The Grid Shared Desktop: a bootstrapping environment for collaboration

Pascal Dugénie, Philippe Lemoisson, Clement Jonquet, Monica Crubézy, Claude Lorenço

THE GRID SHARED DESKTOP: A BOOTSTRAPPING ENVIRONMENT FOR COLLABORATION

Pascal Dugénie*, Philippe Lemoisson*, Clement Jonquet*, Monica Crubézy**

Abstract

The paradigm shift from an information sharing infrastructure (i.e., the Web) to a resource sharing infrastructure (i.e., the Grid) has open new perspectives for CSCL (Computer Supported Collaborative Learning). With Grid, it is now possible to envisage a scalable infrastructure that offers live collaborative environments in a secure manner. The Grid Shared Desktop (GSD) is such a collaborative environment that inherits from the desktop as a natural human-machine interface to become a multidimensional humans to humans interface via several dedicated desktops.

However, the success of such environments depends upon several consideration that we will develop here. We have not so far identified any equivalent solution that can fully suit CSCL requirements. In fact, all solutions are either ad-hoc system-oriented or they are not scalable since they cannot manage resource efficiently. In order to satisfy the CSCL needs, we propose a platform independent solution that benefits of the intrinsic advantages of the Grid technology. This goal is greatly enhanced thanks to the capability of Grid, to support stateful, dynamic services.

In this paper, we tackle also the problem of bootstrapping and supporting a collaborative environment. As we target communities of non computer-literate people, we investigate easy-to-use and flexible solutions. Finally, we present our latest experimental case study with the GSD in the context of collaborative construction of a shared ontology.

Key Words

CSCL, Computer Supported Collaborative Learning, Grid, GSD, Grid Shared Desktop, collaborative ontology building

1. Introduction

Nowadays, everyone is familiar in using a desktop to interact with a computer. Graphical user interfaces are designed to allows users to manipulate graphical representations of concepts. Although the idea of desktop is commonly established, the idea of sharing a desktop is having much less interest for various reasons. Among these reasons, privacy and security are often seen as the major reason for people to be reluctant. Therefore, the main requirement for accepting to share a desktop with others is to share an environment that is not private. Furthermore, we believe that this environment should be ubiquitous and dedicated to the purpose of the community. We adopted the Grid as underlying technology to provide such a pervasive environment. Amongst other advantages, Grid offers a scalable, secure and reliable infrastructure that enables to achieve a GSD environment in which users may import new services and introduce new users dynamically according to the needs of the collaboration. Among the eLearning and semantic Grid activities, the GSD represents a typical example of what DeRoure et al. call a *Live Information System for Collaboration* [22].

We have proposed our GSD solution [26] to boost CSCL (Computer Supported Collaborative Learning) activities. In a paper we published 2004 [11], we started to outline our approach by defining the bootstrapping aspects of a collaboration. Then, we presented the original GSD framework last year [10]. Here, we present our latest experimental case study and results to determine how the GSD can facilitate the collaboration process in the context of developing of a shared ontology.

More specifically, within a collaboration session, collaboration is realised following two modes: (i) a *screen sharing mode*, where each participant may alternatively (according to a turn-taking mechanism) broadcast a part of his/her own desktop to all the other participants of the collaboration session; (ii) a *common environment m_ode* where each participant may alternatively (according to a turn-taking mechanism) act on a special desktop common to all participants of the collaboration session.

One of the challenges of the GSD is to provide a service-oriented architecture that enables collaborative work. We will see how Grid provide such a solution allowing the GSD to be itself a service and an environment for service exchange. The choice of Grid technology is motivated by recent research on these topics [5,21,24] that provides a significant contribution in understanding the actual Grid potentialities related to distant collaboration, and that promotes the development of service-oriented usage (and expansion) of

* LIRMM, CNRS & University Montpellier II, 161, rue Ada 34392 Montpellier Cedex 5, France; email: {dugenie, jonquet, lemoisson}@lirmm.fr

** Stanford Medical Informatics, Stanford University, Stanford, CA 94305-5479, USA, crubezy@stanford.edu

this technology in all contexts and especially where human collaboration is required (e-learning, e-government, e-health, e-business, e-science).

2. Requirements for a collaborative construction of an ontology

One frequent goal of human collaboration is to define a common corpus of knowledge about a domain of expertise; when formalised and computerised, this knowledge representation artefact is known as an *ontology*. A collaborative construction of an ontology consists of repeated short cycles of elicitation, specification, implementation, evaluation steps within a scenario. Collaborative ontology construction however presents a certain number of challenges (such as explanation of viewpoints, negotiation of terms and meaning, decision among modelling options) that lead to a number of requirements.

Informal learning. A notifiable effect during a collaborative ontology construction consists in enhancing learning as a result of the interaction. This kind of learning, known as informal learning, happens as the side effect of activities and observations that have not learning itself as their aim [8]. Laurillard [19] has outlined how knowledge acquisition can be linked to guided concrete action in the context of a conversational framework. Therefore collaboration can be seen as a normal opportunity for learning, as it interweaves individual experiences and exchange of concepts, in the context of an interactive use of language: conversation. Reversely, conversation establishes the necessary ingredient for cooperative activities including the conversation itself: trust. As a result, collaboration not only allows a community to perform a set of joint activities effectively, but also allows the members of that community to learn new knowledge and skills and to improve shared understanding overall. Therefore, a major goal of developing a general-purpose collaborative environment is to allow informal learning to occur.

