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 boosted the development of a new generation of online services. In particular, Grid services are stateful, dynamic and operate in a secure environment. Therefore they offer capabilities that are essential to remote collaboration. In this paper, we tackle the problem of bootstrapping and supporting a collaborative environment over a Grid infrastructure. As we target communities of non computer-literate people, we investigate easy-to-use and flexible solutions. As a result, we developed the Grid Shared Desktop (GSD) - a Grid-based, Web-accessible environment that provides members of a virtual community with a set of desktops supporting collaboration in both synchronous and asynchronous mode. A desktop is a familiar tool allowing users to interact with graphical representations of concepts. Thus, the GSD is a powerful interface to communicate these representations and build collaborative knowledge. In this paper, we propose an overall architecture and an implementation of the GSD including a bundled set of bootstrapping services required to set up and maintain a collaboration activity among distant participants. Finally, we summarise the results of recent experiments conducted with the GSD deployed in the concrete context of collaborative construction of a shared repository of knowledge.

Key Words: Collaboration, Collaborative learning environment, Computer Supported Collaborative Learning, Grid, Grid service, shared desktop, virtual community, ontology, collaborative ontology construction.

1. Introduction

The recent explosion of communication means and tools such as email, Web forums, instant messaging, short messages, videoconferencing and so on, shows the real growth of computer-assisted human interactions. The use of these tools has evolved, and nowadays, people interact all around the world not just to exchange information but also to exchange services. In fact, interactions and communication allow people to collaborate and realise Aristotle's adage "the whole is greater than the sum of its parts:" Resolving interactively common distributed problems is a real need for international companies, governments, academies and world wide consortia (e.g., for distant tutoring, distributed problem solving, instant cooperation, business process management, etc.). There is hence now a genuine need for collaboration methods and tools to address collaboration problems. In [11] we previously presented an analysis for bootstrapping a collaboration which led us to developping a new kind of tools for collaborative environment. The work that we present in this paper offers a solution to the core problems of remote collaboration among humans. Our solution, called the Grid Shared Desktop, or GSD provides the members of a virtual community with a shared, online collaborative environment including protocols, services and facilities for bootstrapping and supporting their collaborative construction of shared knowledge. The GSD framework was originally presented in [10].

The Grid Shared Desktop approach. Our approach to human collaboration emerged from the concrete collaboration needs that we experienced within the European research project ELeGI [3]. This project is concerned with the use of Grid technology [14,16] for enhancing collaboration and learning in distant communities. Analysing and experimenting collaboration among the participants of that project led us to a set of requirements that a collaborative environment should fulfil. We briefly present the results of our analysis in Section 2.1. This analysis further led us to the conceptualization and implementation of an ubiquitous, dynamic and shared[1] environment for collaboration.

We adopted a service-oriented approach over a Grid infrastructure, in order to address generic needs and requirements of collaborative work and collaborative environments. We propose an architecture for the GSD as well as an implementation of this architecture by means of virtual desktops.

The GSD represents a typical example of what De Roure et al. call a *Live Information System for Collaboration* [22]. With the GSD, members of communities can perform various tasks, with other members, without being forced to specify these tasks before they occur. The Grid infrastructure offers a secure and reliable environment in which users may import new services and introduce new users dynamically according

* 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

(paper no.??-??)

[1] The term "shared" entails a one-for-many architecture.

to the needs of the collaboration. Collaboration is accomplished in two ways:

- Within a Virtual Community (VC) alone, by using services that are mostly asynchronous, such as for example file sharing, notifications mechanisms, asynchronous edition of documents, etc.;
- Within multiple Collaboration Sessions (CS), by using services that are mostly synchronous, such as enhanced-presence services (chat, video conferencing…), synchronous edition of documents, etc.

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 mode* where each participant may alternatively (according to a turn-taking mechanism) act on a special desktop common to all participants of the collaboration session.

Collaboration as a Grid services exchange. Service is the central key concept underlying the GSD architecture. The notion of service is now at the centre of the development, implementation and success of distributed systems, such as Service Oriented Architectures, Web/Grid services, Multi-agent Systems and so on [23]. To provide a service means to identify and offer a solution (among many possible ones) to the problem of someone else. True service providing is not as simple as product delivery: a service is unique, adapted and customized by a special provider and for a special user in a special context. Following this definition, the exchange of services entails a kind of collaboration. One of the challenges of this work is to provide a purely service-oriented, collaborative architecture that enables developing an environment that is easy to learn and to use and that supports collaborative work. We will see how the Grid and Grid services provide such a service oriented architecture solution allowing the GSD to be itself a service and, in the same time, a service exchange environment. 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 this technology in all contexts and especially where human collaboration is required (e-learning, e-government, e-health, e-business, e-science).

