Towards an agile infrastructure to provision devices, applications and networks: a service-oriented approach

Liu, Sandy; Spencer, Bruce; Lian, Y.; Xu, B.; Zhang, L.; Brooks, Martin

The First IEEE International Workshop on Requirements Engineering for Services (REFS'07)/The 31st Annual IEEE International Computer Software and Applications Conference [Proceedings], 2007

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Towards an Agile Infrastructure to provision Devices, Applications and Networks: A Service-Oriented Approach **

Liu, S., Spencer, B., Lian, Y., Xu, B., Zhang, L., Brooks, M.F.
July 2007

* published at The First IEEE International Workshop on Requirements Engineering for Services (REFS'07)/The 31st Annual IEEE International Computer Software and Applications Conference. Beijing, China. July 23-27, 2007. NRC 49324.

Canada

Towards an Agile Infrastructure to Provision Devices, Applications, and Networks: A Service-oriented Approach

Sandy Liu, Bruce Spencer, Yong Liang, Bo Xu, Libo Zhang, Martin Brooks
Institute for Information Technology, National Research Council Canada
{*FirstName.LastName*}*@nrc.gc.ca*

***Abstract*—Most industries and organizations use collections of tools, devices, and applications that are growing in complexity. New tools or applications may be acquired and old tools may become obsolete over time. They are often running on a variety of platforms, have different bandwidth and QoS requirements, and in most cases they cannot be accessed through a single point of entry. Moreover, some tools may require specific configurations done by technical experts. To address these issues, we propose an extensible, reliable, and simple software architecture that can hide the complexity of provisioning the network and running the tools. This paper introduces a service-oriented approach for creating an agile infrastructure to provision devices, applications, and their underlying networks.**

The Eucalyptus prototype is developed as an empirical application to test this approach. Eucalyptus is built on a set of generic fine-grained Web Services to manage and configure available resources, where new resources can be custom-built or imported from a third party. They can be integrated into Eucalyptus using a set of Web Service-enabled APIs. Our user community consists of architects and industrial designers. Eucalyptus can manage and configure the resources needed by geographically distributed groups of architects who need to collaborate in real time on the design of buildings, in a virtual Participatory Design Studio (PDS). Eucalyptus provides a single point of entry for the architects to access a wide variety of tools: videoconference applications, visualization services, rendering services employing parallel computers, etc. Eucalyptus provides a set of upper layer services for users to provision devices and applications running on high-speed broadband networks, as well as the commercial IP networks.

I. Introduction

Although the personal computer made its debut only decades ago, today, we find networks of computers running communicating software applications in almost every discipline and every aspect of our lives. Most industries and organizations use collections of tools, devices, and applications (which we refer to as resources in this paper) that are not supplied from one single vendor or they were built by different groups of developers over time. Some will become obsolete and new tools will be acquired. As a result, many resources are working in isolation - on different machines, platforms, or networks.

On the other hand, different resources require different network bandwidth and QoS. For example, a real-time application like VoIP, has little tolerance of any network interruptions and requires high priority for its packets; high-definition video conference applications require stable broadband connections. These resource demands have implications for the underlying network infrastructure. Currently, networking resources and configurations are mostly pre-configured by network engineers. As shown in the seven-layer OSI model [1], each layer can only communicate with the layer above or below. The application layer is at the top of the stack, typically it cannot talk to the network layer directly. But with Bill St. Arnaud's vision [2], the network should not be treated as a invariable component to the application; it can also be seen as a resource that provides different capabilities and topologies as services tailoring to the end user or application needs.

Beyond the demand for networks, many resources require custom configurations by technical experts. Since the users of these applications or tools are not necessarily IT professionals, they may not want to know the details of how to configure the tools and networks. However, as subject experts, they may want to control how these applications or tools are connected and how the data should be directed or shared amongst different geographically distributed groups.

Therefore, there is a need for a light-weight and agile infrastructure that can provide an extensible tool box allowing query, insertion or removal, and remote configuration of the resources, and can coordinate the usage of the resources in a configurable network. In this paper, we propose a Service-oriented approach for resource provisioning and management to fulfill these needs.