Synchronous and asynchronous collaboration. Considering the timescale (duration) and the size (number of members, amount of resources) of a group, we distinguish between two modes of collaboration: synchronous and asynchronous modes. The synchronous mode is more appropriate to consider for short-lived collaboration sessions that involves a small number of participants, whereas the asynchronous mode fits better long-term collaboration activities that involves many members. These two modes of collaboration imply different requirements and characteristics for the collaborative environment. For example: (i) in a synchronous collaboration, users may inform others of their presence, they should follow a turn-taking mechanism, they may communicate by direct means supporting audio or video channels (e.g., chat, videoconferencing, shared desktop); (ii) in an asynchronous collaboration, users communicate with indirect means (e.g., email, shared files) that entail delayed responses, they should follow a different kind of turn-taking mechanism (e.g., file locking). A flexible collaborative environment should address the two modes of collaboration.

Trusted environment. Users need a trusted environment to collaborate freely with others. One problem in remote collaboration is the fact that people are not physically interacting. Collaboration occurs via an environment in which they exchange their knowledge. This environment requires security to ensure privacy and reliability to ensure anytime availability. Among other features, such a collaborative environment needs to implement a simple user-authentication mechanism and needs to manage private user accounts and displays as well as shared areas of collaboration.

Enhanced-presence. In a collaborative environment, participants need to be aware at all times of the presence and availability status of each of the other participants, and more generally of the life of the community. Indeed, participants need to be able to discover, locate and contact other participants easily, to have access to public availability information to schedule work sessions, to communicate directly or indirectly, and so on. In recent years, several network communication tools (such as videoconferencing or chat software) have been developed to include simple, yet powerful “enhanced-presence” features coupled to audio-video devices such as webcams and headsets. Those tools not only indicate the presence, location and availability of people, but also instill a feeling of community belonging and awareness that mimics the social dynamics of a local work team. For example, BuddySpace, [1,12] an enhanced-presence environment with instant messaging functionalities, allows for multiple views of collaborative workgroups. Presence awareness increases emotional well-being [25], and users benefit from knowing who else is around via presence and messaging tools. Another tool is for example Flashmeeting [2]. Incorporating enhanced-presence at the heart of the collaborative environment may improve the effectiveness of joint activities.

Persistent, searchable memory. The collaborative environment should not only enable the conduct of collaboration sessions, but also serve as a permanent, one-stop repository of the virtual community’s work. The environment hence needs to offer a structured space

for creating, storing and consulting shared documents and artefacts, as well as to maintain a searchable, traceable history of collaboration activities, with timestamps and provenance metadata.

Dynamicity. One can not know in advance precisely which tool, service or pedagogical technique will be employed within collaboration activities. Similarly, one cannot predict which members are part of the community (members may join or leave at any time). But one can provide the means to enable members to "do the right thing at the right time". A collaborative environment needs to bootstrap and support the collaboration by enabling people to dynamically: (i) import or remove services; (ii) discover and approach each other; (iii) notice who is available at a given time; (iv) schedule collaboration sessions; (v) communicate directly or indirectly; (vi) trace and analyse the history of community life. The notion of dynamically-generated, custom-tailored services thus becomes central to the design of a collaborative environment, that needs to be capable of instantiating and providing appropriate services to participants based on explicit requests, stored preferences or dynamically identified collaboration patterns.

Usability. Accessing the collaboration environment must be possible with a simple click and without installation of any third-party application. This can be achieved through a thin terminal and a stateless client as long as the state is managed elsewhere (i.e. provided by a stateful service).

3. Technological background

3.1. Potentialities offered by Grid

In order to achieve an ubiquitous collaborative environment with the requirements described in the previous section, Grid offers three major capabilities: scalability, security and reliability. In addition Grid has made a major step towards standardization that enable Grid technologies to evolved from ad-hoc solutions to the so-called Open Grid Services Architecture (OGSA) [4].

OGSA aims to support "flexible, secure, coordinated resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations" [14,16]. It was originally designed to be an environment with a large number of networked computer systems where computing and storage resources could be shared as needed and on demand; therefore OGSA provides the protocols, services and software development kits needed to implement flexible, controlled resource sharing on a large scale. Grid users are members of virtual organizations. A virtual organization is a dynamic collection of individuals, institutions and resources brought together by common goals of sharing resources and services. OGSA introduces two major characteristics in the so-called service-oriented architectures by distinguishing service factory from service instance. In other words, services are instantiated with their own dedicated resources and for a certain amount of time. These characteristics enable: (i) service state management: Grid services can be either *stateful* or *stateless*; (ii) service lifetime management: Grid services can be either *transient* or *persistent*[1].

Grid standardization has been extended with the Web Service Resource Framework (WSRF) [13] to adopts Web services (Foster [15]). WSRF defines uniform mechanisms for defining, inspecting, and managing stateful resources in Web/Grid services. Grid service concepts and standardisation are described in detail in [9]. Grid has now started to evolve toward the Semantic Grid [17,22] and the Learning Grid [5,21,24].

Figure 1 represents the key concepts of Grid in the OGSA terms. The bottom of the figure represents the Grid resources and the dual process of virtualisation/reification into service containers. Two different size containers are represented at the centre of the figure. Users may access to the services running in these container if they belong to the virtual organisation associated with this container. Adopting a Grid infrastructure as the backbone of a collaborative environment therefore enables to benefit from robust, ready-to-use infrastructural means that address many of the implementation requirements identified earlier.

[1] Whereas Web services have instances that are stateless and non-transient.

