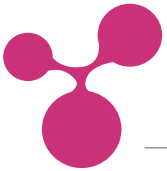
Technische Universität Dresden – Fakultät Informatik
Professur für Multimediatechnik, Privat-Dozentur für Angewandte Informatik

Prof. Dr.-Ing. Klaus Meißner
PD Dr.-Ing. habil. Martin Engelien
(Hrsg.)



GeNeMe '07

Gemeinschaften in Neuen Medien

an der
Fakultät Informatik der Technischen Universität Dresden

Unter Mitwirkung der
Comarch Software AG, Dresden und der
GI-Regionalgruppe Dresden

am 01. und 02. Oktober 2007 in Dresden
http://www-mmt.inf.tu-dresden.de/geneme/
geneme@mail-mmt.inf.tu-dresden.de

A.13 Instant Collaborative Web-Browsing with VCS

Stefan Pietschmann[1], Matthias Niederhausen[1], Tobias Ruch[2], Roman Wilkowski[2], Johannes Richter[2]
[1]Technische Universität Dresden, Lehrstuhl für Multimediatechnik
[2]Comarch Software AG

1. Introduction

Web2.0 is currently changing the Internet by introducing a user-centric structure, social networks and highly interactive content. However, it does not yet provide the means for "real" collaboration on the Web, i.e., synchronous interaction among people. Looking at flickr[1] photos together with friends you met on holiday although they are from the other end of the world, or shopping for Mom's birthday with all your siblings at Amazon[2] is not possible on today's WWW.

In real life, people are embedded in families and work teams, i.e., they spend most of their time in social environments/communities. The majority of people enjoy and prefer working and spending time together with others. Moreover, it is proven that communities and groups are generally more efficient in problem-solving [Shaw, 1932].

Since its beginning, the purpose of the Internet has mainly been information exchange. Web browsing meant someone sitting alone in front of his computer searching for web content of his very own interest. With the emergence of Web2.0 users are no longer only information consumers but also information providers, thus becoming so-called "prosumers". The role of the WWW has now changed from an encyclopaedic information store to a space of entertainment, social interaction, communication and collaboration. Yet, while the users' power to shape the virtual world they act in has increased significantly, web browsing has not changed at all. People are still sitting alone in front of their computers, interacting asynchronously via forums, mailing lists or blogs. The need for actual real-time collaboration becomes evident when we look at the large number of parallel communication networks that have been established: instant messaging, voice chat and many more are the result of peoples' wishes to share their online experience.

The central question that comes up is: If so many people are browsing the WWW, why can't we see them? If social interaction plays such an important role in our lives, why do we limit the browsing experience to sole information exchange and separate it from communication networks? At this point the concept of collaborative browsing comes

[1] http://www.flickr.com
[2] http://www.amazon.com

into play. There are already a number of systems that allow people to share applications running on their computer, e.g., web conferencing and remote desktop solutions (cf. chapter 4). However, none of them truly addresses the problem of transporting real social interaction protocols into the context of a web browsing application.

In this paper, we introduce VCS (Virtual Consulting Services), a new system for collaborative browsing. The research project around VCS aims at enabling unrestricted end-user collaboration on already existing web-based content. As a service platform, VCS bridges the gap between social networking in real life and the online world.

This paper is structured as follows. The next chapter illustrates how co-browsing can extend common web scenarios as well as offer completely new fields of application. Chapter 3 describes the co-browsing system VCS in detail, gives an architectural overview, provides information about the browsing synchronization and explains the features of context-aware co-browsing. Chapter 4 discusses the related work. Finally, chapter 5 draws a conclusion and gives an outlook on the future work in this field.

2. New Fields of Application

Due to the reasons described in the previous chapter, co-browsing is a promising and exciting approach towards a more social and useful Internet. It can be applied in various online scenarios, such as co-shopping, consulting and online communities. In the following we will only explain the benefits of co-browsing to these three scenarios, although many more possible fields of use exist, like e-learning on the open Web or guidance for on-site supporters through virtual manuals from the main office.