We employed this approach in developing a Participatory Design Studio (PDS) called Eucalyptus for architects and industrial designers. Architectural design is an advanced profession requiring collaboration of diverse teams exploiting a large set of powerful design, visualization, modeling, and deployment tools. This set of tools is not fixed; each may be running on a different platform requiring a different configuration. For example, the IBM Deep Computing Visualization runs only on Linux platform, and the Pleora uncompressed Standard Definition video-conference devices are accessible only through Windows machines.

Challenging factors such as design complexity, economic and environmental factors, new materials, and construction planning require the design team to access diverse and often distributed expertise. Architects who are away from their

design studio to do field work often need to communicate with what they have observed on site with their counterparts in the studio, and also need to use resources such as the rendering farm to conduct their work efficiently. Until now, insufficient bandwidth and crudely coordinated tools have resulted in distributed, task-based modes of collaboration, which often hinders the full participation by members of a distributed design team. As most architects are not IT experts, the motivation of Eucalyptus is to develop a single access point for architects to provision the resources required in a participatory design session without knowing the details of the configuration and the network.

Since some resources such as the UltraGrid [3] uncompressed high definition video-conference requires at least 880Mbps bandwidth, the corresponding network needs to be able to accommodate this much traffic volume. Therefore, the system needs to be able to switch to the suitable network topology accordingly. Thus we opt for the use of UCLP (User-Controlled Lightpath Provisioning)[4], and therefore it is a UCLP-enabled PDS (UCLP-PDS, a.k.a. Eucalyptus).

In the next section, we briefly introduce the underlying technologies, the Web Services based Service-oriented Architecture (SoA) and the UCLP tool. In section 3, we outline the overall design of our proposed solution; then we describe how we follow this approach to implement Eucalyptus. Finally we conclude with some issues we experienced, the contributions, and future work.

II. Enabling Technologies

A. Web Services based SoA

SoA is the latest software architecture style to built flexible and extensible applications. OASIS describes SoA as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [5]. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. Web Service's component-based, web-oriented, standard-based, language, platform, and domain independent nature makes it an appropriate solution for many data integration projects. We adopt this approach for provisioning resources spanning from networks to devices.

B. UCLP, Articulated Private Network and Hybrid Networks

With the exponential growth of the Internet and the increased cost of routing, the layer 3 (IP network) sometimes cannot provide the required bandwidth and stability required by certain applications. Many e-science projects involve the usage of remote sensors and instruments generating very large volumes of data that need to be delivered and processed in far away facilities. Architect design teams, our users, need to share high quality multimedia files in real-time, which calls for high transport capacity networks, often for a limited set of destinations. Advanced network organizations such as CANARIE (A non-profit organization who provides a national optical Internet research and education network in Canada. *http://www.canarie.ca/canet4/index.html*.) have been investigating and promoting ways to provide application-oriented and user-controlled networks services.

A resulting product is called the UCLP tool. UCLP [2] is a Web Services based solution for provisioning lightpaths. A *lightpath* is an abstraction of a connection between two or more switches in an optical network, and typically connects two points on the network at speeds up to 10 gigabits per second. UCLP can be thought of as a configuration and partition manager that exposes each lightpath in a physical network and each network element associated with a lightpath as an "object" or "service" that can be put under the control of different network users to create their own logical IP network topologies. The network users can then reconfigure and partition the lightpaths. This privately articulated end-to-end network is therefore called Articulated Private Network (APN). Within each APN, a number of network scenarios (i.e. logical topologies) can be specified to support different applications and usage scenarios. The APN Web Service can then be generated and deployed for users or applications to invoke it.

Geared with Web Services and UCLP technologies, we can operate a hybrid network that can offer end users both traditional services that runs on the IP network and services supported by the dedicated lightpaths. This can be achieved by setting up gateway computers that have access to both IP networks and lightpaths; or it can also be configured through corporate switches/routers. With hybrid networks, people can conveniently look up resources (including network resources) via the conventional Internet, and provision the resources (including the underlying networks) on both networks effectively through different Web Services.

III. An SoA for Resource Provisioning

A. Resource Wrapping

As stated, we consider resources to be any software applications, devices that can be controlled through computer programs, network elements, and the network itself. To make all resources accessible through Web Services, we developed a generic approach for resource wrapping. Note that we are only concerned about provisioning of the resource, not the actual data communication among resources. The basic tasks include launching, shutting down, or checking status of resources. For example, to start the multi-point video-conference application, we start the conference application with the proper parameters and configure the underlying network to make sure it can support the bandwidth requirement. The actual communication among different conference machines is handled by the native application. In order to generate the wrappers efficiently, we define each non-network resource with an XML description file. Figure 1 shows the schema for defining resources.

