Towards a Generic Service Oriented Framework for Integrated User Management of Virtual Communities

Konrad Wulf

Towards a Generic Service Oriented Framework for Integrated User Management of Virtual Communities

Konrad Wulf
High Performance Computing Center Stuttgart (HLRS), Stuttgart, Germany
wulf@hlrs.de

A central Service layer for the European eLearning Grid Infrastructure (ELeGI) is the one for the management of members and the provision of collaborational services (Virtual Community Services). Because it is a lot of effort to totally newly (re-)write the collaboration services like e.g. a videoconferencing service, it is much more desirable to take existing applications and just to wrap them with a Web Service interface to programmatically access the existing functionality relevant to user management. Thus, it is intended to create a pic'n'mix architecture, where existing collaboration tools can be easily added or removed.

Furthermore it will be thought about how the user interface could look like for the members of a Virtual Community, providing single sign on and making the switching between the tools as smooth and elegant as possible while always keeping an eye on the work involved for developers and administrators.

In the area of integrated user management, Web and Grid Services can live up to their full integrational power, providing real added value with the achieved interoperability. This article focuses on the Virtual Community service layer and provides first deeper thoughts on how the architecture regarding user management and user interface integration could look like.

Keywords: Integration, User Management, heterogeneous collaborative services, Web Services, Grid, Architecture draft, Virtual Communities, eLearning.

1. VIRTUAL COMMUNITIES IN THE CONTEXT OF THE ELEGI RESEARCH PROJECT

Collaboration and enhanced presence is a major building block of a pedagogically sound eLearning infrastructure. This is why one focus of the European Learning Grid Infrastructure (ELeGI)[1] project has been put on these aspects. The picture below illustrates the preliminary architecture of the ELeGI as it has been outlined in the recently published project's deliverable D 13 (see **FIGURE 1**).

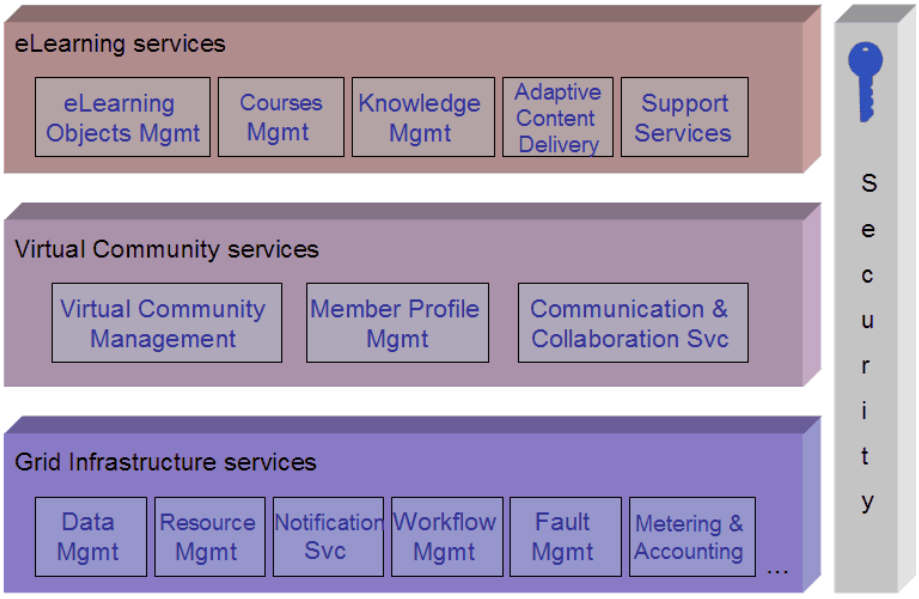

FIGURE 1: The preliminary ELeGI architecture as outlined in D 13

The architecture foresees that on a foundation layer, there will be essential Grid Services that do not depend on any application domain and also do not require virtual communities. Then, on top of it, you will find everything

[1] The official project web site can be found at http://www.elegi.org

related to the lifecycle management of virtual communities, which is also quite generic in a sense that there are many application domains that benefit from such a layer, like e.g. collaborative engineering or international research projects. The uppermost layer then includes the domain specific eLearning services, such as courses management and knowledge representation. It should be noted here that this doesn't necessarily mean that these layers are totally loosely coupled. For example, it might be reasonable that the member profiles in the community layer also include domain specific learner-profile information.

For this article, we will concentrate on the layer in the middle, the "Virtual Community Services" layer. In the following we will draft an architecture for integrated user management in a architecture based on Grid and Web Services. Existing and adjacent approaches will be discussed and compared with the design proposed in this article.

