CHIME: A Metadata-Based Distributed Software Development Environment

Stephen E. Dossick and Gail E. Kaiser

Dept. of Computer Science, Columbia University[1]

CUCS Technical Report #CUCS-006-99

Abstract. We introduce CHIME, the Columbia Hypermedia IMmersion Environment, a metadata-based information environment, and describe its potential applications for internet and intranet-based distributed software development. CHIME derives many of its concepts from Multi-User Domains (MUDs), placing users in a semi-automatically generated 3D virtual world representing the software system. Users interact with project artifacts by "walking around" the virtual world, where they potentially encounter and collaborate with other users' avatars. CHIME aims to support large software development projects, in which team members are often geographically and temporally dispersed, through novel use of virtual environment technology. We describe the mechanisms through which CHIME worlds are populated with project artifacts, as well as our initial experiments with CHIME and our future goals for the system.

1 Introduction

Software development projects typically involve much more than just source code. Even small- to medium-sized development efforts may involve hundreds of artifacts -- design documents, change requests, test cases and results, code review documents, and related documentation. Significant resources are poured into the creation of these artifacts, which make up an important part of an organizations' "corporate memory." Yet it can be quite difficult for new project members to come up to speed -- and thus become productive contributors to the development effort.

The user interface research community has, in recent years, paid much attention to the development of techniques to help users assimilate a broad range of related information easily, through the development of 3D-based information visualizations (see [1] for example). These techniques excel at putting large volumes of related information in context for an unfamiliar user, as well as making it easier for more experienced users to find particular information they are looking for. See [2] for a description of many experiments and results in this area.

---

1. *Author's address:* Dept. of Computer Science, Columbia University, 500 W. 120th St. 450 Computer Science, New York, NY 10027 USA. *Tel:* +1 (212) 939-7000 *E-mail:* sdossick@cs.columbia.edu

Hypertext is another technique which has been widely recognized as being useful for contextually relating documents and other artifacts[3]. By placing links among related items, experts can leave trails through the information space. Later users can follow a "trail" of hypertext links which lead them to many related documents. In doing so, a user may, for instance, gain important insight into how various components of a software system are interrelated. Links may be automatically generated by tools as well (see [4]).

CHIME, the Columbia Hypermedia IMmersion Environment, is a framework that aims to synthesize results from both these research communities to create a software development environment for managing and organizing information from all phases of the software lifecycle. CHIME is designed around an XML-based metadata architecture, in which the software artifacts continue to reside in their original locations. Source code may be located in a configuration management system, design documents in a document management system, email archives on a corporate intranet site, etc. Thus CHIME does not handle storage of artifacts -- users continue to use their existing tools to access this data while CHIME provides data organization and hypertext linking capabilities. CHIME maintains only location and access information for artifacts in the form of metadata. CHIME uses an extensible Virtual Environment Model and dynamic theme component (described below) to generate a Multi-User Domain style virtual world from the metadata. The virtual world may take the form of a 3D immersive virtual reality (as in many contemporary games like "Quake" from Id Software) or a simple text world (as in the original "Adventure" and "Zork" games from the 1970's and 1980's).

In the CHIME virtual world, users interact with project artifacts by "walking around," where they potentially encounter other users' representations (avatars). While incidental encounters add a sense of realism to a virtual world, a potentially more useful application of this technology (which CHIME supports) is to allow a novice user to collaborate with an expert by finding his avatar and beginning a conversation. Geographically and temporally dispersed users thus easily gain context with work being performed elsewhere.

This paper proceeds as follows: in the next section, we discuss the model which underlies CHIME, followed by a discussion of our current architecture. Next, a description of a recent experiment in which we used CHIME to build a 3D environment from the Linux operating system kernel, documentation, and email archives. Following this, we describe related work in a variety of domains, including Software Development Environments (SDEs), Software Visualization, Information Visualization, and Metadata architectures. Finally, we discuss the contributions of this work and some future directions.

2 Model

The conceptual model underlying CHIME is comprised of three main components: Groupspaces, Groupviews, and Software Immersion. In this section, we describe these components and their relationships.

We use the term Groupspace to describe a persistent collaborative virtual space in which participants work. The participants may be geographically or temporally distributed, and they may be from different organizations cooperating on a common project (subcontractors on a defense contract, for example). Contained within the groupspace are project artifacts as well as the tools used to create, modify, and maintain them. Artifacts may be organized and re-organized at will by project participants.

Central to the Groupspace concept is the idea that project artifacts continue to exist in their original form in their original repositories. This differs from traditional Software Development Environments (SDEs) (like Oz[5], SMILE[6], Desert[7], Sun NSE[8], Microsoft Visual C++[9]), as well as most traditional Groupware systems (like eRoom[10], TeamWave[11], and Orbit[12]) in which artifacts are under the strict control of the environment. In these systems, users are expected to access artifacts only through the development environment's cadre of tools or via COTS tools specially "wrapped" to work with the environment. In a Groupspace, artifacts continue to exist in their legacy databases, configuration management systems, bug tracking systems, rationale capture tools, etc.