Collaborative construction of a shared 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*. Our approach[2] consists of repeated short cycles of elicitation, specification, implementation, evaluation steps within a scenario: the collaborative construction of a shared ontology. 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 typically call for a set of interaction services. The objective of this real-world scenario was to show how the GSD may provide an engaging environment for helping users to reconcile, formalise and capture knowledge that is initially informal and distributed in the minds and documents of several people, into a shared ontology that can be then used within or outside of the GSD context (such as in an e-sciences or e-learning context).

Paper overview. The rest of the paper is organised as follows: In the following section we present the context of the paper, including an overview of related work and an identification of main requirements of collaborative environments. In Section 3 we present the Grid Shared Desktop, starting with the underlying Grid infrastructure; followed by the GSD architecture and its implementation, the GSD service, by means of virtual desktops. In Section 4 we detail the problem of collaborative ontology construction and explain how we addressed the problem within GSD experiments. Finally, we give some perspectives and conclusions in Section 5.

2. Collaborative environments

This section lays down the context of our study. First, we derive requirements from some ELeGI theoretical results about collaborative environment. Second, we provide a brief technological state of the art about Grid and collaborative tools.

2.1. Requirements for a collaborative environment

Informal learning. A crucial requirement of collaborative environments consists in enhancing learning as a result of communicating and exchanging services in a collaborative perspective. Learning happens in the context of collaboration and conversation 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

[2] We did not develop this work following a classic software engineering approach that completely specifies a well-identified problem before implementing it, but rather following an iterative approach in which the GSD elements are designed step-by-step according to the needs, constraints and requirements that emerge from real experimental situations.

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 collaborative 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 involve 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 both 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, an enhanced-presence environment with instant messaging functionalities [1,12], 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 the videoconferencing service 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 supported with just "one click" through a thin terminal and without installation of any third-party application. We consider this as a fully service-oriented approach.

2.2. Potentialities offered by Grid services

Of the available technologies for supporting and implementing a collaborative environment, the Grid technology offers many interesting aspects and ready-to-use components that directly address the requirements identified in the previous section. This section factually describes Grid concepts and synthesises how Grid fits the needs of collaborative environment.

Sharing resources among virtual organizations. The essence of the Grid is nicely reflected by its original metaphor: the mandate to the electricity network to offer us the service of providing us with enough electric power as we need it, when we need it, even if we do not know where and how that power is generated. At the end of the month, we pay a bill that corresponds to our consumption. The Grid 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 (Grid computing) and storage (data Grid) resources could be shared as needed and on demand; therefore Grid 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.

Grid services and Grid standardisation. Grid technologies have evolved from ad-hoc solutions, and de facto standards based on the Globus Toolkit, to Open Grid Services Architecture (OGSA) [4] which adopts Web service standards and extends services to all kinds of resources (not only computing and storage). Foster et al. call “service” [15]: A (potentially transient) stateful service instance supporting reliable and secure invocation (when required), lifetime management, notification, policy management, credential management, and virtualization. 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*[3]. More recently, the Web Service Resource Framework (WSRF) [13] defines uniform mechanisms for defining, inspecting, and managing stateful resources in Web/Grid services. For a recent precise overview of Grid service concepts and standardisation see for example [9]. Grid has now started to evolve toward the Semantic Grid [17,22] and the Learning Grid [5,21,24].

Grid as a basis for collaborative environments. Grid has a lot of interesting aspects for collaborative environments:

- Trusting the environment happens through Grid security mechanisms (e.g., X509 certificate and Community Authorisation Service);
- Trusting the environment also happens because of enhanced reliability provided by the Grid infrastructure;
- Synchronous and asynchronous modes of collaboration are brought by the technical layers of OGSA, thanks to stateful and dynamic services;
- Persistent memory is brought by Grid services which allow to integrate and capitalise upon the past thanks to stateful resources;
- Service dynamicity is brought by transient aspect of Grid services.
- Member dynamicty is brought by virtual organisation management.
- Usability is improved by the fact that Grid services are compliant with service-oriented architecture standards (i.e., WSDL, SOAP, UDDI, etc.). Grid services subscribe to the same logic of standardisation and interoperability as that of Web services (e.g., Semantic Web services, Business process management, XML, etc.).