2.1 Co-Shopping

While shopping in the Web is becoming a common way of buying goods, the classic way of shopping still has one decisive advantage over e-commerce: consumer goods are often bought together with friends or family and thereby shopping comprises a social component. Though e-shopping is convenient, takes less time and is often cheaper, going to the mall is a social event which cannot be reproduced by web shops yet. Another advantage of classic stores is the shopping assistant. If you are looking for a present, want to know if a product is available and where it is located, or simply need a recommendation, the assistant can help and guide you through the store. Proof for the importance of this fact is given by the recent study "E-Shopping-Trend 2006"[3], which concludes that German e-businesses lose 9 Billion Euros every year because customers miss helpful advice and feel left alone. By addressing this deficiency in web-based shops, Internet businesses can raise their income and attractiveness. Further, by using

[3] http://www.novomind.de/index_ht.html?press/2006/rel_117.html

collaboration features, a web shop becomes more distinguishable compared to other shops and attracts new customers [Manhart et al., 1998]. As [Farnham et al., 2000] state, people generally prefer shared browsing to single browsing when shopping. Even if this "social service" offered by web shops does not directly result in increased sales, it still can contribute much to the loyalty of customers towards a shop [Cyr et al., 2006]. While it is clear that co-browsing can enhance online shopping in general, the application of co-browsing to web shops can be further refined into three use cases.

First, co-browsing can be used to transfer the idea of real-world collaborative shopping to the Web. Friends can meet on a specific website at a certain time, as they would do in real life. This can be an additional feature of a web shop and allows friends to amble around the shop, showing each other details and maybe discussing about a present for a common friend. Additionally, friends do not have to be at the same place for this "social event". Consequently, co-browsing combines the advantages of both the real world and the Web: shopping together regardless of users' location.

Second, co-browsing can be used to advise customers during their shopping session. Instead of visiting a web shop with a friend, a trained assistant can be called and – like in every "offline shop" – answer questions and give advice. This kind of usage especially addresses the problems discussed in the study mentioned above. Details on how consulting in general can be accomplished with the help of co-browsing are given in chapter 2.2.

A third possible employment of co-shopping is a mixture of the previous two use cases and is closely related to Web2.0 applications like collaborative social networking and communities. Instead of taking one's friends along, one can co-browse with other customers, help each other, discuss about products and thereby form a so-called "shopping community". Co-browsing just by coincidence could be allowed between customers of a web shop, comparable to random encounters in a mall. As users already share their thoughts and knowledge in blogs or wikis, co-browsing would allow them to directly exchange their experiences on different products or to give ideas for presents.

2.2 Consulting

In the field of customer support, co-browsing can become an excellent and flexible alternative to the common helpdesk hotline. For companies offering complex online transactions like booking a flight or changing the modalities of an insurance contract, proper support is a crucial selling point. However, call centres, the common way to address this problem, are cost-intensive. Furthermore, they are inefficient in that they simply lack a shared view between customer and operator. It is quite common that a customer calling a help desk lacks the knowledge to explain what his problem actually

is. Instead, he just wants to show it. Then again, the operator has to navigate customers without seeing what they actually do. Due to these problems, more and more companies try to establish self-care portals to allow customers to solve their problems themselves. Thereby, the responsibility is shifted towards the customers, who are left alone with systems that are often hard to understand and demand a level of both domain knowledge and confidence in dealing with web-based services. Consequently, customers again find themselves using a phone hotline to get help in a system which is supposed to supersede the very hotline they use.

With co-browsing techniques, support can be provided directly on the Web. Operator and customer share the same view and act on the same page. In addition to an easier and by far more convenient way of handling support requests, customers gain more confidence which finally leads to lower numbers of repeated help requests. Co-browsing must, however, include new domain-specific features that closely resemble real-life interactions. As an example for a consulting scenario we mention the “pencil metaphor”. It allows consultants to fill in web forms with a “pencil”, though they do not have the right to submit it. Customers see the changes written in “pencil” style and can erase, edit or confirm these suggestions. Like in real life, it is easy to get help with filling out a form without losing the freedom to decide what and when to submit it.

