed and need frequent interactions, both synchronous and asynchronous, during the design, development and testing of the software. Our proof-of-concept is SOCSVN, the integration of a social platform, Elgg [14], a well known open source social network engine, with SVN, a source code versioning system widely used in software development.

II. AWARENESS IN SOFTWARE DEVELOPMENT TEAM

Large software development can be a world-wide distributed activity and therefore supporting collaboration and team awareness is a widely recognized need: in spite of distances, team members need to share a clear idea of the project and its goals, the identification of problems, definition of tasks, respective activities and workplans, as well as interdependencies between all these elements [15]. A smooth coordination of the work leverages group awareness information, to provide every one with an understanding of the current state of the project and of required and expected next steps [16].

Over the years the interest in supporting team awareness in distributed software development context is increased, together with the available technologies, as witnessed by a wide and recent review of state of the art presented by Steinmaker et al. [17]. The authors have classified the papers on the bases of the 3C collaboration model [18], [19], [20] and of the Awareness Framework presented by Gutwin et al. [21].

The **3C collaboration model** defines three dimensions to support team collaboration:

- **C**ommunication: the exchange of messages among people to negotiate and make decisions;
- **C**oordination: the management of people, their activities and their resources, in a manner that prevents loss of communication and of cooperation efforts;
- **C**ooperation: the joint operation of members of the group in a shared workspace.

The **Awareness Framework** presented by Gutwin et al. [21] defines four dimensions for the team awareness:

- *Workspace Awareness*: knowledge about others' interaction in a workspace and its artifacts;
- *Group-Structural Awareness*: knowledge about such things as people's roles and responsibilities, their positions on an issue, their status, and group processes;
- *Informal Awareness*: in a work community, the general sense of who's around and what they are up to, the kinds of things that people know when they work together in the same office; informal awareness is the glue that facilitates casual interaction;
- *Social Awareness*: information that a person maintains about others in a social or conversational context, things like whether another person is paying attention, their emotional state, or their level of interest.

In their review, Steinmacher et al. [17] found that Coordination and Cooperation (in the 3C model) and Workspace Awareness (in the Gutwin's Awareness Framework) are the most supported aspects, while the aspects related to Communication (in the 3C model), Social and Informal Awareness (in the Gutwin's Awareness Framework) are less supported. Indeed, the results of the classification on both models suggest the same conclusion: so far most efforts have been focused on supporting coordination and work activities. Less attention has been dedicated to support, basically, the creation of relationships in distributed teams. However, this is considered crucial to create a shared context and a common knowledge among the team members, to avoid misunderstandings and increase opportunities for interactions, in other words to improve how the team members works together. Our paper follows and try to address this aspect, without modifying the current work practices, adding a social layer to team activities.

To enhance contextual collaboration, that is allowing people to use their core applications with collaborative capabilities embedded within these tools, Hupfer et al. proposed Jazz [9]. Their main goal was to embed collaborative components and capabilities into the Eclipse application development environment. Supporting ad-hoc collaboration among software developers has been explored also by Hedge and Dewan [22]: they integrated in Visual Studio a set of tools to support workspace awareness and team cooperation and coordination.

Among the works supporting team awareness, most are application specific and provide support to users when they are using a certain application in which social features have been integrated. Lightweight solutions for supporting collaborative software development include a work from Fitzpatrick *et al.* [23]. Their idea was to augment the Concurrent Version System (CVS) with a lightweight notification system and a tickertape tool on which developers can display CVS messages and chat with one other to enhance timely interactions. Their analysis showed that these communication facilities helped developers to: (1) increase the length of messages entered when making commit actions; (2) decrease the number of commit with empty messages.

Calefato et al. [11], which explore the usage of social network to support group awareness in distributed software development teams. They developed an extension of Jazz, to integrate social websites (leveraging FriendFeed, a social aggregator) to speed up the establishment of shared context and personal relationships. On one hand, their solution reduces

the friction between the social and work activities, on the other hand, it is application specific (only for Jazz) and could cause an overload of information in the CDE, as recognized even by authors which developed filters to face this issue.

The idea to integrate social features into the CDE has been explored also by Bani-Salameh et al. [24]. They developed a Social Collaborative IDE (SCI) which offers several collaborative features: presence, chat, notification about users activities on workspaces objects as well as a 3D virtual environment with users profiles showing working activities. Even in this case the proposed approach is application specific and could cause an information overload during working activities.

A. Our work

Based on our analysis of the current literature, supporting Communication and Social and Informal Awareness appears a key aspect to provide team awareness and group memberships. At the same time, our idea is that integrating the collaborative features into a specific application is in same way a constrain: users can access to such features only through that application, while the workflow often includes several phases and applications. Our idea is to provide team awareness by integrating social features *with* working applications instead of *into* them: this allow us to implement the integration with several applications (instead of being application specific) and at the same time we avoid information overload into the working application. In the consumer context, the social networks (Facebook, Twitter) offer a rich awareness about family and friends activities, plans, interests. However the usage of common public social networks is not recommendable in a work context: for security and privacy policies sharing information about the work in progress is not acceptable. Moreover, often these networks are filtered in the work places. Therefore, supporting team activities should leverages a private internal social network. Among several social network software, we chose Elgg[2] [14]: an open source project with a wide developer community. It offers almost 1800 extending Plugins, and was downloaded almost 3 millions times; it is used as private social network by (among the others) the NASA, the Australian and British Governments, the Stanford University and the Johns Hopkins University. As proof-of-concept we have implemented SocSVN the integration of Elgg with SVN [25], a typical tool used in software development activities to share and synchronize source code among distributed teams.

