Collaborative Integration of Classic Applications in Virtual Reality Environments

Andreas Kopecki
High Performance Computing Center Stuttgart (HLRS)
Stuttgart, Germany

ABSTRACT

When working collaboratively with others, it is often difficult to bring existing applications into the collaboration process. In this paper, an approach is shown how to enable different applications to work collaboratively. It enables a user to do three things: First, the ability to work collaboratively with the application of choice, selecting those applications that fit the need of the scenario best, and the user is comfortable to employ. Second, the user can work in the environment he chooses, even if the application is not specifically designed for this environment like Virtual Reality Environments or mobile devices. Third, the technology presented makes it possible to mesh applications to gain new functionalities not found in the original applications by connecting those applications and making them interoperable. Taking a Virtual Reality Environment and a standard office application, the use and fitness of this approach is shown. It should be specifically noted that the work underlying this paper is not specifically on multimodal usage of Virtual Environments, although it is used that way here, but rather showing a concept of meshing application capabilities to implement "Meta-Applications" that offer functionality beyond their original design.

Keywords: Collaborative Engineering, Collaborative Work, Virtual Reality, Scientific Visualisation

1 INTRODUCTION

Traditional collaborative work that is state-of-the-art today focusses on screen sharing and video conferencing. Although this method works, as it is simple and does not imply a lot of demand on the systems used, it is awkward and uncomfortable. Modern collaboration demands more than just screen sharing. Especially in research and development the data is much more interesting than just the visual representation that you can capture using screen sharing alone.

The design of a new prototype in R&D often requires the consideration of a multitude of parameters that influence the final product. The product design impacts on the process in manufacturing and on the physical properties and vice versa, costs have to be calculated and reduced as much as possible and physical prototypes have to be somehow correlated to the initial virtual design. This often requires a multitude of data and views on that data, each usually bringing along its own application for display.

Thus, a design process involves several documents that contain different data that is somehow interrelated. An Excel sheet could contain the specifications for a virtual prototype that is visualised in a Virtual Environment, and a simple text file may specify the log and parameters for a simulation run whose results are displayed on top of the displayed prototype. Also, different views using the same application are often required to assess a development. This does not just include different viewpoints to a data set, but also different representations and different aspects. Sometimes, different tasks require different media for display or kinds of interaction, like making it necessary to relocate a discussion from a meeting table to a Virtual Environment for a closer discussion of an issue that has arisen.

Especially in aforementioned Virtual Reality Environments – but also on emerging every-day technologies like multi-touch screens – it is difficult to work with "classic" applications, as the interaction paradigms there are quite different from those used on the desktop. Here, the traditional screen sharing approach definitively fails, as it is impossible to translate the unique input restraints in those environments directly to the shared screen, thus limiting the collaboration to the same kind of the original device.

In this paper, an approach is introduced to make applications accessible in a collaboration, even beyond the boundaries of a single application. This technology will be used to access data from Microsoft Excel in the collaborative Virtual Reality Environment OpenCOVER for COVISE. COVISE is a modular and collaborative post-processing, simulation and visualisation framework enabling the analysis of complex data sets in engineering and science [1]. OpenCOVER, the COvise Virtual Environment Renderer first described in [2], supports Virtual Environments ranging from workbenches over Power Walls, curved screens up to full domes or CAVEs and head mounted displays. Using OpenCOVER, users can analyse their datasets intuitively in a fully immersive environment through state of the art visualisation techniques, including Volume Rendering and fast sphere rendering. Physical prototypes or experiments can be included into the analysis process through Augmented Reality techniques. OpenCOVER features an extensible plug-in framework that allows to add further functionality to the environment. OpenCOVER already supports collaboration at every level. Users can connect from different locations, analyse data sets, include audio and video conferencing in their session, mark and document collaboratively features of the data set, and other. Using COVISE and OpenCOVER, it is even possible to join a collaboration just using a plain web browser without any specialised plug-ins, enabling collaboration from everywhere [3].

2 Related Work

It is generally accepted that applications especially created with collaborative use in mind are less functional and used than the single user applications that are commonly used by the end users (e.g. [4] [5]).