Each resource is assigned by a Resource ID and is described by a set of non-functional descriptions, including name, category (e.g. communication tools, visualization tools, etc.) , physical

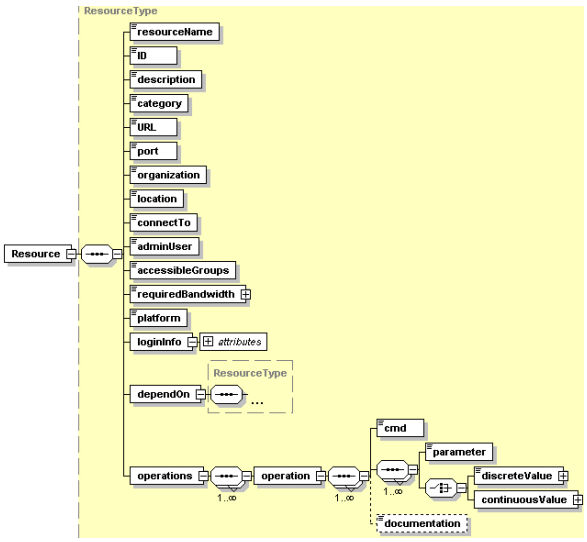

Fig. 1. Resource Schema

location, URL, port, the admin user, the access restrictions (which user group has access to this resource), what platform is this resource is running on (i.e. Windows, Linux), the login information for accessing the machine, which router or switch it is connected to, the bandwidth requirement, and any resources it depends upon. In addition, the XML file also specifies the operations supported by this resource; each operation is described by a command and the corresponding parameters. With this information, we can use the resource wrapping utility to quickly generate the corresponding Web Service for each resource.

As a result, for adding a new instance of an existing type of resource, all we need is do deploy the Web Service of the same type to the machine that hosts that resource.

B. Management Services

By wrapping the resource with our Resource Wrapper, we have the resources accessible for provisioning through Web Services. However, we also need a set of management Web Services to manage and coordinate the usage of the resource, such that we can use the Web Services as a control plane to all the available resources. This section will introduce the set of generic management Web Services. These management services form the basic building blocks of the infrastructure we have built.

1) Resource Management Web Service (RMWS): The RMWS provides services to add new resources and modify the properties of existing resources such as changing the IP address of a resource. The RMWS also acts as a registry, where one can look up available resources by its properties, and get the current status of different resources.

2) User Management Web Service (UMWS): The UMWS provides services to define user profiles, add new users, modify and delete existing users. It also keeps track of the contact information, login status, as well as the user groups.

Typically each resource specifies the access restriction. When a resource is being requested, the RMWS will contact the UMWS to verify if the user has the proper permission to use that resource.

3) Session Management Web Service (SMWS): A session is a collection of users and resources. A session starts when a user engages in using some resources. A user can potentially be involved in multiple sessions. For example, a user can participate in a session using a video-conference application with another two users while using a visualization resource for displaying some designs.

To reserve a session, a thread-safe check is made that each included resource is available to all included people. Then the session can be reserved. In more detail, a resource is available to a person if at least the following conditions are met: 1) The resource can be provided to that person via their own computer, or via another computer in the same room (location check is required); 2) The resource is permitted to be used by this user; 3) The resource is not otherwise allocated to other user through some other session, unless that resource can be part of two different sessions. Text messaging systems can be part of multiple sessions, as one can text messaging in more than one conversation at the same time, but two different video-conference sessions may not be hosted by one computer.

4) Workflow Management Web Service (WMWS): A workflow clearly defines the sequence of activities that must be performed in order to accomplish a certain task, and for each activity, it defines the preconditions required. In the context of Eucalyptus, activities are performed by Web Services. The Web Service workflow can be defined by a workflow language such as WS-BPEL[6]. Users can orchestrate a set of Web Services together to perform certain tasks. The workflow defined has to comply with the dependency relations associated with every resource participating in the workflow. We employ ActiveBPEL [7] as the runtime engine for workflows.

IV. Eucalyptus: An Empirical Application

Equipped with the Management Web Services, Eucalyptus incorporates the set of resources needed by architects into a uniform service-oriented system. This section outlines some implementation details of this system.