III. SocSVN: Systems and Features

SocSVN aims to provide team awareness about activities on the repositories by augmenting SVN with the social features provided by Elgg. Users work as usual with SVN and their actions trigger notifications and produce comments on their social network. We first describe SVN and Elgg, then we present the functionalities offered by their integration.

[2]A comparison among several social software can be found wikipedia: http://en.wikipedia.org/wiki/Comparison_of_social_networking_software

A. Systems

Apache Subversion [25] (i.e, SVN) is a software versioning and revision control system distributed under an open source license. It is used to maintain current and historical versions of files such as source code, Web pages, and documentation.

Additionally, we used WebSVN [26] an online subversion repository browser. Its main features include: showing subversion repositories, browsing the directory structure of the repository, viewing different versions of the repository, viewing the content of a file, comparing revisions of files. Finally, it provides RSS feed support for watching any resource.

Elgg [14] is an open source social networking engine that provides a robust framework on which to build all kinds of social environments. For each user, it offers a personal Wall page, with personal posts and related comments of other users. Moreover, it provides the possibility to manage bookmarks, blogging, sharing files, create and sharing pages. Each of these element can be commented. Furthermore, Elgg provides a wide set of Plugins, that allow to add extra functionalities. Among the available Plugins, we used the Calendar Plugin[3], which allows to add events on a calendar, and the Mentions Plugin[4] that allows to tag users in order to generate personal notifications (i.e., `@Alice` to allow an automatic notification to Alice). Exploiting this possibility offered by Elgg, we developed a specific Plugin to integrate in a social environment resources and notifications coming from SVN. In our examples, we used a Facebook-like skin to improve the degree of familiarity with the interface.

B. SocSVN Functionalities

SocSVN provides team awareness to SVN by leveraging the social features offered by Elgg. The integration between SVN and Elgg has a twofold view: when a user works on SVN, some of his/her actions are visible in Elgg as social events, but when logged into Elgg, users can visualize and comment both repository content (i.e. files and directories) and SVN actions of other users. Each user has an account both on SVN and Elgg, with a unique correspondence between these accounts.

Specifically, when a user commits a file on the repository, the following events happen on Elgg:

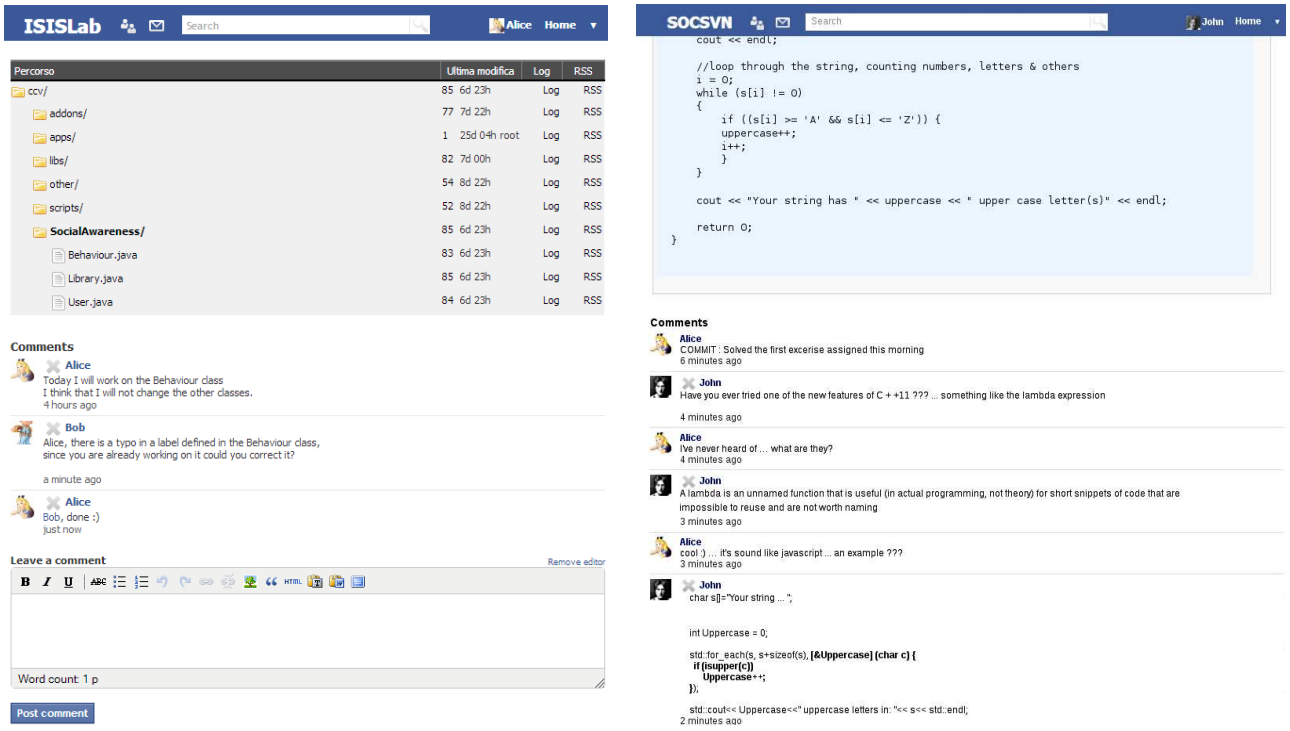
- a synchronous notification is shown to all Elgg users (which are his/her friends) logged in;
- an asynchronous notification is sent to all Elgg users (which are his/her friends) in their News Feed pages (something like the Facebook Timeline), shown in Fig. 1(a); if a user commits a file on SVN adding also a comment, the comment is also reported on the notifications, and if the user tags somebody in the comment (i.e. something like "`This should solve the problem @Alice had yesterday`"), the user is personally notified.
- a notification associated with the commit is added on a calendar, shown in Fig. 1(b);

[3]http://community.elgg.org/plugins/384926/0.84/event-calendar
[4]http://github.com/Elgg/mentions

(a) SVN commit notifications on Elgg

(b) SVN commit notification on the Calendar

Figure 1. The commit on the SVN produce automatically events on the calendar and post on Elgg which can be commented. The picture on the left shows the interactions between Alice and Bob about a commit made by Alice; the picture on the right shows the SVN activities on the calendar.