All given arguments about facilitating co-browsing for consulting purposes are especially valid for e-government applications. Public authorities provide online services for registration and tax declaration to achieve higher efficiency, faster processing of requests and lower costs. By moving services to the Web, long waiting queues and the workload of civil servants in government agencies could be reduced. However, because of the legally required accessibility to government resources, direct support is even more critical than for companies as mentioned above. Co-browsing techniques can be utilized here to improve accessibility.

2.3 Communities

Online communities are an essential part of Web2.0. Within certain fields of interest, millions of people share their creativity, experiences and ideas. Collaboration is generally asynchronous, e.g., in the form of discussion boards, mailing lists, blogs, tags and comments, while the means for synchronous web-based interaction are still limited and include text chat and other proprietary communication solutions.

The co-browsing system presented in this paper allows for much richer interaction principles available to online communities. People can actually see each other interacting with the shared content of interest. Imagine a co-browsing-enriched version

of social networking portal like deviantart[4] or the German StudiVZ[5]. Members can browse collaboratively through their profiles and pictures, discuss their last holiday, exchange news via voice chat or help each other finding friends from school. Thus, online communities become places to actually spend time together. Co-browsing even provides the means for online games. The simplest websites without any game logic can serve as a place to meet and play, since players can collaborate just like in real life on a board or paper. By easily providing new activities, such websites can keep their users online for a longer time, thus raising advertising revenues, while being even more attractive to users.

3. The VCS Co-Browsing System

VCS (Virtual Consulting Services) is a thin-client co-browsing tool. To support the variety of application fields mentioned before, it strives to enrich, not to replace web browsing as we know it. Therefore, it employs an event-based (i.e., transferring information on what the user did) instead of an image-based (i.e., transferring screenshots at a given interval) synchronization of shared web content. Using this approach has one major advantage: latency is much lower, thus bringing possibilities for web collaboration to a whole new level. Only this way is it possible to support real-time synchronization of interactions on the shared content and to provide a high degree of awareness.

The main advantage of its thin-client approach is that VCS does not require the user to install any additional software on his PC, be it a stand-alone application or a plugin. Instead, VCS incorporates existing browser capabilities like JavaScript to achieve its functionality. Leveraging the browser as environment allows the user to act within a familiar application and also frees the co-browsing system of the need to render content on its own. As a result, VCS presents itself as a user-friendly, installation-free application in times when users are wary of spyware and viruses all over the Web, not creating any barriers towards the new technology of collaborative browsing.

It is clear that, by choosing thin-client technologies, capabilities like audio or video conferences can not be directly implemented. Therefore, we provide easy and automatic integration of already installed third-party software like Skype or Adobe Flash.

The next chapters will go into detail on architecture, synchronization techniques and context-awareness of our system.

[4] http://www.deviantart.com
[5] http://www.studivz.net/

3.1 Architectural Overview

VCS is designed as a modular, proxy-based architecture. On the server-side, it acts as a mediator between co-browsing users and content providers (Figure 1), extending the common client-server architecture. This is done by synchronizing collaborative activities between the participating clients and representing a single client towards the external content server.