2. DRAFT FOR THE VC SERVICES LAYER ARCHITECTURE

The centre piece of the preliminary architecture for the VC Services layer is the VC Controller which incorporates the master user database as depicted in **FIGURE 2**. From there, all the control information flow is moderated. In relation to each VC Service that is intended to be plugged-in for a particular VC, a person belonging to the administrator group uses the VC Controller to invoke the set up of a new instance of that service, maintains the users and also terminates the created instances. This is steered by the VC Administrator's user interface.

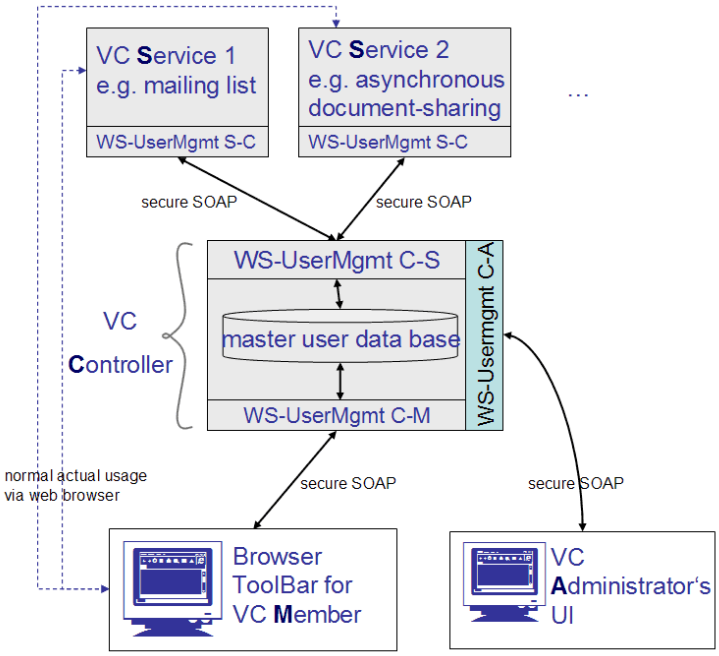

FIGURE 2: A preliminary general architecture for Integrated User Management in a Web Services World

For the maintenance of the VC members, the framework needs to support a mapping of user credentials because the VC services do not necessarily all allow the same syntax for user identifiers and they certainly foresee different passwords or certificates. Also, a initial user identifier chosen by the controller (probably the email address) or the user might already be in use by someone else. Each VC service then in return needs to provide the controller with the following information: icon for the service, a list of personalized activities available with corresponding URLs or command paths in XML format (see **FIGURE 3**), as well as the chosen user credentials for the central user identifier.

Another important issue is *event handling* that also should be possible within the proposed architecture. For example, in a Calendar VC service there might be a new calendar instance that a VC member has been requested to respond to. He should be notified of this (also) via the VC user interface, which we for the time being have decided to be a browser toolbar (see section 3). Also, for Quality of Service reasons, the VC Controller should check the availability of the VC service in regular intervals and notify the VC Members and the VC Administrators when the service is down. Possible candidates for realizing the event handling functionality are the two specifications WS-Notification and WS-Eventing (see [1] and [4]).

When speaking of a central user directory service, many people probably think of LDAP servers and, indeed, there are even Web Service extensions for LDAP available as specified by the OASIS standard DSML (see [8]). An LDAP server can do a lot of the things that the VC Controller is required to do. For each user it can store all sorts of information, such as a user profile, XML documents and even public security keys. Moreover, it can communicate asynchronously (see [2]). But there are some fundamental differences in design compared to the chosen approach here:

1. There is just one user identifier and just one password. No mapping of user credentials is foreseen.
2. It would require the VC service to understand the ldap protocol, so each VC service would need to have a built-in ldap client. 'Whether that is a disadvantage really depends on the efforts involved with it.