(a) SVN resources visualized as tree-structure on Elgg

(b) Comments on a SVN resource

Figure 2. The SVN resources can visualized and commented into Elgg. The figure on the left shows the tree-structure of resources on SVN and the comments associated to a specific resource; the figure on the right shows the source code of file on the repository; on this file Alice and John discuss about the implementation of a simple C++ exercise.

- a comment is added on the corresponding resource in the repository (i.e., its tree-like structure representation), shown in Fig. 2(b).

Additionally, we implemented a hashtag mechanism such that, when a comment contains for example #`developers`, the group in Elgg (if existing) is notified, persistently, on its News Feed Page. Each user subscribed to that group will be personally notified. We also added an automatic tagging mechanism that allows easy team partitioning in subprojects: each time a file is committed, the comment generated also contains the hashtag of the directory of root where the file is stored. If a group with such a name exists, it is notified.

When logged into Elgg, beyond the typical offered activities, a user can:

- visualize all the notifications triggered by the SVN actions;

- browse the repository, shown as a tree-like structure inside Elgg; each item of the tree can be commented (see Fig. 2(a));
- visualize the files and comments about them (see Fig. 2(b));
- visualize the events associated with commit actions on a calendar; each event shows the name of the committed resource and its SVN comment (see Fig. 1(b)).

C. SocSVN Use Cases

In this section, we outline some use cases of our proof-of-concept system. We will discuss two different scenarios: the first involving programmers working on the same project while the second involving students studying together on programming exercises.

Implicit work synchronization: Bob is working on the Java class `User.java` and knows that it is related to other classes (via the UML Class Diagram), such as `Behaviour.java` and `Library.java`. Suddenly, through the social stream, he sees that Alice has just committed `Behaviour.java`. He can quickly comment on the activity and discuss with Alice what changes have occurred and if they do impact or not on his current work. When potential conflicts are settled, Bob can continue his work and finish it, committing to the SVN repository the file `User.java` (see Fig. 1(a)).

Hours later, when Bob and Alice are offline (e.g., out for a meeting), John begins his work on `Library.java`. He was not online during the comments exchanges between Bob and Alice (as often is the case, in large development teams, cooperation may occur across different timezones), but he is aware of the dependencies and sees (from the Calendar Plugin, shown in Fig. 1(b)) that Bob and Alice committed when he was not connected. By accessing to the social network, and checking theirs News Feed Pages (i.e., timelines), he is quickly briefed and can proceed to commit or can further comments, if more synchronization of the modifications is needed.

Work collaboration: Alice and John, students of the first year of a Computer Science programming course are studying together on a simple C++ exercise. Alice develops a simple program to count the number of uppercase characters available in a string, and when finished, she commits the file on the SVN. Bob receives a synchronous notification and click on the name of the corresponding file starting, therefore, a discussion with Alice (see Fig. 2(b)). The topic was about the knowledge of a new C++ expression (i.e., Lambda Expressions, anonymous function that maintains state and can access the variables that are available to the enclosing scope). By also using the provided chat John is able to explain to Alice how it can be used.

IV. SocSVN Design

In this Section we describe in detail how we have integrated Elgg and SVN to implement the functionalities offered by SocSVN. The overall system (shown in Fig. 3) consists of three Components (additionally, there is a repository of xml files), whereas each of them is composed by several modules.

A. The Teamwork Environment Component

The Teamwork Environment Component includes the SVN Server module an the modules responsible of getting SVN resources, (i.e. the SVN Remote Data Interface and the SVN Web UI modules) and producing notifications (i.e the Hooks Manager). We used the standard SVN Server without making any change to its architecture. The Hooks Manager module uses the *post-commit* hook (a shell script) provided by SVN, to intercept the events. Next, a PHP script notifies the commits performed on the repositories to the Notification Server. The SVN Remote Data Interface and the SVN Web UI modules are part of the WebSVN software [26]. The SVN Remote Data Interface represents the access point to the resources on the SVN repository. It makes these resources available through a standard HTTP request. We used this mechanism to get a JSON representation of the data. From the SVN Web UI module we have extracted (i.e. copy-pasted) the part of HTML and CSS code to render the data in the interface of the users logged into Elgg.

B. The Bridge Component

The Bridge Component includes the modules responsible of transferring resources (the Work Environment Caller module) and notifications (the Notification Server module) toward the social environment.

The Notification Server, implemented in PHP, listens for notifications from the SVN Server and forwards them to all connected clients on the Elgg Component.

The Work Environment Caller module acts as an intermediary between the Teamwork Environment and Elgg. It receives the requests for resources from the Bridge Caller and gets the corresponding responses from the SVN Remote Data Interface. These responses are finally delivered to the Bridge Caller.