A. Overall System Design

All the core functions in Eucalyptus are provided by Web Services, either as a single service or a combination of services. We divide the services into two groups: task-oriented services and management services. As the name implies, task-oriented services offer the capability to conduct a task, such as submitting a rendering job to the rendering farm. We use the generic management services described in the previous section to provide support and management for the task-oriented services. For instance, the Resource Management WS is responsible for managing all the resources that are made available by Eucalyptus. Figure 2 illustrates the overall system design in Eucalyptus.

Note that we consider the APN setup Web Service as a task-oriented service. This service allows the Eucalyptus administrative user to set up different APNs, each with different configuration scenarios through the Web Services provided by UCLP. The Euclayptus end-user can later invoke an APN setup service or switching from one scenario to another within the same APN.

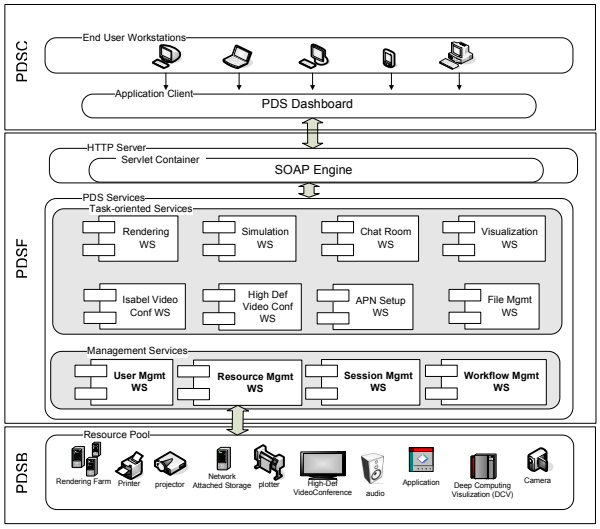

Fig. 2. Eucalyptus System Overview

We group the computers that make up our solution into three categories according to each role:

PDSF refers to **PDS Framework** computers. These computers have the Web Services platform installed and are generally used for exposing resources in Eucalyptus. A Web Services platform typically includes an HTTP server (*e.g. Apache*), a SOAP engine (*e.g. Axis*), and a Servlet container (*e.g. Tomcat*) that hosts the Web Services.

PDSC refers to **PDS Client** computers. These computers have the *PDS Dashboard* installed and are typically used by end users. The *PDS Dashboard* will be explained in the following section. In some cases, a PDSC computer can make certain local resources (*e.g. VNC*) available to the people in the session, but these resources are not exposed as Web Services on the local machines; instead, the access parameters are posted via the resource management service. For example, a PDSC can host a VNC server, and publish its access information (e.g. IP address) through the resource management service. Generally there is one active user using each PDSC computer.

PDSB refers to **PDS Backend** computers or devices such as the Deep Computing Visualization Server (an IBM BladeServer) and the Rendering Farm, which is a collection of high performance computers devoted to rendering. They are accessed via the PDSF computers and do not communicate directly with PDSC computers.

Note that one computer can play multiple roles. For example, a computer can run the Eucalyptus dashboard (PDSC) while hosting some Web Services (PDSF) for a few resources.

We categorize task-oriented services into different types according to the functionalities of the associated resources: communication tools, visualizing tools, management tools and so on (as shown in Figure 3). For example, Isabel (A multipoint videoconference for PCs, *http://www.agora-2000.com/*) is a kind of communication tool. This categorization allows us to retrieve available resources by type and also facilitates our implementation which will be mentioned in a later section.

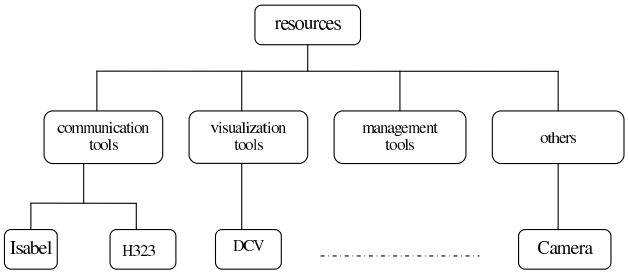

Fig. 3. Resource Category Definition

The resources in each category have similar functions, which means every resource in the same category has similar operations to be exposed to the user. For example, in communication tools, all resources should have following functions: `start(), stop(), getStatus()`. Different resources have different input and output parameters in their functions. To make the generic Web Service interface extensible to all of the more specific resource type, we declare all the input parameters and the return type as `String`. We will use parameter information described in XML resource description file to parse the input and output strings properly.