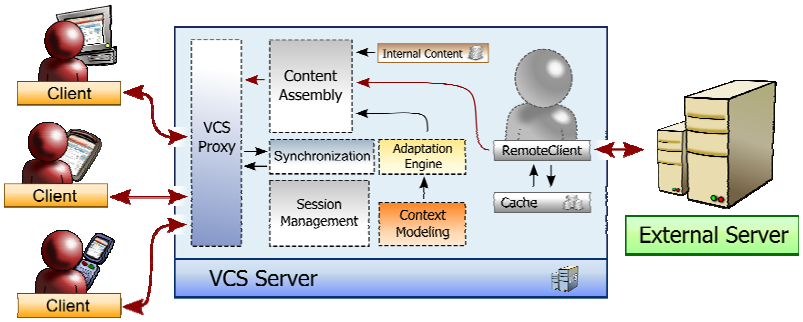


Figure 1: VCS architecture

The VCS server itself is designed in a modular fashion. As a central component, the *VCS Proxy* is responsible for processing incoming client requests. HTTP requests are relayed to the *Remote Client* which retrieves the data from the external server or its own cache. To ensure that all requests are routed via the *VCS Server*, a *Content Assembly component* modifies delivered web pages on-the-fly. Every hyperlink is rewritten to point to the VCS server. Further, a set of client-side components are embedded into every page. The *Session Management* and *Synchronization* components are responsible for synchronizing the state of different participants and managing all concurrent sessions running on the server. They also take care of the rights management which is especially important for applications like customer consulting. Finally, the *Adaptation Engine* tailors web pages to specific needs and capabilities of the user group within a session as well as their single members. Further details on the *Adaptation Engine* can be found in chapter 3.3.

As stated previously, the client application is implemented in a thin manner, using only JavaScript. It can be classified a parasite [Marais & Bharat, 1997], as it is embedded into the requested web pages and can control the behaviour of the web browser as well as monitor user interactions. Since it is used in heterogeneous browser environments, it lets the client choose the communication technology (cf. chapter 3.2).

VCS also offers support for specialized application fields (cf. chapter 2) and heterogeneous devices. To this end, a domain-dependent, component-based GUI is utilized to provide the user with an appropriate interface.

3.2 Browsing Synchronization

A crucial demand to co-browsing applications is the proper real-time synchronization of shared content and interactions of participating users. A common answer to this demand is screen sharing, in which the entire view of a client is transferred to other participants on a pixel-by-pixel basis. This ensures that, regardless of the content, all shared views look the same. However, those solutions lack important features needed to establish real synchronous collaborative browsing. Today's client devices greatly vary in their capabilities. Even just a different screen resolution can ruin the co-browsing experience if one participant constantly has to scroll in his view because his device only supports half the resolution another participant uses. When it comes to more heterogeneous situations, e.g., participants with mobile devices or visually impaired users, image-based applications fail to provide the necessary means for effective co-browsing. That is why VCS realizes an event-based approach for content synchronization, which is now discussed in more detail.

As its content our system uses common web documents provided in (X)HTML. All session members browse on the same document and can see each other. Browser events, such as *MouseMove*, *KeyDown*, etc. are transparently caught by the client-side script components. Using an open, text-based protocol, these events are propagated to the server and from there distributed among the other clients in the session.

The client-server connection is established through an Ajax channel, a socket connection held by a Java-Applet or a Flash container. Events from other clients received via the server are processed according to the co-browsing interaction protocol. This protocol specifies what actions are triggered by certain events. Moving the local mouse pointer over a certain page element will, for example, result in a semantically equivalent representation in the browser of a remote participant. Other user interactions like changing the content of form elements, e.g., text fields or selection boxes, are synchronized as well.

The flexibility of event-based synchronization comes at the cost of new problems which have to be addressed carefully. One difficulty is to ensure the consistency of shared content. A participant joining a co-browsing session has to receive not only the page currently shared but also all changes which have been applied during the session. For example, form fields have to be synchronized (i.e., filled with contents) and page modifications (made by scripts or participants) have to be reproduced. This task is accomplished on the server side within the *RemoteClient* which keeps track of the latest state of a form element and delivers that state whenever a new client enters the session. Whenever synchronization messages are lost, e.g., due to network failure, clients can request content synchronization, so that consistency of the shared content is ensured.

A more complex problem is non-deterministic content. Some web application providers deliver random or user-tailored images like advertisements; web shops show random offers of the day varying from page request to page request. Thus, co-browsing users might end up seeing different things on the same page – a situation which has to be avoided under all circumstances if users are to talk about the same subject. The VCS architecture addresses this problem by caching a requested page and delivering this page to all clients of the same session who request it afterwards.

One major advantage of the event-based synchronization approach is that the individual representations of the shared content (i.e., the page layout) need not be the same for all participants. As user interactions are synchronized in relation to document elements, no matter where they are positioned or how they are rendered, the presentation can be adapted to the individual needs of the end user and his device. We are aware of the fact that differing presentations may destroy the common ground for collaboration, wherefore we suggest only adaptations that do not change semantics of content, such as adjusting presentation or level-of-detail of information. The following chapter provides more insight into the adaptation process in VCS.