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.

2.3. Current collaborative tools

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 solutions 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.).

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

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

2.4. Specifying the collaborative environment

As a result of identifying the requirements for a collaborative environment and of assessing the Grid specifications, we enunciate here the main functionalities that specify a collaborative environment. We have identified six important functionalities:

1) [**Notification]** to provide immediate awareness of the life of the community;

2) [**Membership**] to manage (add or remove) members;

3) [**Service**] to manage (import or remove) new services;

4) [**Activation**] to activate new instances of services;

5) [**Session**] to initiate a collaboration session;[4]

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

Section 3.2.2 shows how these six functionalities are mapped into six bootstrapping services of the Grid Shared Desktop architecture.

As we said in the introduction section, we distinguish two kinds of groups: a virtual community (VC) and a collaboration session (CS) (detailed in Section 3.2.1). We will further explain how this distinction allows us to address the synchronous and asynchronous modes of collaboration.

3. The Grid Shared Desktop

3.1. The Grid underlying infrastructure

As explained previously, the Grid provides the infrastructure that meets the GSD's core requirements. Figure 1 describes the Grid model as it is defined by the Open Grid Service Architecture (OGSA) [4,15]. At the very bottom are the Grid *resources* (for computation and storage). These resources are physically distributed anywhere and coupled in Grid *hosts* via networks. The role of Grid core mechanisms is to virtualise these resources and reify them in a secure and reliable manner:

Resource reification is achieved by Grid *service containers*. A Grid service container is a hosting environment for the service instances. It is allocated to (and created for) one and only one group of Grid users, called a *Virtual Organization*;

[4] Notice that within a collaboration session functionalities 1), 2), 3) 4) and 6) are required.

Secure and reliable access to service instances is ensured by *handles* and the exchange of *X509 certificates*. Every user must hold a valid X509 certificate in order to be a member of a virtual organization and thus be granted access to services. Grid hosts must also hold a X509 certificate to be recognised as valid Grid resource. Furthermore, the *Community Authorisation Service* (CAS) stores permissions (right levels) between virtual organization members and services instances in a service container.

According to Grid specifications, the unique service present by default in a service container is the CAS. Furthermore, our GSD architecture provides six additional bootstrapping services described in the next sections.

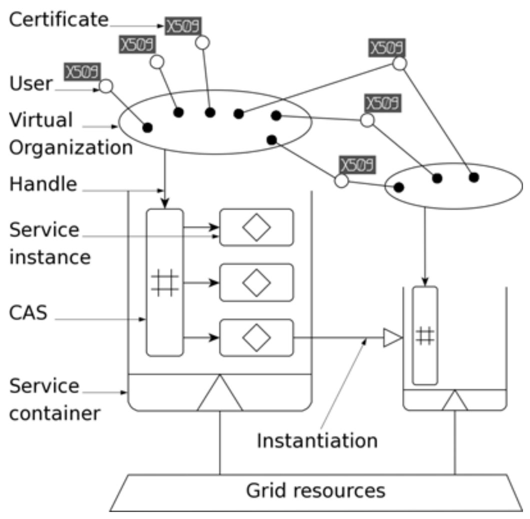

Figure 1. The Grid underlying infrastructure.

3.2. The GSD architecture

3.2.1. Users, VC and CS

The GSD architecture is an organisational structure based on three elements: users, VC and CS. A *user* who wishes to collaborate with other users must hold a means of authentication (following Grid security specification, e.g., a X509 certificate). This certificate allows identifying this user among others; it may be viewed as an electronic passport substituting any login or password. It realises the simple sign-on and identification needed to access the collaborative environment. In the GSD architecture, a user can become member of several *groups*. A group aims at sharing common goals by assembling, collaborating, and communicating in a loose, distant, virtual way, using network communication facilities, services and resources. A user can be member of more than one group. As mentioned before we distinguish two kinds of groups in the GSD architecture:

- A *Virtual Community* (VC), which has a lifetime in the order of months or years. We will further talk about "members" to represent users in a VC. Because of the lifetime of a VC, members know and trust each other. VC are composed of at least one member to dozen of members;

- A *Collaboration Session* (CS), which has a lifetime in the order of hours or days. We will further talk about "participants" to represent users in a CS. A CS is organised whenever a group of users of the same VC decide to collaborate in a synchronous mode for some time with some specific purpose like planning future work, finalising a document, and brainstorming. Participants share services and documents as well as communication tools. A CS necessarily involves more than one user but the number of participants cannot be very large.