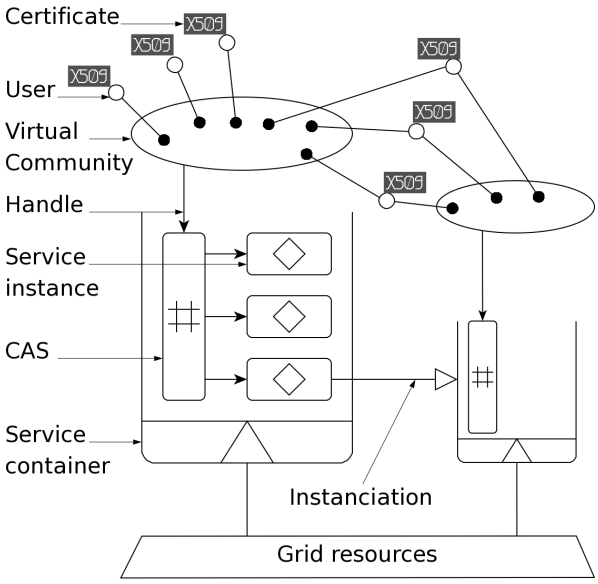


Figure 1.The Grid underlying infrastructure.

3.2. Current collaborative environments

The domain of distant collaboration is in real expansion both in research and business. An important number of tools are becoming available. We have evaluated some of them to check whether they meet the requirements that we have identified as crucial for collaboration. In particular, we evaluated Goto Meeting (www.gotomeeting.com), GatherPlace (www.gatherplace.net), Glance (www.glance.net), Beam4Free (http://beam4free.com), WorkSpace3D (www.tixeo.com), Groove Virtual Office (www.groove.net), AccessGrid (www.accessgrid.org). The idea of using desktops on the Grid was suggested in the Entropia project [7]. A study of Computer Supported Collaborative Learning (CSCL) applications in a Grid context was also addressed in [11]. An example is Gridcole, a Grid based system that enables easy integration of CSCL application [6]. These works adopt various approaches to offer three main types of functionalities:

- *Screen sharing* (or desktop broadcasting) that allows users to share their screen display on a network for other users to see exactly the same screen that they see.
- *Desktop sharing* that allows all the users to take a remote control of the devices (mouse, keyboard…) of the desktop owner. Screen/Desktop sharing functionalities are traditionally used in collaborative work sessions for slide shows or application demonstration.
- *Common environment* that provides all users with a virtual environment which can be a simple set of windows or a complex 3D space with avatars for users' representation. Users can interact and work together through the applications available on the common environment (chat, slide show, document edition, browsing, etc.).

However, it appeared through our evaluation that: (i) none of these tools satisfies the totality of identified requirements for collaboration. For example, the two first functionalities are only suitable for synchronous collaboration mode. The last one is suitable for both synchronous and asynchronous but lack the full dynamicity required (e.g., none of them support dynamic introduction of users); (ii) few of them fully benefit from the totality of the potentialities offered by Grid services.

4. The GSD architecture

The GSD architecture relies on the capability of Grid services to be dynamic and stateful. The GSD architecture adopts the Grid service container as a model of GSD instance. In a GSD instance, a number of services are persistent, in the sense that they are instantiated during the initialisation of the service container. For this reason, we call these services *bootstrapping services.*

Also we distinguish between two levels of GSD instances. The VC (Virtual Community) level and the CS (Collaboration Session) level. In this section, we describe the bootstrapping services and the subsequent differences between a VC and a CS service container.

4.1. The bootstrapping services

As a result of identifying the requirements for a collaborative environment and of assessing the Grid specifications, we have identified seven bootstapping services:

[**Authorisation**] to provide mechanisms to allow a user to access to the services of the community;

[**Notifications]** to provide immediate awareness of the life of the community;

[**Members Management**] to manage member such as add, delete or modify profiles;

[**Services Management**] to manage a set of services that are relevant for the community;

[**Services Activation**] to activate new instances of services;

[**Sessions Management**] to manage the different collaboration session within the community;

[**History**] to trace the history of collaboration within the community.

Authorisation. This service, called CAS (Community Autorisation Service) in the OGSA specifications, specifies the user rights levels towards any given services including the bootstrapping services. These rights level can be represented in a *MxS* matrix, where *M* are the members and *S* are the instances of services. For example only member m_x as a service manager may be allowed to import new services, while any member m_y would be allowed to set a new CS.

Notifications. This service is a broker responsible to dispatch messages between members to enable them to be immediately informed on everything concerning the VC life. These messages include the change of presence state of members, importation of a new service, invitation to participate to a CS, changes in relevant documentation, etc.

The Notifications Service is tightly connected to the History Service which traces the community history.

Members Management. This service is responsible for introducing or removing dynamically users in a group. The Members Management Service plays an important role in the GSD architecture 'group dynamics'.

Services Management. This service is responsible for importing new services in the service container of a given group. These services become accessible and can be activated. It is also responsible for removing services. The Services Management Service offers to users the ability to bring new tools as services for the group.

Services Activation. Any service available within a group service container may be activated by the Services Activation Service, generally requested by a user, for a given duration and for certain users. The corresponding technical term used in Grid specifications is "service instantiation:" a Grid service instance has its own state, lifetime management and allocated resources. The Services Activation Service allows using all the other services in the GSD. The Services Management Service and the Services Activation Service play important roles in the GSD architecture 'service dynamics'.

Sessions Management. Each member of a VC may decide to initiate a CS to address a specific collaboration need (more often synchronous). The Collaboration Sessions Management Service enables users to create, manage and delete CS. The life of this CS is then managed by six local services (cf. Figure 3) which are created as soon as the CS is created. The member initiating a CS, adds into the CS service container services he likes to benefit in the CS using the local Services Management Service. Then he invites participants to the CS with the local Members Management Service and manages rights between these participants and local services with the local CAS. He also activates the local services when needed with the local Services Activation Service. In the GSD architecture, the Collaboration Sessions Management Service is the only service capable to instantiate service containers.