B. Customizable Service Client

To provide a single entry point to provision the resources, we provide a dashboard to provide a graphical user interface for users to effectively use resources and provision participatory design sessions. One of the goals of Eucalyptus is to provide easy access remotely to many high-end resources that are typically not available in most labs. Thus the dashboard hides the complexities of configuring the resources required, providing access to those resources through a few clicks of buttons.

In Eucalyptus, we decided to develop the integrated client as a desktop application as opposed to a web application for several reasons: 1) Web applications have limited functionality on the client computer. There is no easy way to access the local file systems. 2) The implementations of the HTML, CSS, DOM and some other tools are browser specific and they often act inconsistently in different browsers. 3) It is less convenient to maintain an accurate reflection of status of all the resources.

Fig. 4. The Eucalyptus Dashboard

To maintain a desktop application over many computers is normally not an easy task. However, with the help of Java Web Start [8], the deployment and maintenance of Java desktop applications become easier. The advantages of Java Web Start include automatic application update, desktop integration, platform independence, Java runtime environment management, and security.

The Dashboard interface is carefully designed to be unobtrusive and user-friendly. Inspired by DragThing [9], it is implemented as a floating dock, similar to the Mac OS X system dock. Each resource is represented as a graphical icon as shown in Fight 4, the dashboard (sometimes also referred to as FloatingDock) only appears at the bottom of the desktop, and it can be anchored to any other edge of the desktop.

Each resource has its own button on the dashboard. The user can define his/her own often used-resources on the dashboard and can add them as new buttons. Existing executable application can also be dragged and snapped into the dashboard, as a shortcut to launch that application.

C. Steps to Provision a Session

There are a few steps involved in setting up a session. The Web Service for each specific resource can only be invoked by the RMWS. If someone requests an Isabel session, the request will be first handled by the SMWS. The SMWS will contact the RMWS and the UMWS prior to the invocation of the Isabel video-conference WS to ensure that only authenticated and authorized user have access to it. The example shown in Figure 5 demonstrates how to start a four-site Isabel video-conference session. (1)The user selects the participants' sites and provide the input parameters through the Dashboard GUI. (2)(3)The SMWS asks UMWS to get the user's access permission information. (4)(5)The SMWS then makes the request to the RMWS, then the corresponding Isabel Web Services will be invoked in parallel. (6)Each Isabel computer returns its status indicating whether the resource is successfully reserved. (7)(8)The session status is returned to the user.

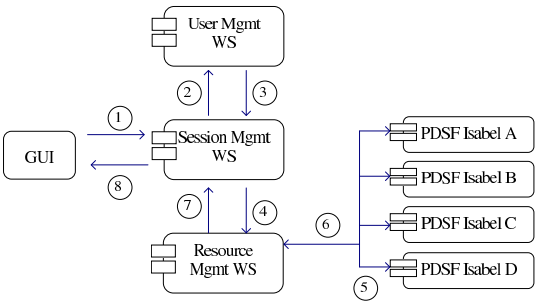

Fig. 5. Steps to Launch an Isabel Session

In order to correctly provision a resource, we need to properly detect its status. However, Web Services are typically stateless and passive. They only respond to service request initiated from the requestor, but not vice versa. One way to get the status of a particular resource is by creating Web Service that can return the status and having the client to poll such a service periodically. This approach introduces traffic to the service host and depending on the polling interval, it may not reflect the latest status of that resource. This is a common problem for Web Service projects.

In the current prototype, we introduce a lightweight TCP daemon, namely PDSDaemon, to detect changes and relate the changes back to clients who are interested in such changes. PDSDaemon is a TCP listener based on the MVC (Model-View-Controller) architecture. It listens to the client's request, forwards the request to management service and then sends the result back to all registered clients.

When a user logs in to Eucalyptus, the Floatingdock will send a connection request to the PDSDaemon. The PDSDaemon then creates a thread for the client who is requesting a connection. The resulting connection between the Floatingdock and the PDSDaemon will stay alive until the user logs out of the system. For each connected Floatingdock client, there is a separate thread that connects back to the daemon.