General VC collaboration is mainly in an asynchronous mode whereas CS collaboration is most of the time in synchronous mode. However, the real argument for our distinction between a VC and a CS is related to the dynamics of the collaboration: VCs may engender CSs, while CSs may not engender CSs nor VCs. Resource virtualisation mechanisms of Grid infrastructure further supports VC and CS management. From a Grid point-of-view, VCs and CSs are both virtual organizations and therefore they are each associated to a Grid service container which will be destroyed 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 permissions 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 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.

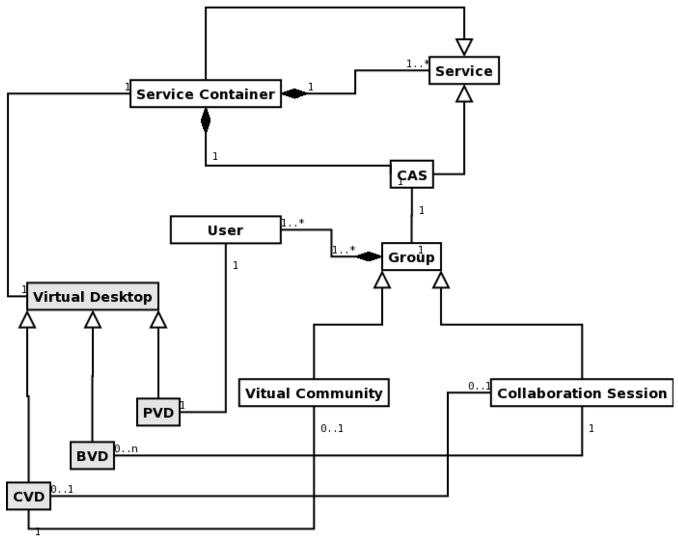

Figure 2. UML class diagram of the GSD architecture.

The power of Grid partly lies in the fact that a Grid service container is itself a Grid service instance that may be instantiated by another service (see instantiation on Figure 1). In the GSD architecture this allows a user to simply instantiate a new service container each time he or she wants to initiate a new VC or CS. The right part of Figure 2 (in white) shows a UML class diagram of these concepts.

3.2.2. Bootstrapping services

The GSD architecture is based on the capability of Grid services to be dynamic and stateful. The GSD architecture models the environment where a set of services are available within a Grid service container. 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* (the same kind of services is used both for VC and CS). They correspond to the six functionalities presented in Section 2.4 plus the CAS (cf. Figure 3).

Notification Service. This service enables VC members to be immediately informed on everything concerning the VC life: ready (online) members, available services, new members, new services, new results etc. It is composed of a set of notification mechanisms that inform a group that: a new service was imported, a service was activated, a CS was just initiated, a given member is online, a new member joined the VC, a common document was updated, etc. The Notification Service provides likewise a real time feedback on personal as well as other members' actions and work such as for example, information about a current CS. The Notification Service is tightly connected to the History Service which traces the community history (past life). The Notification Service is a pre-requisite to any interaction between VC members because it offers awareness of the presence and availability of the members.

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

Service Management Service. 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 Service Management Service offers users the ability to bring new tools as services for the group.

Service Activation Service. Any service available within a group service container may be activated by the Service 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 Service Activation Service allows using all of the other services in the GSD. The Service Management Service and the Service Activation Service play important roles in the ‘service dynamics’ of the GSD architecture.

Collaboration Sessions Management Service. 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 a 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’s service container the desired services using the local Service Management Service. Then the member invites participants to the CS with the local Member Management Service and manages rights between these participants and local services with the local CAS. The same member also activates the local services when needed with the local Service Activation Service. In the GSD architecture, the Collaboration Sessions Management Service is the only service capable of instantiating service containers.

History Service. An important aspect of collaboration is history of past actions and interactions. Indeed, actions and activities of group users may be traced and logged to keep a 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 user profile evolution, inventory past services activation, trace CS history, realise a shared document versioning, etc.

Community Authorisation Service. This service specifies users’ service rights levels (including for bootstrapping services). This services maintains a kind of members by services matrix. Members of a given group do not all have the same permissions over the services of that group. For example only user “A” as a service manager may be allowed to import new services for a group, while any user “B” would be allowed to initialise a CS. The term CAS comes from Grid specifications as explained previously.