Still, it might be useful to cover parts of the Controller's functionality with an existing LDAP server. Moreover, some collaborative tools do already support authentication via LDAP, e.g. the jabber server "Jive Messenger" (for chat), so it would probably be good to take advantage of that.

```
<offered-service-activities>
    <service-name>Calendar</service-name>
    <service-description>service for sharing parts of your calendar for finding the best date for a meeting</service-descr
    <icon-url>http://www.meetomatic.com/images/cal_logo.gif<icon-url>
    <authentication type="HTTP">
        <user>someLogin</user>
        <password>somePW</password>
    </authentication>
    <activity enabled="true">
        <display-name>set up new calendar</display-name>
        <target-path>http://www.meetomatic.com/calendar.asp?name=Jeffrey+Jefferson&notify=jeff@jefferson.biz</target-path>
        <target-type>URL</target-type>
    </activity>
    <separator/>
    <activity enabled="true">
        <display-name>calendar meeting 'XY'</display-name>
        <activity enabled="false">
            <display-name>send request to VC member</display-name>
            <target-path/>
            <target-type/>
        </activity>
        <activity enabled="false">
            <display-name>view results so far</display-name>
            <target-path/>
            <target-type/>
        </activity>
        <activity enabled="true">
            <display-name>respond</display-name>
            <target-path>http://www.meetomatic.com/answer.asp?id=AMD432</target-path>
            <target-type>URL</target-type>
        </activity>
    </activity>
    <!-- [...] -->
</offered-service-activities>
```

FIGURE 3: The Activity Profile of a VC Member in relation to a particular VC Service

Another approach highly related to the outlined architecture is the Web Services framework for federated identity management of the Liberty Alliance [6]. The framework seems very suitable as it explicitly addresses the issues of cross-boundary Single Sign-On (SSO) and is, as the name already suggests, targeted at service oriented architectures. The framework covers the issues of authentication, message protection, service discovery, policy, access of common data and even transport protocols. Liberty ID-WSF builds on the standards Secure Assertion Markup Language [5] as well as IETF's Simple Authentication and Security Layer (SASL). The latter is a widely used standard and also used by LDAP. A central design decision taken in Liberty Alliance's framework is that the authentication is provided by a third-party, the so called "identity provider". Again, this is what it has in common with LDAP, but a central difference to what is proposed with the draft architecture outlined in this paper. While it makes sense to have a centralized user directory in a single hierarchical organisation it probably is different in a Virtual Community, when collaborational services are provided by several independent organizations and partners change in different contexts. Also, the effort involved in reengineering a collaborational service to retrieve the user information from an external directory is probably more than just exposing some already existing methods via a standardized Web Service interface. Also, the redundancy proposed through the user mapping in the VC Controller may diminish the risk of having a single point of failure.

But what is even more important, is that these approaches do not foresee that access to the integrated services can be joined together in a single user interface on the client side, which is believed to be a central advantage of the sketched architecture described before. To be more precise, the steering of user interfaces is out of scope of the LDAP protocol and the Liberty ID-WSF framework.

3. DESIGN CONSIDERATIONS FOR THE USER INTERFACE – VC CONTROLLER RELATION

The Internet Browser with its plug-in architecture has more and more become a command centre for all sorts of Internet based media and applications. Furthermore, a browser is meanwhile part of every popular standard distribution of operating systems for desktop computers. Therefore it seems more than natural to integrate the organizational view of a VC somehow into the browser environment. Currently two ways of doing it seem most appealing: a Browser Toolbar and an integrated Web page that uses Portlets for Web services. In the following we shall take a closer look at the first option.

The author believes that all the collaborative tools used in a virtual community should be bundled together in a place that is just one fingertip away from the situation where the eWorking/eLearning takes place. Also, the user doesn't want to be bothered keeping in mind all the instructions on how to access each tool. Moreover, he would like to be notified of major events that affect him (and only those). The VCtoolBar as depicted in
FIGURE 4 is a proposed solution for this. Going from left to right, you first see the VCtoolBar icon, which leads to the general menu offering dialogs such as "preferences", "refresh" and "help".
Then, there comes the actually interesting part. Each icon represents a VC service that is available to that particular virtual community:

- "chat" (here: a jabber client named Buddyspace),
- "mailing lists" (there can be several subgroups in the VC),
- a "calendar" for scheduling synchronous meetings,
- a "videoconferencing" tool,
- a "desktop sharing" tool for synchronous use of any type of application (e.g. co-editing of a MS Word document),
- a "document repository" for asynchronous sharing of media (like e.g. eGroupware)
- a topic-centred discussion forum