History. An important aspect of collaboration is history. Indeed, group users actions and activities may be traced and logged to keep an history of the collaboration progress. The role of the History Service is to capture all significant events coming from the use of other services. It can for example log the access to a specific resource, register users profile evolution, inventory the past services activation, CS history, realise a shared document versioning etc.

4.2. Virtual Communities and Collaboration Sessions

We distinguish two kinds of groups in the GSD architecture:

- *Virtual Community* (VC). VC are quite stable and static groups. Their lifetime can be measured in months or years. A VC is composed of members and owns one service container.
- *Collaboration Session* (CS). CS are groups formed for a short period of time, measured in hours or days. A CS is created by a VC when members decide to collaborate in a synchronous mode. Goals of CS are similar to a meeting: planning work, finalizing a document, brainstorming, etc. May need different services, documents or communication tools that are not managed at the VC level. For this reason a CS own also a service container.

Figure 2 summarises the two kind of service containers with the bootstrapping services and the interaction between a VC and a CS. We may notice also that VC collaboration is mainly in asynchronous mode whereas CS collaboration is most of the time in synchronous mode. However, the real argument for distinction between VC and CS is related to the dynamics of the collaboration: a VC may engender a CS while a CS may not engender anything. From a Grid point-of-view, VC and CS are both seen as virtual organizations and therefore they are each associated to a Grid service container. The service container will be destroyed and the Grid resource will be freed as soon as the group lifetime is over.

A typical scenario is the creation of a VC with many members who interact asynchronously on various occasions (e.g., using collaboratively a shared file system with a given set of permission for each member). For each required synchronous interaction, a CS is set for a short time with a subset of the members of that VC (and possibly external invited members). The output of the collaboration corresponding to that CS (e.g., a synthesis of a discussion or decision report that is relevant for the VC) can be stored in a repository belonging to the VC.

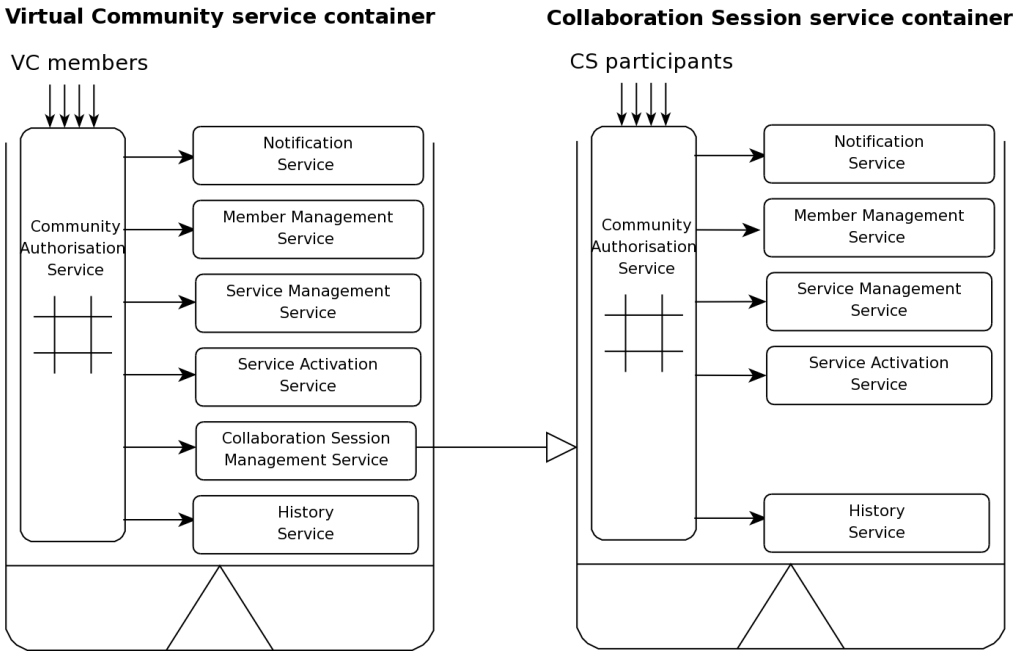


Figure 2.Bootstrapping service in the VC and CS service containers.

4.3 GSD processes

The bootstrapping services are involved in processes that correspond to the structure that defines dynamical relations between these services and interaction with them. We detail here some of this processes.

Service activation process. A member wanting to activate a service within a group requests the Services Activation Service with its user certificate and the service handle (and eventually some other information such as: involved members, lifetime expected etc.).

Then the Services Activation Service checks (by requesting the CAS) the member's right level for this service. Then, the Services Activation Service creates the service instance with a specific state, lifetime and allocated resources. It next requests the Notifications Service to notify group members that a service was activated. In parallel, it requests and activates the History Service respectively for both recording the service activation in the community history and for logging/tracing the user-service interactions from now to the end of the service life. The service is activated and group users may interact with it during the service life. At the end of the service life, the Services Activation Service both requests the History Service to stop recording and to store the service results (by versioning old ones etc.) and the Notifications Service to inform the community of the results of the end over service. Afterwards, it destroys the service instance and frees the allocated resources.

Service importation process. A member wanting to import a service in a group activates the Services Management Service as any other service (cf. service activation process) by giving it a service "external ID" and a table of rights levels for all users. The Services Management Service firstly includes in the service in the group service container and gives it an handle. Then, the Services Management Service requests the CAS to add a column in the members x services matrix. The Services Management Service has to specify the rights level of all members for the added service. Then, it requests the History Service to add an entry in its service database, and give it the trace/log mechanism for this new service. Afterwards, the Services Management Service asks the Notifications Service to notify the group members, that a new service was imported and is now available (following their rights level).