CAS, Member Management Service, Service Management Service and Service Activation Service are needed as soon as we deal with new members, new services, and privileges of members on these services. These bootstrapping services, except for the Notification Service and History Service, can be invoked directly by members according to their own rights within a group. Notification Service and History Service are managed by underlying processes. The life of a CS has to be managed separately from the life of the VC; for instance, a local Notification Service might reflect the current turn-taking status of a CS. However, some local events of a CS may be reported to the whole VC by communication between Notification Services. In the same sense, it is also necessary to manage the history of a CS separately. But, when a CS is deleted the History Service of the CS communicates its data to the History Service of the parent VC.

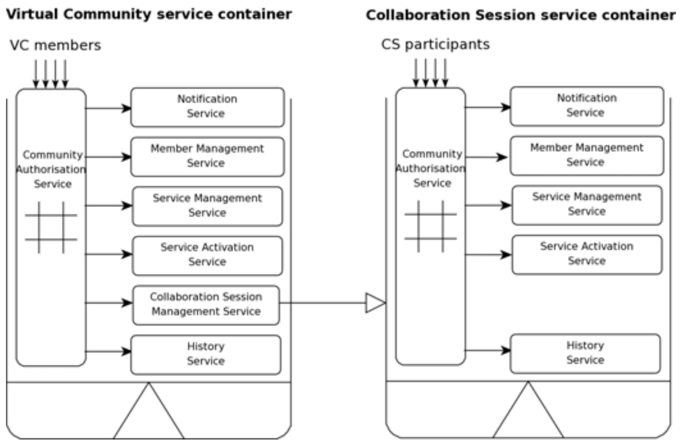
Virtual Community service container
VC members
Community Authorisation Service
Notification Service
Member Management Service
Service Management Service
Service Activation Service
Collaboration Session Management Service
History Service
Collaboration Session service container
CS participants
Community Authorisation Service
Notification Service
Member Management Service
Service Management Service
Service Activation Service
History Service

Figure 3. VC and CS service containers.

The last elements of the GSD architecture are non-bootstrapping services. All of them are imported in a group service container by the Service Management Service and activated by the Service Activation Service.

3.2.3. GSD processes

The bootstrapping services are of course implied in processes that correspond to the structure that defines the logical and temporal relations between these services and the interaction that users have with them. We detail here some of these processes. The *service activation process* is fundamental as it occurs each time that a service is invoked (a previously imported or a bootstrapped one). The s*ervice importation process* and *user introduction process* are quite similar and simple but are essential for the group dynamics. The *collaboration session management process* is more complex as it implies sub-processes with local CS services.

Service activation process. A member wanting to activate a service within a group inquires the Service Activation Service with its user certificate and the service handle (and eventually some other information such as: involved members, lifetime expected etc.). Then the Service Activation Service checks (by inquiring the CAS) the member’s permission level for this service. Then, the Service Activation Service creates the service instance with a specific state, lifetime and set of allocated resources. Next, it requests from the Notification Service to alert group members that a service was activated. In parallel, the Service Activation Service activates the History Service both for recording the service activation in the community history and for logging/tracing the user-service interactions until the end of the service’s life. The service is activated and group users may interact with it during the service’s life. At the end of the service’s life, the Service

Activation Service requests from the History Service both to stop recording and to store the service's results (by versioning past ones, etc.) and the Notification Service to inform the community of the results of the ended service. Afterwards, the Service Activation Service destroys the service instance and frees the allocated resources.

Service importation process. A member wanting to import a service in a group activates the Service 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 Service Management Service first includes the service in the group service container and gives it a handle. Then, it requests from the CAS to add a row in the members by services matrix. The Service Management Service has to specify the rights level of all members for the added service. Then, it requests from the History Service to add an entry in its service database, and give it the trace/log mechanism for the new service. Afterwards, the Service Management Service asks the Notification Service to inform the group members that a new service was imported and is now available (according to their rights level).

User introduction process. A member wanting to introduce a user in a group activates the Member 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 Member Management Service first stores the new user information such as status, name, address etc. Then, it requests from the CAS to add a column in the members by services matrix. The Member Management Service has to specify the rights level of all services for the added user. Then, it requests from the History Service to add an entry in its users list. Afterwards, the Member Management Service asks the Notification Service to notify other members, that a new member was added 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 elements: (i) the list of services that he wants the CS to benefit from; (ii) the list of members that he wants to see participate in the CS; (iii) the information necessary to build the local CAS matrix of rights levels. The Collaboration Sessions Management Service first instantiates a new CS service container. Then, it requests from the local Service Management Service to import each service desired by the CS manager and requests from the local Member Management Service to add each participants listed by the CS manager. These sub-processes are the same as 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 Notification Service to notify the entire VC of the creation of a new CS[5], (ii) the History Service to register the CS creation in the VC's 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 Service 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 Notification Service to inform the VC of the set of results of the ended CS.