Additionally, Groupspaces may contain information generated within the space. A particular Groupspace may contain built-in tools and services to be used by participants, e.g. to add arbitrary annotations to particular artifacts, hold real-time chat sessions, add hypertext links on top of (and separate from) artifacts in the system, semi-automatically propagate knowledge among participants (in the manner of a recommender system [13]), etc.

We use the term Groupviews to describe multiuser, scalable user interfaces used to navigate and work in a Groupspace. In addition to allowing Groupspace participants to find and access relevant information quickly (as they might in a single user system, or a system in which they had no knowledge of other users' actions), Groupviews keep users informed about work being performed by fellow users.

Groupviews build on research and commercial work in Multi-User Domains (MUDs) [14], chat systems [15], virtual environments [16], and 3d immersive games [17]. In a Groupview, a set of virtual environment rooms containing project artifacts is generated from the organization of the artifacts in the Groupspace. Rather than placing artifacts into these rooms arbitrarily or according to some external mapping mechanism (as in Promo[26], where the mapping from artifacts to rooms is created from a software process definition and cannot be modified by users without corresponding modification to the process), a Groupview generates the rooms and connections between the rooms from the artifacts themselves. For example, a software module might become a room in the Groupview, and the source files making up the module might be furnishings inside the room. Corridors might link the modules' room with rooms containing design documentation, test reports, and other artifacts related to the code.

A core aspect of Groupviews is the ability to provide selective awareness of other users' actions. Participants' locations in the virtual environment, as well as their scope of interest (i.e. the project or projects they are currently involved in, portions of the system they are considered "expert" in, related documents they have recently read, written, or modified, etc.) are shared among other users. In the case of a Groupview involving

multiple teams working on separate (but interrelated) portions of a project, it should be possible for users to "tune" awareness so they receive only information relevant to them and their work.

It is important to note that our discussion of the Groupview model does not specify that they must be built as graphical or 3D "virtual reality" style user interfaces. As the very successful IRC system [15] and the literally thousands of text-based MUDs available on the internet have shown, 3D graphics are not necessarily required to provide an immersive experience to users. As shown in [32], users utilizing immersive environments for real work can get quite involved with a simple text-based user interface, even to the extent of ignoring their family life in favor of their virtual one.

In a Software Immersion, the third component of the conceptual model underpinning CHIME, team members collaborate and perform individual tasks in a virtual space defined by the structure of the artifacts making up the software system. This builds in some respects on previous work done in the Software Visualization community (see [18]) in which visualizations of module interactions and interrelations are created. The primary difference here is that a Software Immersion is intended to be built semi-automatically, while most software visualizations are generated by human experts. When visualizations have been created by software, the generating software has been built to handle a certain small class of input (output from sorting algorithms for example as in [20]).

When applied properly, Software Immersion can speed the learning curve faced by new project members. The architecture and organization of the system they are learning is no longer an abstract concept, it is something they can walk around in and inhabit. Software Immersion is similar in concept to emerging technology in use in Civil Engineering. In [21], the author shows quantitatively that new construction project members come up to speed faster and perform fewer mistakes when an immersive, virtual construction environment is built from the building design.

CHIME benefits from the synergy among these three conceptual components. Awareness mechanisms in Groupviews work to enhance Software Immersion, since participants are now immersed not only in the software artifacts but into the actions of others around them as well. Making this information available helps to fulfill the Groupspace goal of aiding geographical and temporal dispersion among project participants. Groupspaces benefit from Groupviews' visualization aspects, as users are able to locate information in the underlying Groupspace by quickly navigating the Groupview. This allows them to learn "where" information exists. In addition, users are drawn to new information via notifications from the awareness mechanisms in a Groupview.

Realization of this conceptual model is challenging. Groupviews are dynamic environments. As artifacts are added, modified, deleted, and moved in the underlying Groupspace, Groupview participants must find the virtual environment evolving as well. This may be as simple as periodically bringing new or newly modified artifacts to their attention, or as complex as "morphing" the world as they see it to a new layout, based on a major change to the underlying Groupspace layout. See [22] for a discussion of the transaction management and version control issues inherent in handling these dynamic notifications. Groupspaces face the problem of working with data in remote re-

positories which may or may not include any transaction or lock support -- and can thus change at any time. The architecture designed to implement this model, described below, attempts to handle these challenges while maximizing the benefits of the CHIME conceptual model.

3 Architecture

CHIME's architecture was designed around three main components, as illustrated in Figure 1. In this architecture, separate services are responsible for organizing artifacts, parameterizing those artifacts as virtual environment types, and dictating how the virtual environment appears to clients. We will describe each of the components in turn.