User introduction process. A member wanting to introduce a user in a group activates the Members Management Service as any other service (cf. service activation process) by giving it a user "external ID" and a table of rights levels for all services. The Members Management Service firstly stores the new user information (status, name, address etc.) and gives him a local ID. Then, the Members Management Service requests the CAS to add a row in the members x services matrix. The Members Management Service has to specify the rights level of all services for the added user. Then, it requests the History Service to add an entry in its user list. Afterwards, the Members Management Service asks the Notifications Service to notify other members, that a new member was introduced in the group.

Collaboration session management process. A member wanting to manage a CS within a VC activates the Collaboration Sessions Management Service by giving it three important things: (i) the list of services that he wants the CS to benefit from; (ii) the list of members he wants to participate in the CS; (iii) the information necessary to build the local CAS matrix of rights levels. The Collaboration Sessions Management Service firstly instantiates a new CS service container. Then it requests the local Services Management Service to import each service requested by the CS manager and requests the local Members Management Service to introduce each requested participant. These sub-processes are the same than the *service importation process* and the *member introduction process* specified before. Afterwards, the local CAS is requested to build its rights levels matrix. The Collaboration Sessions Management Service also requests (i) the Notifications Service to notify the entire VC of the creation of a new CS[2], (ii) the History Service to register the CS creation in VC history. The CS is now created and may be started. Each time the CS manager wants to activate a local service he requests the local Services Activation Service (cf. *service activation process*). The CS takes place with several services and their results. At the end of the CS, the Collaboration Sessions Management Service destroys the CS service container and frees the allocated resources. Local History Service data are transferred to the VC History Service. Finally, the Collaboration Sessions Management Service asks the Notifications Service to inform the VC of the set of results of the end over CS.

5. Collaborative construction of a shared ontology

To demonstrate and validate our approach and implementation of the GSD in a real-world setting, we conducted an experimental scenario involving the development of a shared ontology—computerised representation of knowledge—using a renowned ontology-building tool supported as a service on the GSD, as well as additional human-communication services.

5.1. The problem of collaborative ontology construction

A frequent goal of human collaboration is to define a shared, agreed-upon corpus of knowledge about a domain of expertise common to the members of the collaboration community. When formalised and computerised, shared knowledge can serve as the basis for better understanding among members of the community and better communication with external people, as well as for further development of common resources and for many problem-solving activities. In recent years, the representation of such shared

[2] Notice that CS participants have received two notifications: the one done by the local Members Management Services, and this one.

knowledge has largely been implemented by *ontologies*—formal, computerised conceptualisation of the notions, properties and relationships in a domain [18].

Building an ontology requires the use of a tool that provides means for creating and organising concepts, properties and relationships that are important in a given domain of expertise. Adopting such a tool, however, requires to understand the principles of ontology construction as well as mastering the set of associated user-interface tasks. Building an ontology collaboratively requires an additional level of service that includes multi-user serving of the ontology contents and user-interface views, user authentication and rights management, provision of real-time information on collaborators that are editing the ontology, portions of the ontology that are being modified, history of modifications, as well as provision of locking and commitment mechanisms. The Protégé knowledge-modeling environment (http://protege.stanford.edu) is a *de facto* standard tool that supports single and multi-user construction of ontologies.

Collaborative ontology construction presents a certain number of challenges, that exercise well the different kinds of interactions that can occur among the participants of a collaboration session. Types of interactions include the presentation/teaching of portions of the ontology, the discussion and reconciliation of multiple viewpoints on knowledge, the negotiation of vocabulary terms, their meanings and their relationships, and the confrontation of various knowledge-modelling options. Such interactions typically are not supported by current ontology-building tools; instead they need to be supported at a human-communication level by the collaborative environment with help of additional services. Specifically, by interacting through a multi-user ontology-development service, augmented by presentation services such as slide-presentation software, drawing and annotation boards, and enhanced-presence services such as chat and videoconferencing, in addition to using the GSD CS modes (i.e., screen sharing mode and common environment mode), members of the community are able to teach, learn, share and model knowledge of mutual interest.

5.2. General scenario presentation and experimental setup

Our objective in this scenario is to show how the collaborative environment created by a virtual community and supported by the GSD service can foster the collaborative modelling of shared knowledge in the form of an ontology.

A two steps experiment scenario

We argue that our scenario is typical of the process by which a set of collaborators initiate and execute a joint piece of work. At the start of a collaboration, participants present, discuss and choose their goals and the tools and methods that they are going to employ to achieve their goals. Methods and tools might not be known by everyone in the community; knowledgeable participants hence need to walk other participants through the principles of the tools and methods and their basic operation. This can be seen as a *learning phase*. Once most participants feel confident with the tools and methods, a *productive phase* can start in which the actual collaborative work takes place and more learning can occur along the way. Note that this is a typical context of informal learning. Collaborative ontology construction being no different from other collaborative activity, we divided our experiment scenario into two steps:

- A first, learning experiment in which the GSD service supports participants in teaching and learning principles of ontology development and the use of the Protégé ontology editor;
- A second, production experiment in which the GSD service supports participants in creating, explaining and maintaining an ontology collaboratively.

Consequently, our scenario enables us to show how the GSD service both (1) supports a typical e-learning setting with added features of screen sharing and common environment collaboration modes (in the first step experiment) and (2) provides an engaging framework for helping participants to reconcile, formalise and capture knowledge that is initially informal and distributed into a shared ontology that then can be used inside or outside of the GSD context (in the second step experiment). Notice that the second step experiment would not be possible without considering the results of the first one.