3.3. The GSD service implementation

The GSD is composed of several active desktops that play simultaneously the roles of service containers and graphical user interface. The term desktop means a work environment where users can dynamically activate services. A Grid service available in a service container has a graphical metaphor on the desktop called a "shortcut" (e.g., an icon, a window). A desktop may be accessed in full control mode or in view-only mode. This is analogous to the read/write (R/W) or read-only (R) modes in a file system. In the former case, a user can both see and act on any desktop elements, in the latter, a user can only see actions performed by others on the desktop.

3.3.1. Collaboration virtual desktops

The GSD service is defined in Figure 4. We distinguish three kinds of desktops:

- The *Private Virtual Desktop* (PVD)
- The *Common Virtual Desktop* (CVD)
- The *Broadcasted Virtual Desktop* (BVD)

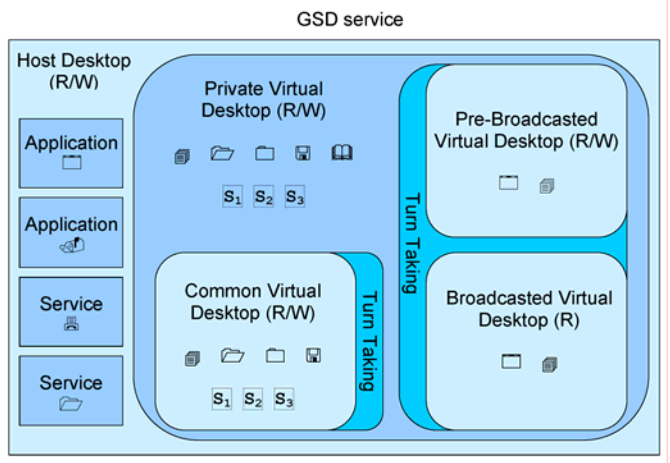
GSD service
Host Desktop (R/W)
Application
Application
Service
Service
Private Virtual Desktop (R/W)
S1 S2 S3
Common Virtual Desktop (R/W)
S1 S2 S3
Turn Taking
Turn Taking
Pre-Broadcasted Virtual Desktop (R/W)
Broadcasted Virtual Desktop (R)

Figure 4. The GSD service's set of virtual desktops.

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

Table 1. Bootstrapping services used in each virtual desktop.

	Community Autorisation Service	Notification Service	Member Management Service	Service Management Service	Service Activation Service	Collaboration Sessions Management Service	History Service
PVD	X	X		X	X		X
VC-CVD	X	X	X	X	X	X	X
CS-CVD	X	X	X	X	X		X
BVD	X	X		X	X		X

The Private Virtual Desktop could be also called the ubiquitous desktop. It can be accessible from anywhere and contains personal settings, documents, applications, bookmarks. A user has full privileges to import and activate any service here[6], store and retrieve private documents, /files/data, or even check an email account. A user may access his PVD either directly from a thin terminal or via a host desktop (e.g., Windows, Linux KDE, X, MacOS).

Via a PVD, a user has access to one Common Virtual Desktop for each VC that this user is a member of. This kind of CVD ("VC-CVD" in Figure 5) is a desktop which belongs to everyone in the VC. It is not the PVD of one of its members, but another desktop, shared, that everybody, according to a specific turn-taking mechanism, can act upon (see also Figure 4). The CVD contains the VC settings, documents, applications, bookmarks etc. Any member of the VC can import and activate services in the CVD.

The GSD bootstrapping collaboration context is constituted, for a given user, of a set of desktops: one PVD and a CVD for each VC that the user is a member of. In order to address the question of CS, the GSD also uses desktops. For each VC a user is a member of, he can activate the Collaboration Sessions Management Service of the VC and run two modes[7] of collaboration within this community (cf. section 2.3):

- *Screen sharing mode.* In this mode each CS participant is owner of a (Pre-)Broadcasted Virtual Desktop. This PreBVD can be broadcasted to all other participants. The BVD is the desktop every participant see at the same time. Notice, that a turn-taking mechanism specific to this mode is mandatory.
- *Common environment mode.* In this mode all the CS participants share a CVD dedicated to this CS. This kind of CVD ("CS-CVD" in Figure 5) is the same as a VC-CVD. Any CS participant can alternatively act upon the CVD following a turn-taking mechanism.