A few methods exist that enable different users to share their applications with each other. There are the rare kind of applications especially designed for sharing. They incorporate concepts for collaborative work and inherent sharing. The advantage of that approach is that, as the sharing is done at the application level, it is usually much more sophisticated, efficient and powerful compared to all other approaches. Drawbacks here are that the shared application is usually developed primarily as a research vehicle for sharing concepts and often lacks the functionality typically found in state-of-the-art single user applications [5]. This is still true, though recent efforts of e.g. Microsoft [6] and others show that the once single user applications start to be extended to further support collaboration, integrating more and more into collaboration enabling Groupware systems. When looking at application design, collaboration features form usually a very small part of a complete application [7]. Thus the common collaboration research applications are doomed to fail comparing to state of the art applications. Collaboration design is nevertheless important, but it seems more straightforward – if the original software creator refuses to add internal collaboration facilities – not to replicate the application just for adding collaborative features, but rather to extend the existing application for collaborative functionality.

Different methods of transparent application sharing have been deployed. The first efforts undertaken were focused on image based sharing of the application. Here, the desktop screen or parts thereof is captured and transmitted to the partners in the session. This is currently the most accepted form of application sharing, used in various state-of-the-art commercial collaboration software like WebEx [8], TeamViewer [9], and others. The reason behind the wide adoptance is the ease to share arbitrary applications, nowadays usually without a dedicated application framework but through the web browser with appropriate standard plug-ins. The main drawback is generally a comparatively high bandwidth needed while sharing the application and that the application shared is not really collaborative, but rather just one user can work with a single application instance at a time, the others are viewers only. This limits the usability and flexibility of this approach, leading to other ways of transparently sharing an application.

In research, more semantic approaches are common nowadays. Those approaches use transparent sharing that enables to share applications without them explicitly knowing that they are being shared. In contrast to the screen sharing approach, they usually only capture raw input events like key presses or mouse clicks rather than pixel data and transform those into more abstract events using knowledge about the application. The clicks and inputs are then replayed in all the participants' locally running applications and thus keep the application state consistent on all connected hosts. Systems using this approach are described e.g. in [5] and [4].

The main advantage of this approach is that it is feasible for all current applications. But it requires the applications to be visually equivalent to capture the right button at the right coordinates. If one user reconfigures his user interface, this method may fail completely. It is also usually limited to a single operating system type as it accesses low level operating system features not found on other systems. Furthermore, only the surface of collaboration is touched this way. Capturing the application state and the content currently processed in the application cannot be included in the synchronisation process, although it is typically the most important part in the collaborative work.

A more sophisticated system will try to access the internals of an application and use the loaded documents and application states to improve the collaborative experience [7]. Almost all state-of-the-art commercial applications, especially on Microsoft Windows, nowadays support some kind of API a programmer can connect to and issue commands to the application or retrieve information about the document loaded or the state the application is in. This kind of API makes it often very easy to implement a collaborative layer, quickly enabling an application to be used in a collaborative way.

In the last decade, several such solutions emerged that did not simply broadcast screen contents and capture keyboard input, but rather analyse the input events and the application semantically and replicate the events and synchronise the application contents on remote machines. This approach is surprisingly successful, as several examples show, extending off-the-shelf software like Word [7], Visual Studio [10], or Maya [11]. Xia e.al. [7] for example use Operational Transformation to map events from an application to simple insert and delete operations on a linear object space. As those operations are quite simple, it is also relatively easy to assert the consistency of the two documents when edits change cursor positions in the document. Thus, no direct data is exchanged, but the events occurring in one application are analysed, packaged into an abstract, simpler representation and replayed in the target application. This is currently done between the same applications though, forfeiting much of the power of this approach, but this is maybe owned to the linear operational space and the simple operations.

3 Objectives

Many sophisticated applications are available that offer a plethora of functionality. Usually, this functionality is limited to a certain environment – usually the desktop – and is available to a single user only. Users should be enabled to choose the application and the environment they desire and collaborate freely with others nonetheless, even if the other partners prefer other applications or environments. Thus participants should be enabled to work in Virtual Environment if desired or at their local desktop, or on mobile devices. It is also desirable to switch between those environments and let others participate in what they are working on.