Experimental setup of the GSD

For the purpose of our experiments, we created a small VC, called “okprotege,” denoting the fact that members are willing to collaborate on a given task. First, we conducted four remote sessions focused on testing technical aspects of the GSD, and configuring the ontology-building and enhanced-presence services supported. Then, we conducted three short-time (2 hours) CSs involving five participants each accessing the GSD service by logging onto their ***Private Ubiquitous Desktop*** (PUD) in a Web browser. The chosen collaboration mode for these CSs was screen sharing mode. Each PUD was equipped with necessary collaborative and preference-

setting services as well as a one-click toggle granting a read-only access to the ***Broadcasted Virtual Desktop*** (BVD) and full access to the member's own PreBVD. At any time, participants could switch from operating the services on their PUD to visualizing both the PreBVD and the BVD.

During the experiments, the Protégé service supported by the GSD was available to each participant via their PUD as a client to a central Protégé server. Participants accessed and controlled shared ontology files on the server, following a classical client/server approach with authentication mechanisms allowing for parallel editing of the shared ontology. At the same time, participants communicated in private or joint conversations by audio, video or text chat using the FlashMeeting [2] videoconferencing service and the BuddySpace [1] chat service, both running on their private desktop[3]. In addition, collaborative tools such as text and document editors, a file system explorer and a Web browser were all available as services to CS participants within the GSD.

5.3. Prelimilary results

In our experiment, our main interest was to assess the use of the different collaboration modes supported by the GSD. The main collaboration mode that we adopted for this scenario was screen sharing. Motivations behind this choice were: (i) at any time during a CS, participants could make modifications to the shared ontology in parallel, hence fostering participants' initiative; (ii) participants could work on their own PreBVD when they needed to do isolated work or try out ideas, either on the common ontology or on a local copy of the files; (iii) according to a turn-taking mechanism, participants could each broadcast their actions, ideas or work to the community once they were ready to do so. This collaboration mode was appropriate both for a learning and a production phase.

Very interestingly, our experiments demonstrated the gradual transition from a learning phase of collaboration to a more productive phase, by which the role of participants evolved. The first two experiments were very much organised as an "instructor" remotely walking a small group of "students" through the concepts and user-interface metaphors underlying ontologies and ontology editing with the Protégé tool (see Figure 3). In these sessions, the instructor was the main participant broadcasting her desktop, demonstrating how to use Protégé and explaining ontology-building principles along the way. The instructor also guided other participants into trying simple exercises in front of others, by taking their turn on the BVD.

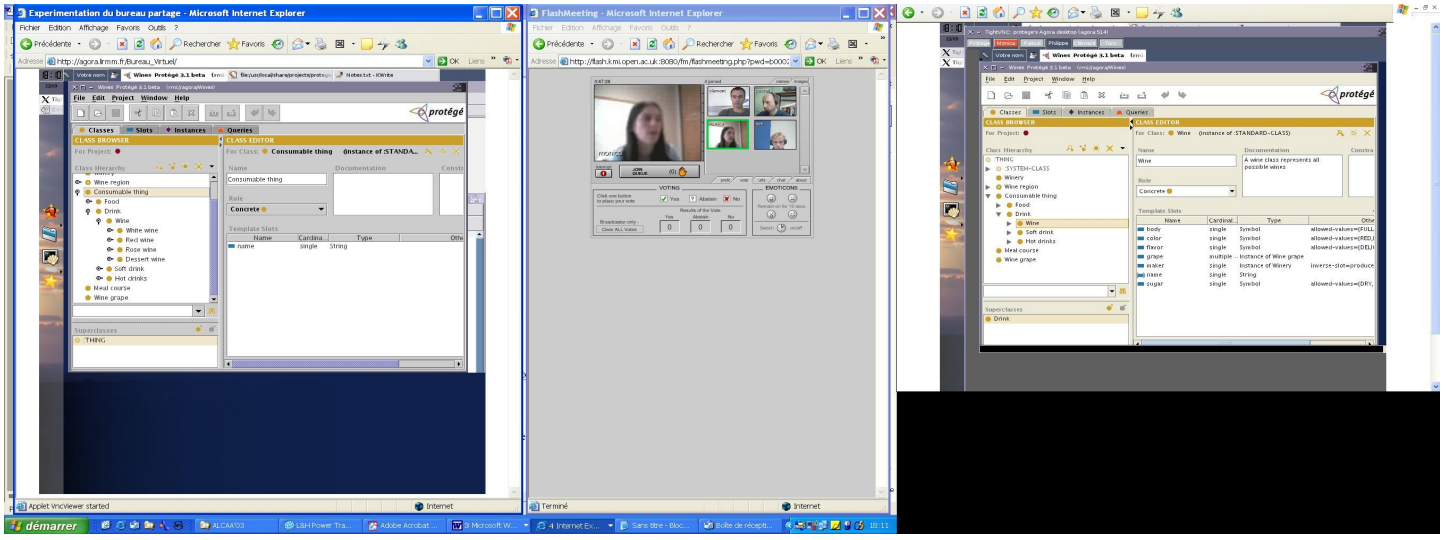


Figure 3. Screenshot of the Grid Shared Desktop in action, during our first collaboration session experiment.

The third experiment revealed an evolution of participants' roles and actions by which more initiative was taken by initial "students" and less direction was instilled by the initial "instructor." Although these initial experiments did not lead us to a full production phase in a newly-created ontology of particular interest to the okprotege community, they still demonstrated production-level creation of new concepts and properties in the tutorial ontology.

[3] For the sake of simplifying these first experiments, enhanced-presence tools were used externally to the GSD, on each participant's host desktop.

Figure 3 shows a screenshot of the GSD during one experiment. Left, the Web-accessible PUD of a participant is shown (here a Linux KDE desktop), focused on that participant's Pre-BVD, displaying his Protégé client serving a tutorial ontology about Wines. Center, a FlashMeeting videoconferencing session is running in parallel, showing the instructor explaining what she does with Protégé and other participants listening and following instructions. Right, another part of the participant's PUD is shown, focused on the BVD, broadcasting the PreBVD of the currently active participant (as denoted by the orange tab) displaying her Protégé client .