[6] It is important to notice that we do not talk about applications anymore; functionalities available on the desktops are only services.

[7] A *desktop sharing mode* is also possible (because a BVD can technically be seen and acted upon by all CS participants) but it presents some big drawbacks from a security and privacy point of view. Moreover, all the benefits from this mode are included in the common environment mode.

Consequently, we may distinguish four types of virtual desktops: The CS-CVD implements the CS service container. The VC-CVD, PVD and BVD implement the VC service container. Nevertheless, some bootstrapping services are not usable as Table 1 shows. For example, a member can not introduce users in his own PVD and consequently he can not create a CS alone.

Remark: The left part of Figure 2 (in grey) completes the UML class diagram with these new concepts.

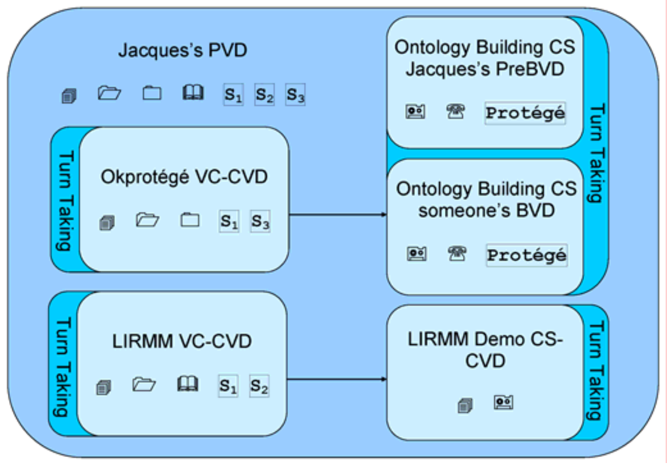

Figure 5. A PVD example in the GSD.

Figure 5 illustrates a user's PVD: Jacques is a young researcher at LIRMM. He is a member of the LIRMM VC and of the "okprotege" VC. As a member of these VCs he has access to their CVD from within his PVD. The LIRMM director uses the LIRMM's Collaboration Sessions Management Service to start a one day CS in common environment mode to allow young researchers (i.e., forming a subgroup of all LIRMM members) to show demonstrations of their work with a video replay service. The same day, the okprotege team has a short (one hour) CS in screen sharing mode to work on the problem of collaborative ontology construction problem, using the Protégé ontology-development service (further exposed in Section 4).

3.3.2. Processes in the GSD service

General case. Using a service in the GSD means to activate that service's shortcut in the appropriate desktop. This action starts the service activation process (hidden for the user) in the corresponding service container and runs the service's interface (e.g., a window-based GUI) that allows the user to interact with the service during the service's life.

Specific cases. Bootstrapping services of the GSD architecture have special implementations that we detail here:

- The Notification Service is realised by enhanced-presence indicators, turn-taking, or other services allowing immediate awareness of the life of the group (for example email, instant messaging, sounds, etc.). The Notification Service is very important in the GSD service as it makes the real binding between a member and a group.
- The Member Management Service, when called at a VC level to introduce a new member, creates (if necessary) a new PVD (i.e., a login/password/private space set), and grants access to the VC-CVD in the created PVD. When called at the CS level, to introduce a new participant, the Member Management Service simply includes access to the CS-CVD in the user's PVD.
- The Service Management Service adds the imported service's shortcut in the desktop corresponding to the service container in which the Service Management Service is called.
- The Service Activation Service is not directly accessible, but called each time that a user activates a service in a desktop.
- The Collaboration Sessions Management Service is able to create a BVD or a CS-CVD according to the collaboration mode that the CS manager chooses. In screen sharing mode, a PreBVD is created for each participant of the CS; in common environment mode, a CVD is created for all the participants of the CS. Services that the manager imports for a given CS are available in the BVD or CVD as shortcuts.
- The History Service is realised by a set of scripts that log and trace a group's history. This history is, of course, interfaced and available for group members.

4. 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—a computerised representation of knowledge—using a renowned ontology-building tool supported as a service on the GSD, as well as additional human-communication services.

4.1. 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, 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 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.

4.2. 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.

4.2.1. 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.

4.2.2. 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 PVD in a Web browser. The chosen collaboration mode for these CSs was screen sharing mode. Each PVD was equipped with necessary collaborative and preference-setting services as well as a one-click toggle granting full access to the member’s own PreBVD, and a read-only access to the BVD. At any time, participants could switch from operating the services on their PVD 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 PVD 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[8]. 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.

