Linking together reservation systems and remote labs

Jasper Bedaux[1], Leendert van Gastel[1], Theo Koreneef[2], Jos Loonen[3] and Kees Uiterwijk[4]

[1] Universiteit van Amsterdam, Faculty of Science, AMSTEL Institute, Amsterdam, The Netherlands
[2] Haagse Hogeschool, TH Rijswijk, Academie voor Engineering, Rijswijk, The Netherlands
[3] Fontys Hogescholen, Institute for Information and Communication Technology, Eindhoven, The Netherlands
[4] Hogeschool Utrecht, Institute for Information and Communication Technology, Utrecht, The Netherlands

***Abstract*— In the technical track of a new remote labs project called Labs on line, we will look at the linking of two reservation systems for remote labs. Instead of integrating both systems into one new system we are exploring possibilities of designing a generic interface for remote lab reservation systems.**

By using this approach organizations will be able to keep and manage their own reservation systems while linking them with the labs of other organizations. This is a very flexible and scalable solution, both from a technological point of view as from an organizational point of view. Individual implementations can thus be customized and integrated with other systems without affecting existing reservation systems.

***Index Terms*— Remote labs, reservation system architecture, online experimenting.**

I. INTRODUCTION

In the recent past, two different remote lab projects in the Netherlands were finished: *e-Xperimenteren+* [1] (Universiteit van Amsterdam, Universiteit Twente, Vrije Universiteit Amsterdam, Fontys Hogeschool Eindhoven) and *FLEXlab* [2] (Hogeschool Utrecht, Technische Hogeschool Rijswijk). Both projects built a reservation system and several remote experiments involving physics experiments (e-Xperimenteren+) and telecom and electronics experiments (FLEXlab).

Recently, a new project has been started called *Labs on line* that will build on the results of the two preceding projects. The new project will consist of several tracks: an existential track (why remote labs are useful), a content track (selecting and building remote experiments), a didactical track (embedding of the remote labs into education), an organizational track (how to continue and manage the remote labs after the project finishes) and a technical track.

This paper will focus on the technical track, especially on how to link together existing (or new) reservation systems for remote labs. We are looking at the possibility that a student can make a reservation into one reservation system using another, different reservation system.

Because the project is in the definition phase, a lot of choices are not made yet and we are open for suggestions and cooperation with other projects. Especially for the topic of linking together remote lab reservation systems cooperation is important evidently. Therefore we invite related projects or projects that are interested in cooperation or projects that might benefit from the results of our project to contact us.

II. PROPOSED LINKING OF SYSTEMS

When the reservation systems of the projects e-Xperimenteren+ and FLEXlab were compared, it turned out that, although there are remarkable similarities, each system has its own features and strengths. The projects that designed them had different wishes and requirements, e.g. one of the projects did want a feature to save individual marks for students, while the other system put more effort in a sophisticated system for determining the rights to make reservations. Also, the systems are implemented using different technologies. When we look at future wishes, there is a clear wish to integrate the reservation system with the existing Electronic Learning Environments (ELE) of the participating institutes and to establish a connection to the student administration systems of the participating universities.

One solution to bundle the results of the two preceding projects would be to redesign one of the systems or build an entirely new system. We decided that it would be more flexible to keep both systems side by side, but link them together through a generic interface.

To ensure maximum flexibility some starting points are determined:

1. Each participant may have its own reservations system, i.e., there is no central system and there are no requirements about the platform or programming language of a reservation system.
2. A user (student) only needs one portal to make a reservation, access experiments and access logs and/or measured data.

There are a lot of reasons why not to use one central system but instead keep the possibility of different systems open, we name a few:

1. Organizations may want to administer and manage their own systems; transferring control to a central system may be undesirable from an organizational point of view.
2. Distinct systems already are in use and many organizations may not be willing to replace their existing systems.
3. Different universities may want to integrate their reservation system with their own Electronic Learning Environment (ELE) or student administration system.
4. Organizations may want to add custom functionality and change appearance or language of the interface.

III. RESERVATION SYSTEM INTERFACE

In this section we are talking about an interface between two different reservation systems, i.e., a machine-machine interface. The general requirements are that a user of reservation system A can make a reservation for an experiment or object of reservation system B while only interacting with reservation system A.

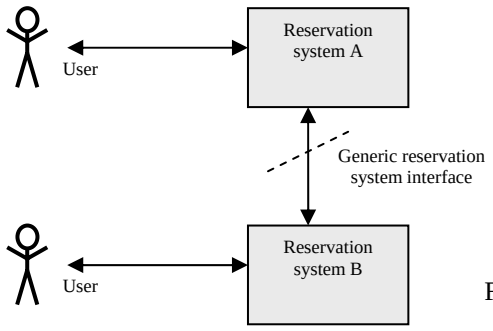

Figure 1

In this early stage of the project the actual requirements and the interface itself are not designed yet, but we can already describe some functionality to get an idea of this interface:

1. Get list of experiments / objects that are available.
2. Get scheduling policy (sometimes reservation is not needed in case of very short experiments).
3. Get a list of available time slots for a certain experiment / object.
4. Make a reservation for a certain experiment in a certain time window.
5. Get a list of reservations that the user made.
6. Cancel an existing reservation.
7. Get the user authentication information for a certain reservation, see below for details.
8. Get the URL of the experiment / object.
9. Get logs and or measured data of a certain experiment.