3.3 Context-Aware Co-Browsing

An important trend to be faced by co-browsing tools is the increasing heterogeneity of web-capable devices. While in the 1990s the majority of PCs were rather similar in their capabilities, this has changed dramatically during the recent years. Target systems for co-browsing tools nowadays range from Smartphones, PDAs and Ultra-Mobile PCs (UMPCs) to Desktop PCs, and thereby screen sizes, interaction modalities and computing power differ tremendously. Furthermore, besides the problem of device-independence, it can be useful to take the situation (i.e., the context) of users into account. An example are web presentations, where the capability of taking private notes should be provided per-user, while buttons for moving backward and forward need only be seen by the presenter.

Figure 2: Adaptive Co-Browsing Views

For all these reasons it is necessary to adapt content shared by a co-browsing group to the capabilities, preferences and needs of their individual members. Thereby, co-browsing becomes context-aware (Figure 2). VCS introduces a concept for this adaptation process. Compared to the common approach of "relaxed WYSIWIS" (What You See Is What I See, i.e., adaptation only to the screen size) [Greenberg et al., 1996], our approach is more flexible and powerful.

In VCS, adaptation is performed on the basis of three related concepts, whose interplay is illustrated by Figure 3: a) semantically enriched web content, b) user and group context models, and c) the "Generic Adaptation Component"(GAC) [Fiala et al., 2005].

To allow for more sophisticated adaptation of web content, the adaptation subject – shared web content – needs to be described semantically. Since standard (X)HTML elements provide only little insight into the "meaning" of the content (i.e., *img* stands for an image, *p* for a paragraph of text), we need to enrich the documents with additional semantic information. Therefore, we use RDF/A [Adida and Birbek, 2006], a set of elements and attributes that embed RDF (a standardized format to model information in the form of subject-predicate-object expressions) into HTML and thereby allow for linking elements of a web page to concepts of external ontologies (data models). The embedded semantics can describe the content as well as relationships between web page components. Examples include concepts like "technical image" or "navigation bar" and relations such as "is-alternative-for" or "is-more-detailed-than". With the help of these semantics, it is possible to adapt the content to the user and his device, accordingly. Therefore, a concise model of the context is needed [Dey and Abowd, 1999]. We are working on an ontology-based context model that represents all relevant data needed to characterize the user, his device and situation. Contextual information is inferred from basic data monitored and sent by the VCS

clients, e.g., user interactions or hardware, software, and browser characteristics. With the help of consistency and inferencing rules, a "semantic context" can be derived from the purely "technical context", e.g., deducing the lighting conditions from the current time and location. Furthermore, we can model group characteristics in a so-called group context model which represents the context of all members of a co-browsing session. It can contain information on the lowest common denominator of all users as well as higher-level information that results from the combination of all users' parameters, e.g., "all project members are participating the session".

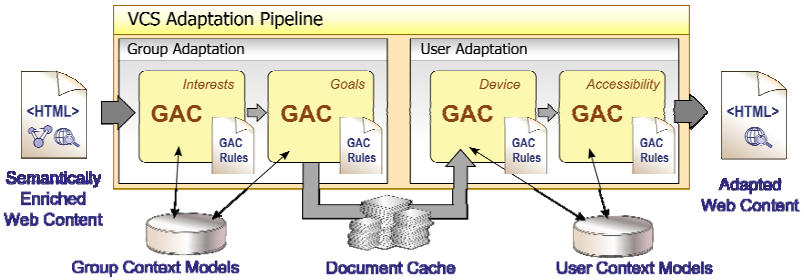


Figure 3: Adaptation Engine

The adaptation at runtime is performed by a pipeline-based *Adaptation Engine* (Figure 3). It performs a stepwise transformation of the input documents based on generic adaptation rules. For the transformations we use the Generic Adaptation Component (GAC), a transcoding tool facilitating adaptation in XML-based web applications. It is controlled by a rule-based configuration language, which is described in detail in [Fiala, 2007]. Due to the genericity of the rules, the *Adaptation Engine* works with any XML-based content, even standard HTML documents. Thereby, we not only support adaptation of RDF/A-flavoured content, but of arbitrary external content as well. We further believe that web content in general will be increasingly extended with semantics, especially as a result of automated authoring processes being part of Web2.0, as in blogs or wikis.