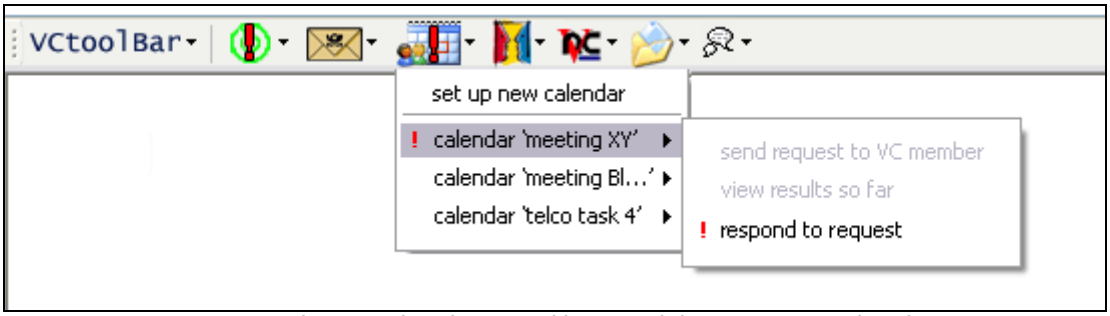

FIGURE 4: The VCtoolBar integrated into a web browser as user interface

In my opinion, the "desktop sharing" is a very interesting service as it can potentially provide a vast range of shared, interactive material. On the other hand, because of the richness of possibilities, such a service is also potentially a high security risk for an organizational network, so that such a service should be isolated by a firewall on the network layer and run with a restricted user that has only permissions for the intended applications. In the picture below, people might recognize the VNC icon that has been used for "desktop sharing". VNC is a protocol and software that has been originally designed for remote computer administration and so doesn't support scheduling and sessions. To make it a proper VC service it therefore needs to be combined with a scheduling system.

The enumerated services above are, of course just an example, and any other VC service should be possible to integrate via the envisaged framework. A more comprehensive systemisation of (possible) collaborative services and available standards is available at [10].

Concerning the user interface and the requirements derived from it the following issues need clarification:

1. How can the client tool for a collaborative service be launched on the basis of what an entity outside provides with the bundled activity profiles? As the composition of the displayed menus are directly steered by the VC Controller, the controller can't usually know where a client application has been installed to. In general, there are three possible ways, that could be used as a work around for this:

- easiest: an URL with http authentification or "deep link" authentication (authorisation string as HTTP GET parameters): the browser already knows how to handle this.
- Java Web Start
- Look for environment variable analogous to "%JAVA_HOME%" like e.g. "%BUDDYSPACE_HOME%" and assume standard sub path to executable file. The problem here is that it is not clear how the VOtoolbar can provide the user credentials to the client in a standard way except for passing the login parameters along the command line.

2. How can the notification that is shown in **FIGURE 4** with a red exclamation mark be turned into real? Can WS-notification or WS-Eventing be used for this type of event handling? What about firewalls and server triggered communication?

4. CONCLUSION

While probably most of the readers will agree on the overall benefit of a framework for integrated user management that can easily integrate existing collaborative tools using secure and asynchronous Web Service technology, there is just as probably disagreement on how this framework should look like. Here we have sketched one possible architecture that looks like a promising approach to the author and have compared it with the approaches chosen by LDAP and Liberty ID-WSF. But a good framework can only mature through discussion, explorative prototyping and further elaboration. Therefore, feedback on this draft architecture is more than welcome.

REFERENCES

[1] Czajkowski, K. et al. (2004). *The WS-Resource Framework, Version 1.0.* (http://www.globus.org/wsrf/specs/ws-wsrf.pdf)
[2] Donnelly, M. (2000). *An Introduction to LDAP.* (http://ldapman.org/articles/intro_to_ldap.html)
[3] Foster, I. et al. (2004). *Modelling Stateful Resources with Web Services, Version 1.1.* (http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf)
[4] Geller, A. et al. (2004): *Web Service Eventing (WS-Eventing)*. (http://msdn.microsoft.com/ws/2004/08/ws-eventing/)
[5] John Hughes, Eve Maler (2004). *Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1.* (http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf)
[6] John Kemp et al. (2004). *Liberty ID-WSF – a Web Services Framework*. (http://www.projectliberty.org/resources/whitepapers/Liberty_ID-WSF_Web_Services_Framework.pdf)
[7] L. Pearlman, C. Kesselman, V. Welch, I. Foster, S. Tuecke (2003). *The Community Authorization Service: Status and Future.* (http://www.globus.org/security/CAS/Papers/CAS_update_CHEP_03-final.pdf)
[8] OASIS (2001). *Directory Services Markup Language v2.0*. (http://www.oasis-open.org/committees/dsml/docs/DSMLv2.doc)
[9] Wahl, M. et al. (2000). *Authentication Methods for LDAP*. http://www.ietf.org/rfc/rfc2829.txt
[10] Wilson, S., Blinko, K. and Rehak, D. (2004). *An e-Learning Framework - a Summary.* (http://www.jisc.ac.uk/uploaded_documents/Altilab04-ELF.pdf)