C. The Teamwork Description Repository

The Teamwork Description Repository contains the XML file(s) that describes the resources available in the Teamwork Environment(s). This file is used both by the Elgg Component (by the Bridge Caller, described later in this Section) to know the available resources and by the Bridge Component (by the Work Environment Caller) to know which communication mechanism is required to get each resource. Details about this file are shown in Section V.

D. The Elgg Component

Elgg, as described in Section III, is a powerful open source social networking engine. From the technical point of view, Elgg is written in PHP language and uses MySQL to manage the persistence. It also runs on the LAMP stack (Linux-Apache-MySQL-PHP). Specifically, we have developed an Elgg Plugin, that includes the module responsible of managing resources (the Bridge Caller module), notifications (the Notification Manager module) and of managing the user interface (the UI Management module).

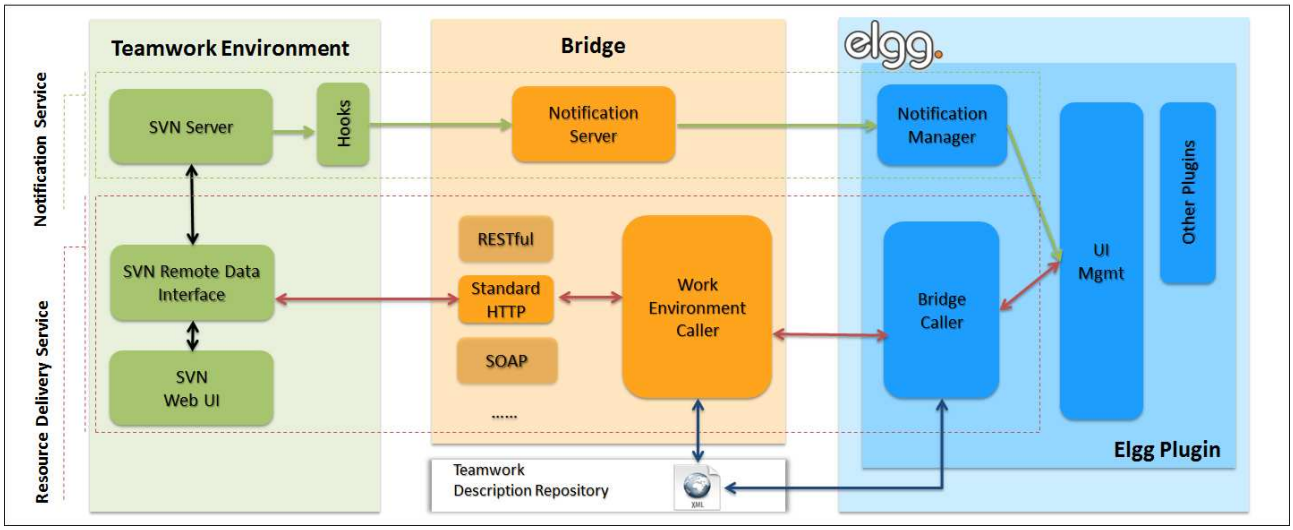


Figure 3. How modules of the Teamwork Environment and Elgg interact with each other.

The Notification Manager registers itself as client for the Notification Server. It receives the synchronous and asynchronous notifications of the SVN actions to create entities which can be managed by the UI Management module.

The Bridge Caller (implemented in PHP language) asks for specific SVN resources to the Work Environment Caller. This approach allows the Bridge Caller to ignore the specific mechanism required to get the resource from the Team Work Environment, so it can ask any resource (available in the XML file) without implementing the related communication mechanism. Once received the resources, the Bridge Caller makes them available to the UI Management Module.

The UI Management Module integrates into the Elgg user interface the Elgg entities corresponding to the requested resources and the produced notifications. This module can be written from scratch or can leverage an existing Web user interface of the Teamwork Environment resources. In our case, WebSVN provides part of the user interface for SocSVN.

The information flow exchanged among all these components and modules is described in the next Section.

V. SocSVN Information Flow

The communication between the Teamwork Environment and the Elgg components is realized through two different interaction mechanisms (shown in dotted boxes in Fig. 3): the *Resource Delivery Service* and the *Notification Service*. The Resource Delivery Service is responsible to get the content requested by users through Elgg from the Teamwork Environment. The Notification Service sends notification from the Teamwork Environment to Elgg, to synchronously notify some events (i.e. commits, lock of files, etc.). In the following we describe the implementation of such mechanisms.

A. Resource Delivery Service

The Resources Delivery Service manages the interactions needed to get the resources, requested by users through Elgg, from the Teamwork Environment. These interactions happen among the Bridge Caller Module (in the Elgg Plugin), the Work Environment Caller Module (in the Bridge Component) and the SVN Remote Data Interface Module (in the Teamwork Environment Component).

1) The Bridge Caller Module: it asks for the required resources to the Work Environment Caller. The Bridge Caller achieves the information to locate the desired resource by accessing to a Work Environment Resource Definition XML file (i.e., WERD) saved in the Teamwork Description repository (shown in Fig. 3). This file (a fragment is shown in Listing 1) contains two different sections:

- the $< bridge\_handler\_configuration >$ section, used exclusively by the Bridge Caller Module, that defines the information to create the communication channel with the Work Environment Caller (i.e., its endpoint reference);
- the $< resources >$ section that defines how to get resources on the Teamwork Environment; from this section the Bridge Caller uses the information defined by the $< service >$ section, which defines the operation to get the resources on the Teamwork Environment.

```
<?xml version="1.0" encoding="utf−8"?>
<werd>
 <bridge_handler_configuration>
  <bridge_handler_url_reference>
   http://172.16.15.43/SVNBridge/bridge_handler.php
  </bridge_handler_url_reference>
 </bridge_handler_configuration>
 <resources>
   <resource>
    <communication_channel>
     <communication_channel_class_reference>
```

```
      SVNCommunicationChannel
    </communication_channel_class_reference>
    <communication_channel_params>
     <location>http://172.16.15.88/websvn</location>
    </communication_channel_params>
   </communication_channel>
   <service>
     <name>getContent</name>
     <description>File content</description>
     <params>
       <param><name>content_location</name></param>
     </params>
   </service>
  </resource>
 </resources>
</werd>
```

Listing 1. The WERD file used by both the Bridge Caller Module and the Work Environment Caller Module.