In Section 2 several examples for coupling two running instances of an application or two applications of the same type were introduced. But while enabling collaboration between different partners, a goal is also to bring together different applications to offer a unique functionality not available out of the box, harnessing the power of a specialised application and connecting it to offer an advanced functionality via a common interface. This separates this approach from others available that allow the collaborative, real-time working on documents like Microsoft Office Live, Google Docs and is even more powerful than the simple screen sharing approach used by WebEx or TeamViewer.

Both, the sharing and coupling of those applications, should be done transparently without the need of explicitly informing the application that it is used in a way thus. Also, the set-up of the functionality should be both flexible and easy to learn, using state-of-the-art technologies available on most platforms.

4 Design

The implementation of the collaborative and inter-application capabilities is designed around a central component – the Application Controller – that is running on every participants' computer. It can be used for starting and stopping the applications as well as – more important – to steer the application and distribute and transform events originating from the controlled applications to other participating applications.

Application Controller

For transparently sharing applications, a component is needed that captures application events, analyses their contents, transforms them appropriately and distributes them to all applications interested in this kind of event. Thus, an Application Controller was developed that is capable of taking control of an application and steer it with a small subset of commands. Using the Application Controller it is also possible to save the application state of the controlled application to storage and to resume the operation later on with the same application state as before.

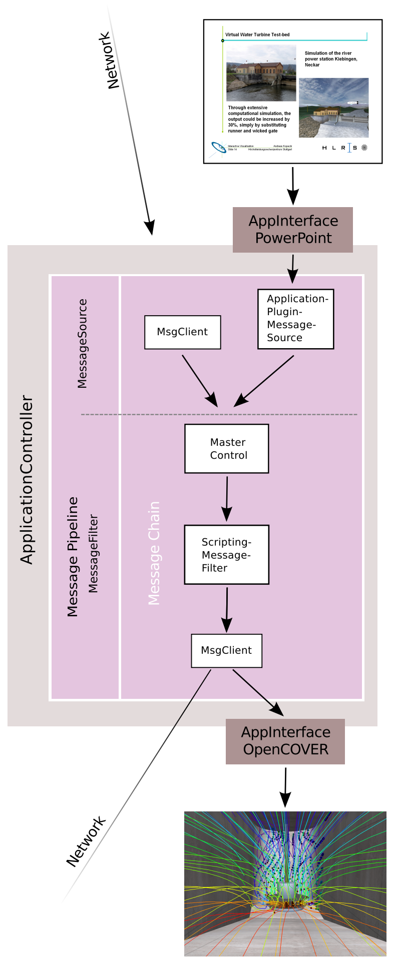
Network
Virtual Water Turbine Test-bed
Simulation of the river power station Kiebingen, Neckar
Through extensive computational simulation, the output could be increased by 30%, simply by substituting runner and wicked gate
Interactive Visualisation
Slide 14
Andreas Kopecki
Höchstleistungsrechenzentrum Stuttgart
H L R S
AppInterface PowerPoint
ApplicationController
MessageSource
MsgClient
Application-Plugin-Message-Source
Master Control
Message Pipeline
MessageFilter
Message Chain
Scripting-Message-Filter
MsgClient
AppInterface OpenCOVER
Network

Figure 1: The message pipeline of the Application Controller

The Application Controller is a small networked service application that is running at every host participating in a session. Every application that is integrated into a session will require a specialised application interface for communicating with the Application Controller. These interfaces are realised as plug-ins, linking them at run-time directly to the Application Controller. The application interface is responsible for translating the commands originating from other applications to mechanisms native to the application, like COM, CORBA, WebServices, signals, and others. It also gathers the feedback and events from the application and sends them back to the Application Controller. Thus the network of the Application Controllers creates a hub that connects all applications in a session, relieving them of the necessity to directly communicate with a common protocol.

The Application Controller itself is listening at a SOAP interface [12], waiting for commands to execute or sending messages to other running instances. Applications are instantiated using this interface and taken control of by the Application Controller. It creates the new application instance by loading the appropriate application plug-in and starting the application process. A communication link is established to the native API of the application.

The capabilities – i.e. the commands an application accepts – are published using a simple XML document. This descriptive document also contains other information about the application like what documents it is able to process, if it can load directly from a certain source (e.g. via the http protocol), and more. Clients to the Application Controller can use this information to create generic user interfaces to steer every application supported by the application controller out of the box. All Application Controllers are connected via a network bus for collaboration beyond host boundaries.