With the concepts described above, VCS allows for the flexible adaptation of shared web content. We do not presume a proprietary content model, but support any XML-based documents such as standard HTML pages. Thereby, arbitrary web applications can be co-browsed in a context-aware manner. The stepwise adaptation process may include various adaptation aspects that can take different contextual information into account. These aspects include among others device independence, accessibility and localization. In addition, our system can use other information, such as the user's

location, preferences, characteristics or roles to present the shared content accordingly. Besides the adaptation to single users our Adaptation Engine also facilitates transformations of web content depending on the context of the co-browsing group. This helps to provide a more consistent view and to improve the overall performance.

4. Related Work

There already exist a number of solutions, both as research prototypes as well as commercial products that address the concern of enabling collaboration on the Web. They can be divided into two approaches, namely application-sharing and co-browsing.

Application-sharing tools allow for sharing arbitrary applications (i.e., desktop programs), over the Internet and are thus not bound to a web browser. However, all of them require a separate software client to be installed (e.g., NetMeeting[6]) or at least rely on previously installed browser plugins (e.g., WebEx[7]). Synchronization is achieved by transferring screenshots in fixed time intervals, which results in high latency and thus negatively impacts the browsing experience. The use of a separate client further widens the gap between browsing alone and collaboratively.

Co-browsing tools, such as the system presented in this paper, are designed with focus on the collaborative use of web content. They are commonly browser-based, so there is no need to toggle between applications when switching to collaboration mode. This is a key requirement of the "1-click collaboration" [Esenther, 2002] and is well-supported by our approach. Due to their event-based synchronization, co-browsing systems are generally much faster than the image-based application-sharing solutions.

In the last decade, different co-browsing systems have been published that allow for the shared use of a web browser. Some of them are very application-specific, e.g. CoWeb [Jacobs et al., 1996], which supports synchronous filling of HTML forms, while others are more general in allowing a group of people to share the whole web page presented. The type and complexity of synchronization differ from the purely navigational replication of URLs and their content to more sophisticated techniques such as shared mousepointers ("telepointer" [Greenberg, et al. 1996] or "CoPointer" [Maly et al., 2001]), view slaving (following another user's scrolling position), annotations, synchronous form-filling (CoWeb) or whiteboard capabilities for web pages (PROOF [Cabri et al., 1999]). While these systems do their best to provide a decent synchronization – even of streaming content, as in CoLab [Hoyos-Rivera et al., 2006] – they rarely support application-specific interaction techniques. Our system can be configured domain-specifically and can therefore provide appropriate techniques for

[6] http://www.microsoft.com/windows/netmeeting/
[7] http://www.webex.com

particular collaboration scenarios. As an example, we mention our implementation of the "pencil metaphor" (cf. chapter 2.2) for the use in consulting scenarios.

Finally, co-browsing tools are facing additional synchronization problems today. With the advent of Web2.0, new forms of web applications have been emerging that heavily use dynamic content. For future web-based collaboration, support for dynamic content is crucial. As many of the previously mentioned solutions are somewhat aged, they typically have no or at least insufficient support for dynamic pages in contrast to VCS.

The majority of co-browsing solutions, e.g., academic prototypes like CoWeb and PROOF or products like BrowserFor2[8] and Bestscout[9], claim to be thin-client; however, they rely on Java applets for interaction monitoring and communication of the client with the co-browsing proxy server. Thus, a client-side installation of the Java Runtime Environment is assumed. Solutions like CoBrowser [Maly et al., 2001] further require the user to manually configure his browser to use a proxy server. This is inappropriate especially for non-computer experts that are the target group for online consultations. Other approaches like CoLab try to solve this by "automatic proxy configuration", which, however, still requires the user to manually adjust the browser settings.