When received the response from the Bridge Component (e.g., JSON, XML, binary format), the Bridge Caller processes it to build an object (regardless of the original response format) understandable by the UI Management Module, which makes the resource available into the Elgg user interface. Finally, this object will be used by a widget defined by the Elgg Plugin to enrich the content with social dimensions (i.e., discussion, sharing, collaboration and so on).

2) The Work Environment Caller Module: The Work Environment Caller Module acts as an intermediary between the Teamwork Environment and the Bridge Caller. It provides a specific module, named Bridge Handler, that is responsible of managing requests received by the Bridge Caller.

It must be emphasized that, for each resource available in the Teamwork Environment exists a corresponding service in the Bridge Component. To each service corresponds a specific communication channel (RESTFul, Standard HTTP, SOAP, and so on). Note that more services can use the same communication channel. Specifically, for each available resource, the Work Environment Caller uses from the WERD file the following information:

- a reference to the implementation of the communication channel ($channel\_communication\_class\_reference$), with corresponding parameters.
- information about the service: name, description and parameters (if any).

3) SVN Remote Data Interface: The SVN Remote Data Interface defines the access point to the contents of the Teamwork Environment. The access points can be implemented in any technology (RESTful, HTTP, SOAP, plain sockets, etc); obviously, the same technology should be used to implement the communication channel on the Bridge Component. The SVN Remote Data Interface is WebSVN, that is a client SVN which provides a Web user interface. It offers access to the resources through the HTTP protocol.

B. Notification Service

The Notification Service manages the notifications about specific SVN events (i.e. commit, lock/unlock of files). It involves the following modules: the Hooks Manager (in the Teamwork Environment), the Notification Server (in the Bridge Component) and the Notification Manager (in the Elgg Plugin).

When specific actions happen in the repository, the Hooks Manager executes a custom program that sends a notification to the Notification Server. The Notification Server receives the notification and forwards it to all the Notification Managers registered as listeners. The Notification Manager has to register itself as listener on the Notification Server, and therefore it will receives all the notifications about the events happening on the Teamwork Environment. Such notifications will be made available for the integration in the social user interface.

1) The Hooks Manager: The SVN server offers a hook mechanism to trigger notifications by executing a script when some specific events happen. The script can receives as parameters information such as version number, name of the changed resource, and so on. This script executes a PHP program which creates a communication channel (a client socket) with the Notification Server and sends information about the event occurred on the repository.

2) Notification Server: The Notification Server receives the updates about the SVN events and sends the corresponding notifications to all Notification Managers registered as listeners.

3) Notification Manager: The Notification Manager receives updates from the Notification Server about the events occurred on the Teamwork Environment (e.g. on the SVN server). It interacts with the UI Management Module to provide feedback to the users. In our proof-of-concept, the communication channel between the Notification Manager and the Notification Server is implemented through Web Sockets.

VI. The Generalization of the Mechanism

SocSVN is a proof-of-concept of a mechanism which is indeed, enough generic: the idea that working activities can trigger comments/events on a social network and that one can socially browse (comment, discuss etc.) shared resources is applicable to Teamwork Environments besides SVN. In the overall system, some modules are generic and do not require changes, while other ones are application-specific. The abstraction of the mechanism is depicted in Fig. 4, where the application-specific modules are shown with a dotted-line border.

This mechanism allows to enrich with Elgg-based team awareness any (Web-based) Teamwork Environment; moreover, it allows to integrate multiple Teamwork Environments with the same Elgg istance: for example to achieve notification on Elgg both about SVN and individual working activities on other applications.

Currently, we are working to apply the same mechanism to provide Elgg-based team awareness to Galaxy, a web-based platform for data intensive biomedical research [27].

In the following we are going to illustrate the modules which require some customization and the requirements of the Working Application.

A. WERD files

The WERD files are application-specific since they contains information to locate the resources available in the Teamwork