The *user authentication information* consists of information that is necessary to authenticate a user. This can be the current IP-address of the user or a username and password hash or a session ID. By communicating this information, the reservation system in which the reservation is made does not need to connect to the user database of the reservation system that made the reservation when the user wants to log in to the experiment or object computer. This is very important in order to keep the design flexible: each reservation system can use a custom solution for (a connection to) a user database, independent of the other systems.

It must be decided also how the interface will be implemented. One of the options is to use web services [4]. Libraries to easily implement web services exist for many platforms and programming languages and web services are based on simple and widely used standards like SOAP [5] (uses HTTP) and XML.

IV. AUTHORIZATION SERVICE INTERFACE

Just as for the reservation system interface, it can be useful to specify an interface between the reservation system and an authorization service.

Of course, it is not *required* to implement this interface, e.g. when the authentication / authorization service is already integrated into the reservation system. In this project, the reservation systems and authorization systems of the projects e-Xperimenteren+ and FLEXlab were compared. Although the different systems were designed and built independent of each other, it turned out that the general architecture of both systems has a lot in common. For example, both projects already had an authorization service that was separate from the reservation system, so interfaces are already present. By using a generic interface it will be possible to exchange the different authorization modules.

Below is an overview of the general architecture of a system where the authentication & authorization is split off from the reservation system:

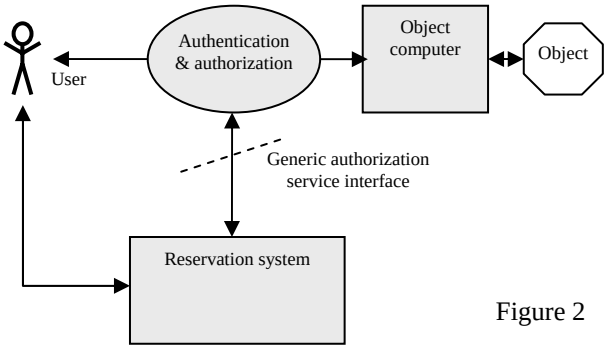

Figure 2

The authentication & authorization service is between the user and the object computer (or experiment computer). The authentication & authorization service gets the user authentication information from the reservation system. The user authentication information can for example be the current IP-address of the user or a username and password hash or a session ID. The reservation system might get the user authentication information from another reservation system (in case the

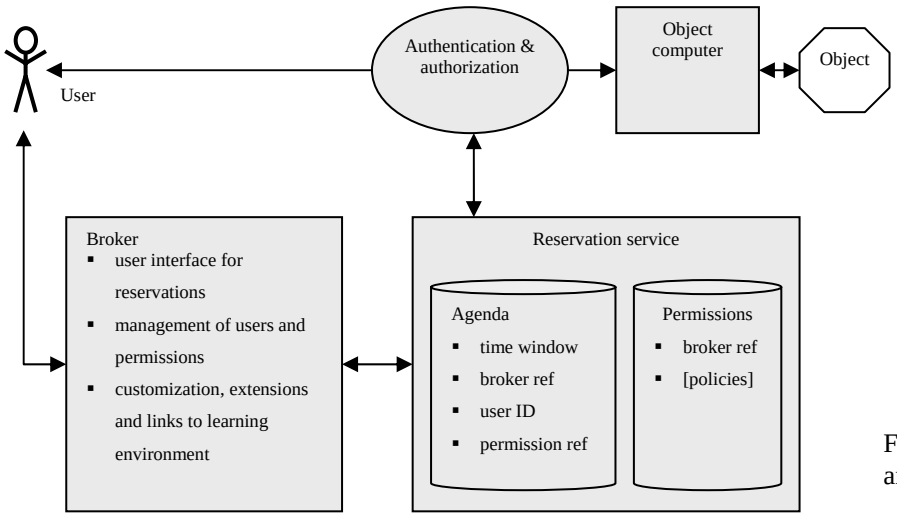

Figure 3: Multi-broker architecture

user made the reservation from another reservation system, see above).

In the earlier two projects, three solutions for the authorization service were built. All these solutions use authentication on the base of the current IP-address of the user.

1. A hardware router is dynamically programmed to let pass or block traffic from the IP-address of the user that wants to access the experiment or object.
2. A Network Address Translation server checks the reservation table and either assigns an object computer to the user or blocks the user if no valid reservation was found.
3. An access list of a LabVIEW server that serves remote panel applications is dynamically updated by a LabVIEW software component.

The first two solutions are very generic: they can be used for any remote lab interface, it doesn't matter what kind of application or protocol is used. The third solution is very convenient when the LabVIEW remote panel technology is used: no additional hardware is needed; just a simple LabVIEW software component can be added.

This interface is also not yet designed in this early stage of the project, but we can already name some of the functionality:

1. Get the user authentication information (IP-address or username + password hash or session ID) of the user that has currently a valid reservation.
2. Get the start and end time of the current reservation.