The VCS-system presented in this paper provides two alternative approaches for the connection of client and proxy server: the default way using Ajax, or via a Java applet connection, offering a lower latency. Thus, it is "truly" a thin-client system, since it only relies on browser-based technologies. Similar work that is just as well only based on JavaScript has been presented in [Esenther, 2002]. However, it only allows content to be co-browsed that is on the same server as the co-browsing system. VCS in contrast supports co-browsing arbitrary websites.

Only very few co-browsing approaches take the heterogeneity of user groups and their devices into account. Most of them, e.g., GroupWeb and CoLab, implement "relaxed WYSIWIS" (cf. chapter 3.3) – something that is done automatically by every browser today. Personalization and device independence have been addressed only by few systems. In WebSplitter [Han et al., 2000], the author of the shared content needs to provide *policy files* to define access privileges for users (personalization) or devices (device-independence) to certain parts of a web page. The adaptation is performed by a proxy server, which then delivers the customized pages to the clients. [Coles et al., 2003] present a framework for multi-modal browsing on multiple clients, which heavily incorporates standards like XForms, XLink and XML Events. The requested content is not replicated, but every client is notified of a page change and requests the content itself, which results in individual views. As opposed to our approach, both systems do

[8] http://www.matthewssoftware.com/BrowserFor2/
[9] http://bestsella.com/produkte/bestscout/

not support arbitrary HTML content, since they use proprietary content models for their shared pages. They further need extensive authoring. Thanks to its genericity, our adaptation approach works well with arbitrary existing web content and does not require an author to prepare the shared content.

The framework for co-browsing on heterogeneous devices presented in [Chua et al., 2006] does not use a specific content model. Instead, it employs automatic web page analysis and structure detection algorithms to provide all session members with a so-called Shared Viewpoint (SVP), i.e., the replica of the web page adapted to the smallest common denominator of all participating devices. Additionally, each user is presented a Personal Viewpoint (PVP), that is, the web page adapted to his personal interests and device capabilities. However, due to the heuristic nature of the utilized web page transcoding algorithms, the adaptation (and especially the quality or usability of the resulting web pages) is often unpredictable [Hwang et al., 2003] and cannot be controlled by the web site's author, as in our system.

5. Conclusion and Future Work

In this paper we have described our co-browsing system VCS, which allows a group of users to collaboratively browse arbitrary and dynamic web pages, thus enabling new forms of collaboration and consultation for the growing number of web applications. Our system acts as a transparent proxy between users and the Web, with client-side parts running inside the browser without any configuration or installation of software. Interactions with the shared content, like mouse clicks, form-filling, highlighting, etc. are synchronized within the group as well as mouse pointer positions. To address the problem of heterogeneous end user devices, we employ an adaptation mechanism that dynamically adapts the shared content to the group's and individual users' capabilities and preferences. We have implemented most of the functionality described in a working prototype.

Currently, we are working on the full implementation of all features and prepare for an evaluation of our system, carried out together with a pilot customer. In the future, we plan to add more features and domain-specific interaction techniques. Another topic of research is security within our collaborative environment. We are investigating different ways to make VCS more secure, e.g., encrypting the client-server-communication and including an authentication mechanism. Another problem we are working on is improved group awareness that will make collaborative work even more realistic and include the evaluation of the importance of social protocols in virtual collaborative environments. We strive for new domain-specific interaction metaphors that make the user feel more comfortable in the collaborative use of his browser. Furthermore, we will

have to improve the handling of concurrent interactions, such as two people filling form fields at the same time. Finally, we plan to evaluate our system regarding scalability and plan to add load balancing mechanisms to be able to cope with support scenarios on a larger scale.

Acknowledgements

The VCS project is funded with means of the European Regional Development Fund 2000 – 2006 and with means of the Free State of Saxony.

References

Cabri, G.; Leonardi, L.; and Zambonelli, F. (1999): "Supporting Cooperative WWW Browsing: A Proxy-based Approach", In *Proc. of the 7th Euromicro Workshop on Parallel and Distributed Processing*, Madeira, pp. 138-145

Chua, H.N.; Scott, S.D.; and Choi, Y. W. (2006): "Framework For Co-browsing On Heterogeneous Devices". In *AINA '06: Proc. of the 20th International Conference on Advanced Information Networking and Applications*, Vienna, Austria, pp. 195-199.