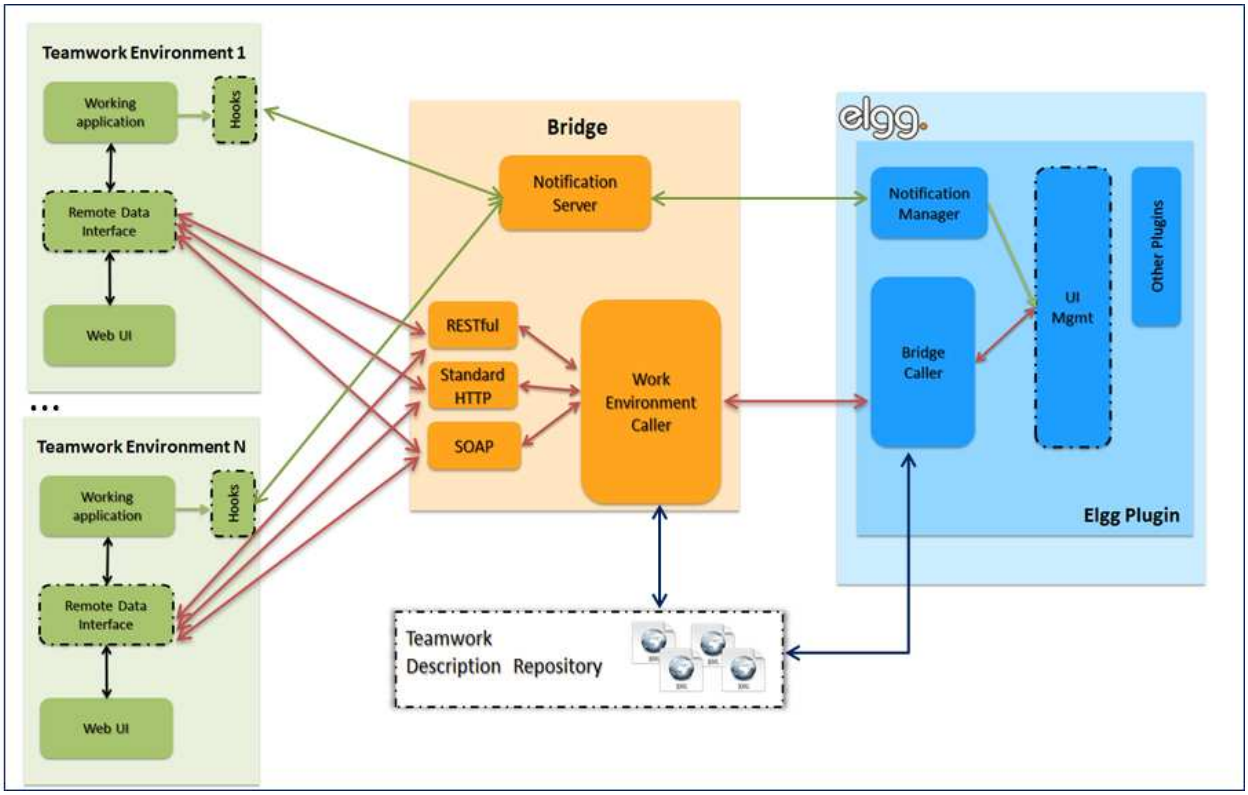


Figure 4. Generalization of the mechanism to enrich web-based Teamwork Environments (on the left side) with social-based team awareness (Elgg component, on the right side). Only the modules with dotted-line border are specific for the working application and require ad-hoc customization.

Environment. The complexity of such files depends on the complexity of the working application to integrate with the social platform.

B. The Hooks Manager

In general terms, the hooking mechanism is any technique to intercept events generated by an application during its normal usage. The purpose is to handle such intercepted events to augment the application behavior. In our case, the hooking mechanism is implemented in the Hooks Manager; it is application specific because it depends on whether the application offers an hooking mechanism or do not.

In SOCSVN we used native SVN hooks to trigger a PHP script able to communicate to the Notification Server information about SVN events. However, if the Team Working Environment does not provide an hook mechanism, it is possible to develop a proper system to intercept relevant events, depending on the nature of the application. For instance, you can use a polling system to retrieve information from the output of the application and then call a module (i.e., our PHP hook manager) to generate a notification. This kind of approach can be adopted in a number of scenarios. You can use the output of a *file system change monitor* to create a notification when a user works on some files or directories; you can run (at regular intervals) a command line output analyzer to retrieve information about system usage (CPU, disk, processes, etc.) and send a notification if certain events happen (i.e. if the disk usage percentage exceeds a threshold).

C. The Remote Data Interface

The Remote Data Interface module is application dependent since it allows to get the data from the Working Application, and, of course, each application has its own mechanisms. In SOCSVN we used a web based tool, WebSVN, to achieve data from SVN through Web requests (Standard HTTP). In the other prototype based on Galaxy (currently under development), the Working Application already offers the possibility to get data through Web requests (Standard HTTP and RESTful Services). In general terms, a Web-based Working Application is expected to offer methods to get data through Web (HTTP requests). Therefore, the work required to specialize the Remote Data Interface module for a certain Working Application often is limited to identify the methods provided by the application to get the data.

D. The Elgg Plugin

The Elgg Plugin is composed of several modules, whereas some of them do not require any modification for different Working Applications, while others require specific customization. In particular, the Bridge Caller and the Notification Manager can be used without changes for any Working Application (except that for optional customization of the notifications visualization). Some specific intervention is required to install the Plugin, to register the pages of the Plugin into Elgg.

The most relevant application-specific module is the UI Management module, since the integration of data in the user interface depends on data type. The development of this module is simplified if the Working Application is implemented by following the Model View Controller pattern and already offers a Web-based UI: in this case, the UI Management module can leverage on the View component of the Working Application. In SOCSVN the UI Management module leverages the UI of WebSVN for the visualization of the tree of resources in Elgg. Conversely, the integration of notifications in the News Feed page, on the Calendar, on the groups and the synchronous advises are developed ad-hoc. Similarly, in the other prototype based on Galaxy, some part of the UI Management module leverages the Web UI provided by the application itself, while other parts require specific development.

Obviously, the development of the UI Management module can be more complex if the Working Application does not provide a Web user interface or if the data to visualize are very complex. In SOCSVN the parts of the Elgg Plugin developed ad-hoc for SVN integration can be quantified in less than one hundred code lines.