V. MULTI-BROKER ARCHITECTURE

In the project e-Xperimenteren+, also a distributed model was designed, where the reservation system is split up into a *broker* and a *reservation service*. A similar architecture was also implemented in the project Co-Lab [3], another project where remote labs figure. In this section we take a look at this architecture because it uses the interfaces described in III and IV and because this architecture adds flexibility and scalability because the broker and reservation service modules can be duplicated and customized independently. Note that

1. It is not *required* to implement this architecture to link a reservation system by using the generic reservation system interface of III; it can however be helpful when designing a new architecture or redesigning an existing one.
2. In the presented architecture below many details and choices are still open for discussion.

In figure 3 is a short overview of the multi-broker architecture. In this picture:

- Each component can appear multiple times.
- A broker can communicate with multiple reservation services and a reservation service can also communicate with multiple brokers.
- A reservation service can communicate with multiple authentication & authorization modules, but an authentication & authorization module can only communicate with one reservation service.
- An authentication & authorization module can service multiple object computers, but an object computer is protected by only one authentication & authorization module.

Permissions at the reservation service are permissions at the level of brokers, i.e. they determine the permissions of each broker that communicates with it. The broker is responsible for managing the permissions of its individual users. A reservation service can implement *policies* about the maximum number of reservations and dates and times a broker may make reservations. This way it is easy for example to guarantee that on a certain day only a certain broker is allowed to make reservations.

Apart from the permissions the reservation service contains a reservation table or agenda that keeps track of reservations. A reservation consist of a time window, a

reference to the broker that made the reservation, a user ID and a reference to the permission used to make this reservation from the permission table. The reference to the broker is used to keep track of which broker made which reservation, so it can be determined whether a broker is allowed to view, cancel or update a reservation. The user ID is a local user ID used at the broker that made the reservation. Although the reservation system itself might have not much use of it, it is useful in the communication with the broker that made the reservation.

Functionality of the broker is providing an interface for the users to make reservations, the management of users and their permissions (to make reservations) and integration with other systems like the ELE (Electronic Learning Environment) or student administration of the institute. Also customizations like additional features and layout and language of the interface can be built into the broker.

Advantages of the multi-broker architecture are that the reservation service component can be kept simple and robust (this is the part that communicates with external brokers / reservation systems) because all customization like interfaces to electronic learning environments and user management can be built into the broker part. Because each module can be duplicated at different locations, this architecture has a very good scalability. Duplicating the broker module can be useful for several reasons. It can be convenient for example to have several brokers for different target groups, e.g. one broker for internal students that is integrated with the student administration system and another broker for external visitors that has another user management and possibly other policies. Also the layout and functionality can be different for different brokers.

VI. Conclusions

By specifying an interface between reservation systems it becomes possible to link together different reservation systems without the need of a central system. Care must be taken that the interface is simple and that as few as possible restrictions are set for the reservation systems that must be linked together. This will ensure that it is relatively easy to implement the interface into existing solutions.

There are a lot of advantages to the approach of using an interface between systems instead of integrating them into one central system. The most important advantages are that the solution is very scalable and flexible (both from a technical point of view as from an organizational point of view) and that customization and integration with other systems is easier, especially when using the multi-broker architecture.

In our project the interface is only needed for two different reservation systems, but it is a nice chance for cooperation with other initiatives or standards and a nice chance for other projects to link their remote labs together.

References

[1] e-Xperimenteren+ website: http://www.science.uva.nl/remotelabs/
[2] FLEXlab website: http://www.flexlab.nl/
[3] Co-Lab project website: http://www.co-lab.nl/
[4] Webservices: http://www.w3.org/2002/ws/
[5] Simple Object Access Protocol http://www.w3.org/TR/soap/

Authors

Jasper Bedaux is with the Universiteit van Amsterdam, Faculty of Science, AMSTEL Institute, Kruislaan 404, 1098 SM Amsterdam, The Netherlands (e-mail: bedaux@science.uva.nl).

Leendert van Gastel is with the Universiteit van Amsterdam, Faculty of Science, AMSTEL Institute, Kruislaan 404, 1098 SM Amsterdam, The Netherlands (e-mail: gastel@science.uva.nl).

Theo Koreneef is with the Haagse Hogeschool, TH Rijswijk, Academie voor Engineering, Lange Kleiweg 80, 2288 GK Rijswijk, The Netherlands (e-mail: t.j.koreneef@hhs.nl).

Jos Loonen is with the Fontys Hogescholen, Institute for Information and Communication Technology, Rachelsmolen 1, P.O. Box 347, 5600 AH Eindhoven, The Netherlands (e-mail: j.loonen@fontys.nl).

Kees Uiterwijk is with the Hogeschool Utrecht, Institute for Information and Communication Technology, Oudenoord 340, 3513 EX Utrecht, The Netherlands (e-mail: kees.uiterwijk@hu.nl).

Manuscript received July 19, 2006. This work was supported by the Digitale Universiteit (DU) of the Netherlands, http://www.du.nl/.