Coles, A.; Deliot, E.; Melamed, T.; and Lansard, K. (2003): "A Framework for Coordinated Multi-Modal Browsing with Multiple Clients". In Proc. of the 12th International Conference on World Wide Web, Budapest, Hungary, pp. 718-726

Cyr, D.; Hassanein, K.; Head, M.; and Ivanov, A. (2006): "The role of social presence in establishing loyalty in e-Service environments". In *Interacting with Computers,* vol.19, no.1, pp.43-56

Dey, A.K. and Abowd, G.D. (1999): "*Towards a Better Understanding of Context and Context-Awareness*". Technical Report GIT-GVU-99-22, June 1999, Georgia Institute of Technology

Esenther, A. W. (2002): "*Instant Co-Browsing: Lightweight Real-time Collaborative Web Browsing*". Poster at the 11th International World Wide Web Conference (WWW2002), Hawaii, USA

Farnham, S.; Zaner M.; and Cheng, L. (2000): "Supporting Sociability in a Shared Browser". In *Proc. of the Interact Conference*, Tokyo, Japan

Fiala, Z. (2007): "*Design and Development of Component-based Adaptive Web Applications*". PhD Thesis, Dresden University of Technology, February 2007

Greenberg, S. and Roseman, M. (1996): *"*GroupWeb: A WWW Browser as Real Time Groupware" In *CHI'96: Conference Companion on Human Factors in Computing Systems,* Vancouver, Canada, pp. 271-272.

Han, R.; Perret, V.; and Naghshineh, M. (2000): "WebSplitter: A Unified XML Framework For Multi-Device Collaborative Web Browsing," In *CSCW '00: Proc. of the ACM Conference on Computer Supported Cooperative Work,* Philadelphia, USA, ACM Press, New York, NY, USA, pp. 221-230.

Hua, Z. and Lu, H. (2006): "Web Browsing on Small-Screen Devices: A Multiclient Collaborative Approach" In *IEEE Pervasive Computing*, vol. 5, no. 2, pp. 78-84

Hoyos-Rivera, G. J.; Gomes, R. L.; Willrich, R.; and Courtiat, J.-P. (2006): "CoLab: A New Paradigm and Tool for Collaboratively Browsing the Web". In: *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 36, no. 6, pp. 1074-1085

Hwang, Y.; Seo, E.; and Kim, J. (2002): "WebAlchemist: A Structure-Aware Web Transcoding System for Mobile Devices". In *Proc. of the WWW 2002 Workshop on Mobile Search*, May 2002, Honolulu, Hawaii

Jacobs, S.; Gebhard, M.; Kethers, S.; and Rzasa, W. (1996): "Filling HTML Forms Simultaneously: CoWeb - Architecture and Functionality". In *Proc. of the 5th International World Wide Web Conference on Computer Networks and ISDN Systems,* Paris, France, Anonymous Elsevier Science Publishers B. V, Amsterdam, The Netherlands, pp. 1385-1395.

Maly, K.; Zubair, M.; and Li, L. (2001): "CoBrowser: Surfing the Web Using A Standard Browser". In *Proc. of the World Conference on Educational Multimedia, Hypermedia and Telecommunications,* Norfolk, VA, pp.1220-1225

Manhart, P.; Schmidt, K.; and Ziegler H. (1998): "Group Interaction in Web-based Multimedia Market Places". In *Proc. of the 31st Annual Hawaii International Conference on System Sciences*

Marais, H. and Bharat, K. (1997): "Supporting cooperative and personal surfing with a desktop assistant". In *UIST '97: Proc. of the 10th annual ACM Symposium on User Interface Software and Technology,* Banff, Alberta, Canada, pp.129-138

Shaw, M. E. (1932): "A Comparison of Individuals and Small Groups in the Rational Solution of Complex Problems". In *The American Journal of Psychology*, vol. 44, no. 3, pp. 491-5