E. Requirements on Teamwork Environment

We summarize here the requirements on the Teamwork Environment to apply the generic mechanism just described. The Teamwork Environment must adopt open standards for communication and offer publicly accessible API for data access. Moreover, to be easily integrated with Elgg, it should

- be Web-based: if not, some amount of work is required if one wants to show the content of the Teamwork Environment in the social network;
- be open-source: it should be possible to "take out" (part of) the WebUI of the Teamwork Environment and bring it in the Elgg plugin;
- follow MVC pattern: this facilitates the work of the designer, by easily identifying the components that are involved in visualization and data model;
- provide the Visualization part exclusively on the client side: some architectures have their Visualization layer that is partly based on server-side widget/components, which makes more complicate the integration in Elgg, that is facilitated, on the contrary, if a client-side Visualization layer is offered, e.g., with a combination of JavaScript and HTML5;

VII. CONCLUSIONS AND CURRENT AND FUTURE WORK

The goal of our paper was to present a generic mechanism to provide team awareness through the integration between a social platform and a Teamwork Environment. Our proof-of-concept prototype, SOCSVN, has been described in details and, then, we explained how the mechanism can be implemented with other Teamwork Environments. SOCSVN will be soon released as free and open source software. We will provide ready-to-deploy solutions, but we will also offer Elgg Plugin templates to integrate additional Teamwork Environments. SOCSVN provides heterogeneous social awareness: teams (i.e. Elgg groups) may be created, that are interested to different aspects (such as developers and system management) within the same social platform. It is dynamic and flexible since Elgg users may belong to different groups at once, and notifications are persistent within the group and can be easily accessed at any time.

Conversely to other works [7], [10], [9], [10] which integrate collaboration into working application, we offer team and working awareness into the collaborative environment. We aim to smooth the context switch between the individual work and the collaborative phases through the automatic integration into Elgg of news (post, calendar events, notifications) about team activities. This approach has two main advantages: on one hand, there is no work overload on the users (since the news on Elgg are generated automatically); on the other hand it is not intrusive in the individual work phase. Moreover, when users switch to collaborative phases they found on Elgg useful news: this is crucial to scaffold users' participation, since a system with few contents is not attractive, and the few contributors are further demotivated as they feel that nobody else is contributing [4].

Moreover, the usage of a social network enriches the awareness and collaborative dimensions which is less explored by current literature (see [17]): the Communication (in the 3C model), the Informal and Social Awareness (in the Gutwin's Awareness Framework). The Communication is supported asynchronously via posts and comments and synchronously through the chat provided by Elgg. These communication channels allow users to express their *opinion* in contextualized conversation (comments on SVN actions and SVN resources); these conversations, built around a topic or a post, implicitly provide awareness about the *interest level* of users involved. Chances of *informal communication* and sharing of *emotional feelings* are offered by typical social network interactions. Additionally, the system offers *presence indication* in the chat; providing presence indication in Elgg and not in the working environment implicitly provides also information about user's *availability* to collaborate. All these elements (opinions, interest levels, informal communication, emotional feelings, presence, availability, etc.) are the aspects which define the Informal and Social Awareness.

Several aspects of our research deserve further investigation. Of course, we have to conduct some experiment to evaluate effectiveness and acceptance of our system among users. Potential problems could arise for the information overload that could be generated by aggregating diverse and different Teamwork Environments into a single social interface that could generate information that could easily overcome the skills of the users, if not accurately trimmed down and carefully configured. It would be necessary to support the designer so that easy configuration of the amount of notifications to be delivered to the social interface is provided to designer/user.

Another aspect that requires our attention is the authentication service. When numerous components are to be integrated, the user account management could easily become complicated. Indeed, in the actual implementation each system has

its own user management mechanism and their integration has been one of the challenges faced during the development. Currently, each user has the same credentials on both the sytems. Of course we should make improvements to provide users with a unique access mechanism which automatically maps users identities between the integrated systems. Further future work will consider including secure authentication schemes, to ensure the scalability of the solution with respect to the number of Teamwork Environments and the number of users. This aspect will be complemented by the need of secure communication, actually non supported in our proof-of-concept prototype. For SOCSVN our idea is to adopt symmetric encryption algorithms, and specifically, the Kerberos protocol [28], since it provides a secure channel over an insecure network. We plan to integrate our system and Kerberos by adding the Key Distribution Center on the Bridge Component. The Work Environment Caller and the Remote Data Interface represent the Service Servers (SSs) components in our system enhanced with Kerberos. If the Remote Data Interface already provides a security mechanism for data access, then we can use it and avoid to consider this module as Service Server of the Kerberos protocol. The adoption of Kerberos allows to manage users so that each SVN and Elgg user is mapped into a unique Kerberos account. This support scalability respect to the number of Teamwork Environments and the number of users. Of course, the authentication and security layer based on Kerberos can be easily generalized as in Sec.VI.

The mechanism we described is generic enough and we illustrated via a working proof-of-concept prototype, but we are currently working on its application in another context, integrating social awareness for a well known bioinformatics collaborative environment, Galaxy [27]. Some other prototypes are also planned, such as integrating phpmyadmin as Teamwork Environment for facilitating the management of complex MySQL databases. Other examples that are planned, involve the monitoring of complex clusters and heterogeneous distributed computational environments as well as a set of personal applications.

REFERENCES

[1] J. Grudin, "Why CSCW applications fail: problems in the design and evaluation of organizational interfaces," in *Proc.of the 1988 Conf. on Computer-Supported Cooperative Work*. New York, NY, USA: ACM, 1988, pp. 85–93.

[2] L. J. Holtzblatt, L. E. Damianos, and D. Weiss, "Factors impeding wiki use in the enterprise: a case study," in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2010, pp. 4661–4676.