Messages

Generally, the design is based on a message passing, message transformation and state preservation mechanism. Messages are usually coming in from the network or generated by the applications with the help of their application plug-ins connecting them to the Application Controller. Thus, whenever e.g. a PowerPoint slide set is loaded, an event is generated in PowerPoint that is captured by the Application Controller plug-in for PowerPoint. It generates a message, indicating that a file was loaded. This message is sent to all other components that are subscribed to these events. In Fig. 1 the message is transformed and passed on to a Virtual Reality Environment, loading a data set corresponding to the slide displayed in PowerPoint. Messages are realised as strings encoding the message body. Arbitrary strings can be sent in the message body, but per convention, an XML format is used for easy parsing and structuring of the message content. But of course, other formats can also be used that are understood by the message destination, like JSON for ECMA-Scripts or Base64 encoded binary data for other destinations. Choosing a string representation has several advantages, like human-readability, non-endianess, same bit representation, and ease of use, but of course limits the amount of data sent between applications. While it still may be feasible to distribute a text document, the overhead for large data like in numeric simulations is just too much and has to be done using different communication channels.

Message Filters

Applications could react directly to those events, but that would be quite inflexible, only allowing a pre-defined behaviour that also would be hard to switch off if not desired. Thus, a more flexible approach is used. All messages are sent down a message pipeline that consists of several filters that process the message on its way to the receivers. Filters can be arranged sequentially in the pipeline, as well as in parallel, allowing the aggregation of results from several independent filters. This aggregation allows the flexible combination of functionality, for example to prevent Master/Slave control from having an impact on network communications. The filters are able to modify messages, add new messages, replace the message with one or more, or discard the message entirely. As messages sent from and to applications are basically strings containing information possibly interesting for others, it is very simple to parse and react on them even from different frameworks like scripting languages or other runtime environments using their own byte-code like Microsoft CLR or Oracle JVM. Messages also contain some extra fields providing information like who generated the messages and a topic describing a message group. Other applications or message filters can subscribe to a topic or subscribe to all topics and filter messages themselves for topics of interest.

Message filters are realised as plug-ins and dynamically loaded by the Application Controller. Message filters can have arbitrary functionality, but are usually used to modify or block messages or send them to other Application Controllers or components partaking in a collaboration. Filters exist for communication, transformation, master/slave floor control, collecting user data, and others. Currently, the most used filters in the Application Controller framework are a network component sending all messages to a message bus for other Application Controllers to react on them, thus enabling collaboration between different users or allow application meshing between applications running on different hosts, and a scripted transformation filter that reacts on events using a scripting language. The advantage of a scripting language for filtering is its flexibility and fitness for quick rapid prototyping cycles. For scripting, QtScript was chosen. QtScript is a subset of the ECMA standard that is also the base for more common languages like JavaScript and should be easy enough to learn by anyone who is already familiar with languages like C++, C# or Java. Scripts can also access all Application Controller functionality by calling its native methods. Thus, scripts can send arbitrary commands to application plug-ins to retrieve further data to be processed.

A script can be easily loaded by directly distributing it amongst the Application Controllers issuing a command or by passing a file name to all Application Controllers to load the script from. This creates what we call a workspace, connecting different applications together, enabling meshed functionality and collaborative features.

5 Application of the Concept

The concept of collaboratively using off the shelf applications in different environments was used in several scenarios ranging from contextual support to complete workspaces for meeting support. To exemplify this technique, a small evaluation scenario was chosen for this paper to demonstrate the possibilities of collaboratively using applications and meshing two applications within a Virtual Reality environment. It consists of a small real life problem in turbo machinery development. Here,