Conducting these initial experiments was crucial in devising the right mode of operation of the GSD in a scenario such as ontology building with Protégé. For instance, our experiments enabled us to improve the configuration of private and broadcasted desktops, to decide on a set of available services, to experiment turn-taking policies and moderator roles, as well as to study voice/video/screen interactions from the point of view both of technological interference and of each participant's experience in managing these collaboration modes simultaneously. In addition, we were able to solve a number of infrastructural details, such as providing a one-click Web-accessible interface for the GSD, managing user accounts, groups and rights, implementing efficient screen-broadcasting mechanisms and identifying multi-user requirements for applications such as Protégé.

Studying the dynamics of turn-taking on the BVD, we experimented with tasking the instructor with the role of moderating the turn-taking queue (as would a teacher in a classroom), as well as tasking any other participant with this role, mimicking a more spontaneous setting. In our current version of the GSD, we decided that spontaneity is most appropriate, and the GSD allows anyone to switch the BVD to any other participant. We still want to set-up safeguard and privacy mechanisms by which participants could prevent their PreBVD from being broadcasted at certain times, as well as a turn-asking queuing mechanism that would organise the dialogue among participants.

6. Conclusion and perspectives

In this paper, we have presented a novel approach to the design and implementation of a collaborative environment for virtual communities of practice. Our solution, the Grid Shared Desktop, or GSD, enables the bootstrap and support of remote collaboration among humans, by drawing from the powerful Grid infrastructure to provide a set of virtual desktops (private, broadcasted and common) to a community of users logging in and interacting through their Web browser. The main new aspect of our solution is a Grid service oriented approach. Via the GSD, users access a context dedicated to the collaboration within the different communities that they form. Virtual desktops play the role of service containers available for both of the two identified collaboration levels: virtual community and collaboration session. Each community member has his or her own Private Virtual Desktop, and both screen sharing mode and common environment mode of collaboration are possible thanks to a (Pre)Broadcasted Virtual Desktop and a Common Virtual Desktop. In this environment, an array of services are available to community members and provide them with means of communicating and working together in collaborative activities, as well as working in isolation on common documents and resources. We believe that supporting mechanisms such as authentication, turn-taking and enhanced-presence services allow users to feel at home in a trusted environment.

We conducted initial experiments using the GSD among ourselves in the context of a common activity in collaboration: constructing a shared understanding of knowledge as an ontology, using a set of renowned, well-accepted tools. The valuable results from the initial experiments provide an initial validation of our GSD approach. These encouraging results are now leading to a large scale experiment within a community of Chemists and Computer scientists. This community has started constructing collaboratively an ontology of Organic Chemistry, using the full potential of the GSD. This effort is part of the ambitious EnCOrE project (Encyclopédie de Chimie Organique Electronique). A preliminary assessment shows that this collaborative work is following the same two-step process that we experienced in our test experiments: from initially instructor-driven, the collaboration is now becoming more spontaneous, allowing for individual initiative as well as for small subgroup formation.

Acknowledgements

Work partially supported by the European Community under the Information Society Technologies (IST) programme of the 6th Framework Programme for RTD - project ELeGI, contract IST-002205 [3]. This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein.

Work performed in 2005 was also supported as a France-Stanford Development Project "Technology-Enhanced Learning on the GRID." by the France-Stanford Center for Interdisciplinary Studies.

References

[1] BuddySpace - Instant Messaging + Maps + Semantics = Enhanced Presence Management for Collaboration, Learning, and Gaming. www.buddyspace.org.

[2] FlashMeeting - The One Click Videoconference. www.flashmeeting.com.

[3] The European Learning Grid Infrastructure. www.elegi.org.

[4] The Open Grid Services Architecture. www.globus.org/ogsa.

[5] C. Allison, S. A. Cerri, P. Ritrovato, A. Gaeta, and M. Gaeta. Services, Semantics and Standards: Elements of a Learning Grid Infrastructure. *Applied Artificial Intelligence Journal, Special issue on Learning Grid Services*, 19(9-10):861–879, November 2005.

[6] M. L. Bote-Lorenzo, D. Hern´andez-Leo, Y. A. Dimitriadis, J. I. Asensio-P´erez, E. G´omez-S´anchez, G. Vega-Gorgojo, and L. M. Vaquero-Gonz´alez. Towards Reusability and Tailorability in Collaborative Learning Systems using IMS-LD and Grid Services. *Advanced Technology for Learning*, 1(3):129–138, September 2004.

[7] B. Calder, A. A. Chien, J. Wang, and D. Yang. The Entropia Virtual Machine for Desktop Grid. In *International Conference on Virtual Execution Environment, VEE'05*, Chicago, IL, USA, June 2005.

[8] S. A. Cerri. Models and Systems for Collaborative Dialogues in Distance Learning. In M. F. Verdejo and S. A. Cerri, editors, *Collaborative Dialogue Technologies in Distance Learning*, volume 133 of *ASI Series F: Computers and Systems Sciences*, pages 119–125. Springer-Verlag, Berlin, Germany, 1994.

[9] C. Comito, D. Talia, and P. Trunfio. Grid Services: Principles, Implementations and Use. *International Journal of Web and Grid Services*, 1(1):48–68, 2005.

[10] P. Dugenie. Orientation et usage de l'Architecture de Services Grille OGSA. In *MajecSTIC 2005*, pages 283–290, Rennes, France, November 2005. IRISA - IETR - LTSI.

[11] P. Dugenie and P. Lemoisson. A bootstrapping scenario for elicitating CSCL services within a GRID virtual community. In *1st European Learning Grid Infrastructure Conference, ELeGI'05*, Naples, Italy, March 2005.