[3] M. Prilla and C. Ritterskamp, "The interplay of web 2.0 and collaboration support systems: Leveraging synergies," in *From CSCW to Web2.0: European Developments in Collaborative Design. Selected Papers from COOP08*, D. Randall and P. Salembier, Eds. Springer, 2010.

[4] S. Dencheva, C. Prause, and W. Prinz, "Dynamic Self-moderation in a Corporate Wiki to Improve Participation and Contribution Quality," in *Proc. of the 12th European Conf. on Computer Supported Cooperative Work, Aarhus Denmark*. Springer London, 2011, pp. 1–20.

[5] L. Nelson, G. Convertino, H. Chi, and R. Nairn, "Studying the Adoption of Mail2Tag: an Enterprise2.0 Tool for Sharing," in *Proc.of the 12th European Conf. on Computer Supported Cooperative Work, Aarhus Denmark*. Springer London, 2011, pp. 41–60.

[6] C. Dugan, W. Geyer, and D. R. Millen, "Lessons learned from blog muse: audience-based inspiration for bloggers," in *Proc.of the SIGCHI Conf. on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1965–1974.

[7] S. Xia, D. Sun, C. Sun, D. Chen, and H. Shen, "Leveraging single-user applications for multi-user collaboration: the coword approach," in *CSCW '04: Proc.of the 2004 ACM Conf.on Computer Supported Cooperative Work*. New York, NY, USA: ACM, 2004, pp. 162–171.

[8] I. Manno, F. Belgiorno, D. Malandrino, G. Palmieri, D. Pirozzi, and V. Scarano, "Introducing collaboration in single-user applications through the centralized control architecture," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, 2010, pp. 1–10.

[9] S. Hupfer, L.-T. Cheng, S. Ross, and J. Patterson, "Introducing Collaboration into an Application Development Environment," in *Proc.of the 2004 ACM Conf. on Computer Supported Cooperative Work*, ser. CSCW '04. New York, NY, USA: ACM, 2004, pp. 21–24.

[10] F. Belgiorno, I. Manno, G. Palmieri, and V. Scarano, "Argumentation tools in a collaborative development environment," in *Proc. of the 7th Int. Conf. on Cooperative Design, Visualization, and Engineering*, 2010, pp. 39–46.

[11] F. Calefato, D. Gendarmi, and F. Lanubile, "Embedding social networking information into Jazz to foster group awareness within distributed teams," in *SoSEA '09: Proc.of the 2nd Int. Workshop on Social software engineering and applications*. NY, USA: ACM, 2009, pp. 23–28.

[12] B. Hoisl, W. Aigner, and S. Miksch, "Social rewarding in wiki systems, motivating the community," in *Online Communities and Social Computing*, ser. LNCS, D. Schuler, Ed. Springer Berlin Heidelberg, 2007, vol. 4564, pp. 362–371.

[13] R. Farzan, J. M. DiMicco, D. R. Millen, C. Dugan, W. Geyer, and E. A. Brownholtz, "Results from deploying a participation incentive mechanism within the enterprise," in *Proc. of the SIGCHI Conf.on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 563–572.

[14] Elgg, "Open Source Social Networking Engine," http://www.elgg.org/.

[15] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38, no. 3, pp. 69–81, Mar. 1995.

[16] C. Gutwin, R. Penner, and K. Schneider, "Group awareness in distributed software development," in *Proc. of the 2004 Conf. on Computer Supported Cooperative Work*. NY, USA: ACM, 2004, pp. 72–81.

[17] I. Steinmacher, A. Chaves, and M. Gerosa, "Awareness support in distributed software development: A systematic review and mapping of the literature," *Computer Supported Cooperative Work (CSCW)*, vol. 22, no. 2-3, pp. 113–158, 2013.

[18] C. A. Ellis, S. J. Gibbs, and G. Rein, "Groupware: some issues and experiences," *Commun. ACM*, vol. 34, no. 1, pp. 39–58, Jan. 1991.

[19] H. Fuks, A. Raposo, M. A. Gerosa, M. Pimental, and C. J. P. Lucena, *Encyclopedia of E-Collaboration.* Hershey: IGI Global, 2008, ch. The 3C Collaboration Model, pp. 637–44.

[20] U. M. Borghoff and J. H. Schlichter, *Computer-supported cooperative work: introduction to distributed applications*. Springer, 2000.

[21] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation," in *People and Computers XI*, M. Sasse, R. Cunningham, and R. Winder, Eds. Springer London, 1996, pp. 281–298.

[22] R. Hegde and P. Dewan, "Connecting programming environments to support ad-hoc collaboration," in *Proc. of the 2008 23rd IEEE/ACM Int. Conf. on Automated Software Engineering*, ser. ASE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 178–187.

[23] G. Fitzpatrick, P. Marshall, and A. Phillips, "CVS integration with notification and chat: lightweight software team collaboration," in *Proc. of the 2006 Conf.on Computer supported cooperative work*, 2006, pp. 49–58.

[24] H. Bani-Salameh, C. Jeffery, and J. Al-Gharaibeh, "A social collaborative virtual environment for software development," in *Collaborative Technologies and Systems (CTS), 2010 Int. Symposium on*, 2010, pp. 46–55.

[25] "Apache Subversion," http://subversion.apache.org/.

[26] WebSVN, "Online subversion repository browser," http://www.websvn.info/.

[27] Galaxy Project. [Online]. Available: http://galaxyproject.org/

[28] Kerberos. [Online]. Available: http://www.kerberos.org/