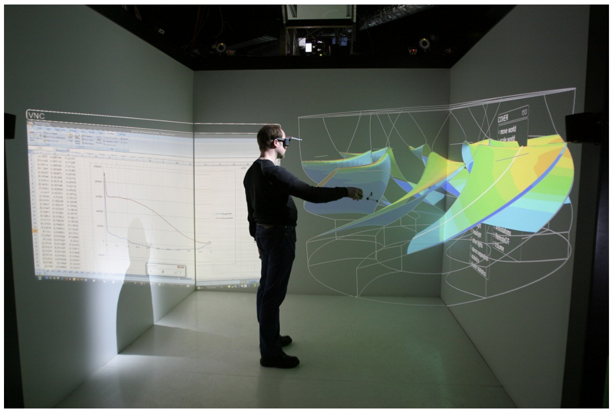
Figure 2: Using Excel data in a Virtual Environment

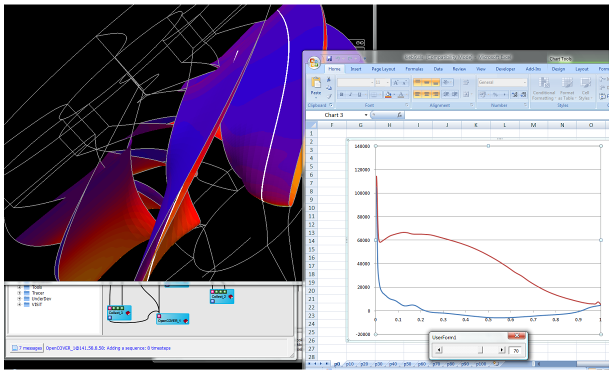
Figure 3: Using the same set-up as in Fig. 2 collaboratively at the desktop

while designing turbines for water power stations, the performance data is usually assessed using two tools: A 3D visualisation of the turbine with various parameters mapped on the turbine blades in the Virtual Reality Environment OpenCOVER and a two-dimensional curve of the pressure profile in Microsoft Excel. Those two views are difficult to merge: The pressure profiles are selected using a percentage of the boundaries of the turbine that is not easily mapped to the real turbine in 3D.

Thus it is difficult to correlate these views that reside in two different applications. Therefore, a method was needed that was able to show the location of the pressure profile in the Virtual Reality Environment to make it possible to establish this correlation. Admittedly, it is possible to somehow display those curves by exporting them from Excel and writing a reader module for the Virtual Reality Environment for visualisation. But it is obviously more desirable to have the data within its original application and share it with others in a transparent way – and to do this using the applications as they are without adding direct functionality. If changes are made, they can be directly applied to the Excel table and will be immediately visible in the Virtual Environment for further analysis without any explicit conversion steps. Using the Application Controller, this also allows to collaboratively share and discuss the data from within several environments.

The applications were connected using the Application Controller and a small script reacting on changes within Excel and sending the data to the Virtual Reality Environment. In the set-up described here, three computers are involved. One head node driving the cluster for the Virtual Environment running OpenCOVER in Linux (Fig. 2), one Windows Workstation running Microsoft Excel and a TightVNC server [13], and one Notebook of a remote partner running a desktop version of OpenCOVER and another instance of Excel (Fig. 3). When the script for the workspace is loaded, it automatically loads the Excel file corresponding to the turbine currently displayed in the Virtual Reality Environment. It also causes OpenCOVER to connect to the workstation running Excel using VNC, displaying Excel directly within the Virtual Environment and allowing a limited interaction with Excel and the data loaded. On the remote notebook, both, Excel and OpenCOVER are started, the appropriate data loaded and displayed. Both users – the one in the Virtual Environment and the other one at the remote notebook – can now analyse and discuss the data, select different pressure profiles via a slider in Excel and compare the pressure profiles to the three-dimensional visualisation of the turbine blade.

When the slider is moved, the chart displaying the pressure profile changes and an event is sent to all participants. The event arrives at the script that consecutively queries Excel for the location of the pressure profile on the blade. The coordinates are stored within the Excel sheet and thus can be accessed by the Application Controller script. The coordinates are transformed to the correct format and sent to OpenCOVER that can use the x-/y-/z-coordinates to display a poly-line on top of the turbine. The user in the Virtual Environment can now compare the pressure profile with the data mapped on the 3D visualisation with ease. The other user sitting at his desktop can also compare the same results concurrently.