V. Results and Conclusion

We introduced an agile infrastructure that is built on Web Services and SoA. It provides the middleware to quickly make a resource available for provisioning through Web Services. It also provides an effective approach to put together a set of *ad hoc* tools into a single integrated service client for quick access and provisioning. We introduced an empirical application, Eucalyptus, and how to use this approach applied in the architectural design field.

In terms of development, we emphasize on a spiral approach, where we have many design, development, testing, deployment (in actual labs) and feedback (from authentic users) cycles. Each cycle we revisit the previous prototype and revise it according to the user feedback. This proved to be an efficient approach. The Eucalyptus prototype is useful in assisting architects to do collaborative design in distributed labs [10] [11]. Consequently, the school of architecture in Pennsylvania State University, and the Carleton University

(Canada) have adopted Eucalyptus in their collaborative course work for their architecture students.

Although the current prototype is applied for architectural design, using Web Services in provisioning is indeed applicable to many industries. St. Arnaud [12] advocates that this type of agile infrastructure will have real potential and significant impact in many other fields and disciplines. In summary, this service-oriented approach can serve as a basic building block for agile low-cost enterprize system without investing on expensive enterprise solutions. Our approach makes provisioning, running, and monitoring heterogeneous networks and network-enabled resources relatively easy and intuitive.

VI. Future Work

Our framework will be used to provision an increasing number of collaborative tools, some of which will have similar capabilities. Thus users may want the system to choose a tool set that is most appropriate at a given time for a given task. To support this, the tools and resources should be grouped into a class hierarchy to show the sets of available choices. Each class will be distinguished by a set of properties that distinguish its capabilities. This knowledge about classes and properties can be represented conveniently by description logic, where resource dependencies can also be outlined.

Semantic descriptions can also add more intelligence to the overall infrastructure. Currently, there is no semantic description for the network elements and the connections that are configured by UCLP, while the configuration and the topology at the layer 1 or 2 need to be understood by the application layer in order for it to take advantage of the articulated network. Ideally, the topology of the network is dictated by the user needs at the application level and the network can be re-configured on-demand if the current configuration does not fit. To fulfill this vision, the end-to-end network and its potential configurations need to be clearly described. UCLP takes care of the physical network up to the switches at the edge of the optical network, but the connection between the LAN and the optical switch are defined neither by the application nor by the UCLP tool. The scenarios defined by UCLP, therefore, have to be pre-defined and shared with the application in order to select the proper scenario at runtime. We plan to study ways to model the end-to-end network and give meaningful descriptions to each level to increase the efficiency of provisioning.

References

[1] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 4 1980.

[2] B. St.Arnaud, "CA*net4 research program update - UCLP roadmap: Web Services workflow for connecting research instruments and sensors to networks," http://www.canarie.ca, December 2004.

[3] the UltraGrid Project team, "UltraGrid: A High Definition Collaboratory," http://ultragrid.east.isi.edu/.

[4] The UCLP Development Team, "User Controlled Lightpaths," http://www.uclp.ca, 2006.

[5] The OASIS Service Oriented Architecture TC, "OASIS Reference Model for Service-Oriented Architecture," http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf, 2 2007.

[6] T. A. et al., "Business Process Execution Language for Web Services version 1.1," "ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf", 2003.

[7] ActiveBPEL, LLC, "ActiveBPEL, the Open Source BPEL Engine," http://www.activebpel.org.

[8] Sun MicroSystems, Inc., "Java Web Start Overview, White Paper," http://java.sun.com/developer/technicalArticles/WebServices/JWS_2/JWS_White_Paper.pdf, May 2005.

[9] J. Thomson, "DragThing," http://www.dragthing.com.

[10] M. Jemtrud, M. Brooks, B. Ho, S. Liu, P. Nguyen, J. Spence, and B. Spencer, "Eucalyptus: User controlled lightpath enabled participatory design studio," in *ACADIA(The Association for Computer-Aided Design in Architecture)International Conference 2006*, 10 2006.

[11] M. Jemtrud, P. Nguyen, B. Spencer, M. Brooks, S. Liu, Y. Liang, B. Xu, and L. Zhang, "Eucalyptus: Intelligent Infrastructure Enabled Participatory Design Studio," in *WSC '06: Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 2047–2054.

[12] B. St.Arnaud, "Cyber-infrastructure and grids for Architecture Collaborative Design," http://lists.canarie.ca/pipermail/news/2006/000362.html, 12 2006.