[12] M. Eisenstadt, J. Komzak, and M. Dzbor. Instant messaging + maps = powerful collaboration tools for distance learning. In *Simposio internacional de Tele-Educaci´on y Formaci´on Continua, TelEduc'03*, Havana, Cuba, May 2003.

[13] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, I. Sedukhin, D. Snelling, T. Storey,W. Vambenepe, and S.Weerawarana. Modeling Stateful Resources withWeb Services. Whitepaper Ver. 1.1, Web Services Resource Framework, May 2004.

[14] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, USA, 1999.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*. The Globus Alliance, June 2002.

[16] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Supercomputer Applications*, 15(3), 2001.

[17] M. Geldof. The Semantic Grid: will Semantic Web and Grid go hand in hand? Technical report, European Commission DG Information Society Unit 'Grid technologies', June 2004.

[18] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[19] D. Laurillard. A conversational framework for individual learning applied to the 'learning organisation' and the 'learning society'. *Systems Research and Behavioral Science*, 16(2):113–122, March 1999.

[20] T. Richardson. Remote Frame Buffer Protocol. Technical Report Version 3.8, RealVNC Ltd., July 2005. www.realvnc.com/docs/rfbproto.pdf.

[21] P. Ritrovato, C. Allison, S. Cerri, T. Dimitrakos, M. Gaeta, and S. Salerno, editors. *Towards the Learning GRID: advances in Human Learning Services*, volume 127 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, November 2005.

[22] D. D. Roure, N. Jennings, and N. Shadbolt. Research Agenda for the Semantic Grid: A Future e-Science Infrastructure. Technical report, University of Southampton, UK, June 2001. Report commissioned for EPSRC/DTI Core e-Science Programme.

[23] M. P. Singh and M. N. Huhns. *Service-Oriented Computing Semantics, Processes, Agents*. John Wiley & Sons, Ltd, 2005.

[24] S. Wesner and K. Wulf. How GRID could improve E-Learning in the environmental science domain. In *1st LeGE-WG International Workshop on Educational Models for Grid Based Services*, Lausanne, Switzerland, September 2002. Electronic Workshops in Computing (eWiC).

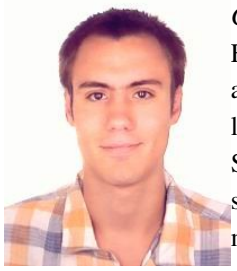
[25] D. Whitelock, D. Romano, A. Jelfs, and P. Brna. Perfect Presence: What does this mean for the design of virtual learning environments? *Education and Information Technologies*, 5(4):277–289, December 2000.

[26] L. M. Vaquero-Gonzalez, D. Hernandez-Leo, F. Simmross-Wattenberg, M. L. Bote-Lorenzo, J. I. Asensio-Perez, Y. A. Dimitriadis, E. Gomez-Sanchez and G. Vega-Gorgojo. *The Opportunity of Grid Services for CSCL-Application Development.* In 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'05): 4-11, 2005.

Biographies

Pascal Dugénie received his MSc. degree in communication systems from the University of Bristol, U.K., in 1993. He has been involved in hardware and software design of telecommunications equipment. He began in radio systems design and participated in network planning for the French broadcasting operator TDF. He has been involved in research in networks, protocols and systems for six years at the Centre for Communication Research, Bristol, U.K. where his interests concerned analysis of telecommunication traffic and performance of fixed and mobile networks. Later he joined Motorola as a system engineer where he improved a heuristic for the optimisation of frequency plans for GSM infrastructure. He chaired an ACTS Group during 1997-98 and participated to ETSI standardisation. Since, June 2004, he is research engineer at the LIRMM and prepares a PhD at University of Montpellier in the area of Grid computing.

Philippe Lemoisson is a graduate of “Ecole Polytechnique” (1977) and of ”Ecole Nationale des Ponts et Chaussées” (1982). Throughout his whole career as an engineer since 1980, he has been working on information management solutions and tools for a better collaboration inside companies: (i) as a senior consultant in the domain of Information System Management; (ii) as a research engineer in the area of telecommunications. Since 2000, he has been sharing his time between the management of Industrial or European projects and Scientific Research in the domain of Computer Science. He is currently working on a conceptual framework for “human cooperation enabled by formal systems”, including a protocol for a “conversational calculus” in the context of a PhD to be achieved in 2006.



Clement Jonquet, obtained a BSc. and a MSc. in Computer Science (CS) from the University of Montpellier, France. He is currently a PhD student in CS at the same university, in the Laboratory of Informatics, Robotics, and Microelectronics of Montpellier (LIRMM – www.lirmm.fr). At the same time, he is training as a junior lecturer and teaches CS to BSc students at the University of Montpellier. At LIRMM, he is a member of the Social Informatics/Kayou team, concerned with topics such as agents and multi-agent systems, distributed systems, Web and Grid infrastructures, intelligent services, ontologies, collaborative learning, e-learning. He is a member of the ELeGI project. Clement Jonquet’s home page is www.lirmm.fr/~jonquet.

Monica Crubézy is a research scientist in the Stanford Medical Informatics Laboratory at Stanford University. Her research focuses on modeling libraries of problem-solving methods and integrating them with domain ontologies to achieve knowledge-intensive tasks. She also studies mapping and mediating knowledge among ontology-based system components, such as in the context of the Semantic Web. Recently, she has been concentrating on the study of ontology building as a central activity in human collaboration. She received her PhD in computer science from the Université de Nice-Sophia Antipolis and the Institut National de Recherche en Informatique et Automatique. http://smi-web.stanford.edu/people/crubezy.