A small evaluation was done with engineers doing turbine assessment in their daily work. They were impressed by the ease they could now compare the pressure profiles and the three-dimensional visualisation. All used both views for assessment while working in the virtual environment. None of the users neglected one view in favour of the other. This shows that meshing applications allows for a wider range of functionality than that available from a single application that is unable to cover everything, but is usually focused at one field it excels. Using the internal API of an application while meshing them gives a lot more possibilities in combining views, data and control. The shown example is very basic in its nature, but shows that just using the off-the-shelf applications it is possible to get results that were not able to be achieved before.

6 Conclusions

In this paper, a technique was demonstrated that not only allows to make applications collaborative in a way transparent for them, but also allows to mesh those applications to add functionality to other applications in use. It was not necessary to change those applications for the functionality implemented. In the given example, we chose an Virtual Environment application and a standard Office application for showing the application meshing concept. The approach is not directly targeted or limited to this example though, but rather a generic coupling method of two or more distinct applications. The coupling is done by generic or specialised filters that transform application events into new behaviours and commands. The flexible approach chosen using a directed message filtering graph allows to combine different filter functionality with ease.

This example is still not "feature complete" of what you may have in mind what can be achieved by coupling of data sources and applications. Of course, when adding further functionality to the applications in question, a more seamless interaction concept is possible. E.g. the Virtual Reality Environment OpenCOVER could be extended by a plug-in that better visualises the location of the pressure profile and even maps the profile directly onto the turbine blade. Also, the slider to steer the Excel table could be integrated as an interaction concept into the virtual environment rather than using the original Excel slider. This would have required a dedicated component in one of the applications, a thing that to do was avoided in this paper to show the possibilities of the method without extending the original application and with relatively minimal effort in scenario programming.

References

[1] A. Wierse, U. Lang, and R. Rhle, "A system architecture for data-oriented visualization," in *Proceedings of the IEEE Visualization '93 Workshop on Database Issues for Data Visualization*. London, UK: Springer-Verlag, 1993, pp. 148–159. [Online]. Available: http://portal.acm.org/citation.cfm?id=646122.680437

[2] D. Rantzau and U. Lang, "A scalable virtual environment for large scale scientfic data analysis," in *Proceedings of the Euro-VR Mini Conference*, 1998.

[3] F. Niebling, A. Kopecki, and M. Becker, "Collaborative steering and post-processing of simulations on HPC resources: Everyone, anytime, anywhere," in *Proceedings of the 15th International Conference on Web 3D Technology*, ser. Web3D '10. New York, NY, USA: ACM, 2010, pp. 101–108. [Online]. Available: http://doi.acm.org/10.1145/1836049.1836065

[4] D. Li and R. Li, "Transparent sharing and interoperation of heterogeneous single-user applications," in *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*. New York, NY, USA: ACM, 2002, pp. 246–255.

[5] B. Xu, Q. Gao, and C. Li, "Reusing single-user applications to create collaborative multi-member applications," *Adv. Eng. Softw.*, vol. 40, no. 8, pp. 618–622, 2009.

[6] Microsoft Corporation. Microsoft Sharepoint Workspace web page. http://office.microsoft.com/sharepoint-workspace/. Accessed 2011. [Online]. Available: http://office.microsoft.com/sharepoint-workspace/

[7] S. Xia, D. Sun, C. Sun, D. Chen, and H. Shen, "Leveraging single-user applications for multi-user collaboration: the coword approach," in *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*. New York, NY, USA: ACM, 2004, pp. 162–171.

[8] Cisco. WebEx home page. http://www.webex.com/. Accessed 2011. [Online]. Available: http://www.webex.com/

[9] TeamViewer GmbH. TeamViewer. http://www.teamviewer.com/. Accessed 2011. [Online]. Available: http://www.teamviewer.com/

[10] J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski, "Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development," in *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2008, pp. 939–948.

[11] A. Agustina, F. Liu, S. Xia, H. Shen, and C. Sun, "CoMaya: incorporating advanced collaboration capabilities into 3d digital media design tools," in *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*. New York, NY, USA: ACM, 2008, pp. 5–8.

[12] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. SOAP version 1.2 part 1: Messaging framework (second edition). Accessed 2010. [Online]. Available: http://www.w3.org/TR/soap12-part1/

[13] TightVNC Group. TightVNC: VNC-compatible free remote control / remote desktop software. Accessed 2010. [Online]. Available: http://www.tightvnc.com/