4.3. 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 6). 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.

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 early experiments did not lead us to a full production phase in a newly-created ontology of particular interest to the okprotege community,

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

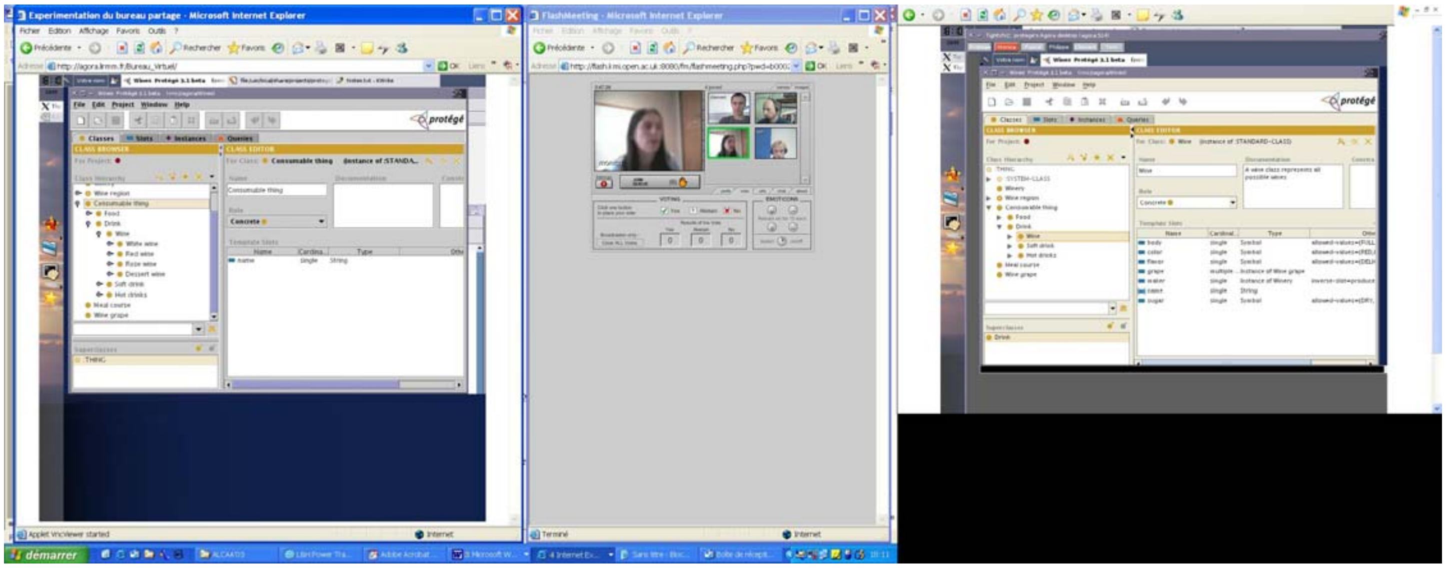

Figure 6. Screenshot of the Grid Shared Desktop in action, during our first collaboration session experiment.

they still demonstrated production-level creation of new concepts and properties in the tutorial ontology.

Figure 6 shows a screenshot of the GSD during one experiment. Left, the Web-accessible PVD 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 PVD 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 points 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.

5. 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.

EnCOrE: Encyclopédie de Chimie Organique Electronique (Computerised databases describing compounds exist, but have the same conceptual organization as the reference tomes from the 19th century, and are often misused; EnCOrE aims to address this problem.)

Disclaimer

A current prototype of the GSD service was deployed on the LIRMM Grid node (at http://agora.lirmm.fr/) and was used within the scenario presented in this paper as well as in the large scale scenario with Chemists. It is based on Remote Frame Buffer Protocol (RFB) [20]. Virtual desktops are KDE Linux desktops and only some services currently available in the GSD are true standard Web/Grid services (the rest are ad-hoc applications for now).

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.

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 Special Interest Group during 1997-98 and participated to ETSI standardisation. He is main author of seven international publications and co author of six articles. 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. As part of the Protégé research effort, 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 holds a BS engineering degree from Ecole Polytechnique Féminine, France and she received her PhD in Computer Science from the Université de Nice-Sophia Antipolis and the Institut National de Recherche en Informatique et Automatique, on the subject ok knowledge-based program supervision for medical image processing. http://smi-web.stanford.edu/people/crubezy.