The Xanth Data Service (evolved from our lab's previous work on OzWeb[24]) address-

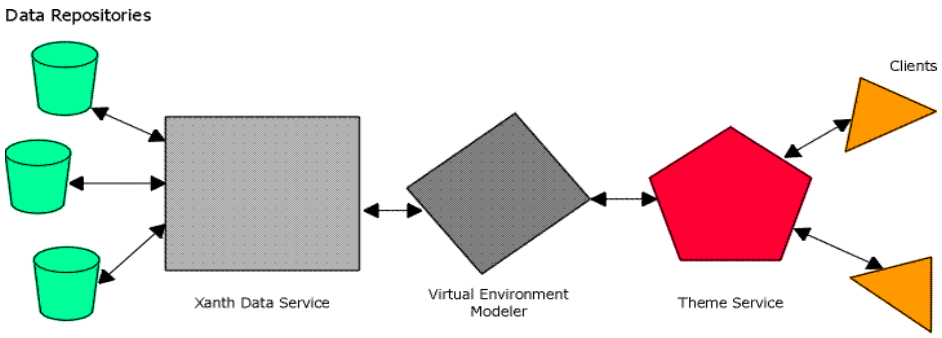

Figure 1 CHIME architecture.

es the data organization and hypermedia aspects of the Groupspace model. In Xanth, data is organized into a multi-rooted tree hierarchy of XML elements known as Xanth dataElements. Each dataElement refers to a single piece of information living in an external data repository of some kind (web server, configuration management system, document management system, relational database, etc.) The Xanth Data Service maintains an XML document (made up of these dataElements) which completely describes the contents of the Groupspace. Figure 2 shows an example dataElement with the minimum set of fields filled in.

From the figure, we see that every dataElement includes a name, a unique id number,

```
<dataElement
name="README"
id="1000"
protocol="http"
server="library.psl.cs.columbia.edu"
port="80"
path="/linux-2.0.36/README"
hidden="false"
parent="0"
behavior="GET"
```

Figure 2 XML description of dataElement

as well as a "parent" field specifying its parent dataElement. The protocol-related fields (protocol, server, port, and path) describe what mechanism is to be used to retrieve the data associated with a particular dataElement. In the example above, the dataElement is named "README" and is an http-accessible file. The server, port, and path fields simply give location information for the dataElement.

Xanth uses protocol plugins to implement the retrieval protocols specified in the dataElements. Continuing our example, an HTTP plugin would be configured into this instance of Xanth. A basic protocol plugin is quite simple; all it can do is verify that the server, port, and path fields of a given dataElement are of the proper format for this protocol. To become more useful, each protocol plugin may provide a set of "behaviors" for its dataElements. Without these behaviors, Xanth cannot perform any actions on the data. The HTTP plugin, for example, may provide behaviors for the basic HTTP methods, namely GET, POST, PUT, etc. If the protocol plugin provides behaviors, it is expected to add a "behavior" field to each of its dataElements' XML. This field contains a list of behaviors supported for this dataElement, and may be used by other components of the system to determine the actions the user can take with a given dataElement. In Figure 2, we can see that the HTTP plugin has added the GET behavior to this dataElement, indicating it is able to fetch the document on behalf of the user, if needed.

To address the hypertext features of the Groupspace model, Xanth includes a Link Service which provides typed, n-ary, bidirectional hypertext links among elements. The Xanth Link Service maintains its own XML document made up of linkElements (see Figure 3). Each linkElement has 3 fields: a unique id number, a descriptive type field, and a list of dataElement id's which are part of this link. The hypertext model followed by the Xanth Link Service is different from the hypertext model underlying the WWW -- in Xanth, hyperlinks are stored separately from the data they reference, while WWW pages embed link references inside their content. Xanth's model is richer,

supporting more sophisticated hypertext among artifacts. It is conceptually similar to the hypertext provided by the various Open Hypermedia Systems [23].

```
<linkElement
id="924"
type="Related Docs"
dsElems="2048,1000"/>
```

Figure 3 XML description of linkElement

The second component of the CHIME architecture is the Virtual Environment Modeller (VEM) service. This service is responsible for parameterizing each dataElement with one of an extensible set of virtual environment types. These types are meant to be representative categories for the various parts of a virtual environment. To date, we have defined only three types: Component, Container, and Connector.

In the VEM schema, Components are a base type; they are the default type given to every dataElement. Child VEM types "derive" from Component in the standard manner of Object-Oriented databases: they inherit fields from the parent type, and can be treated as an instance of that parent type, and also include their own fields as a form of specialization. The type 'Container' (which derives from Component) is given to dataElements which, for the purposes of the virtual environment, have a set of elements inside of them. A Connector (which itself derives from Container) not only may contain elements but explicitly connects two or more Containers. The VEM parameterizes each dataElement from the Data Service by adding an XML field called "VEMtype" to each dataElement. This information is used by the Theme Service (see below) to determine the role of each artifact in the resulting virtual environment.

Note that despite dealing with virtual environment concepts, the VEM does not hardwire a particular notion of how the virtual environment will present itself to a user. We have deliberately designed the VEM and Xanth DataService to remain neutral with regard to the final user interface and display mechanisms used to produce the virtual environment from the Groupspace artifacts. To borrow a term from more industrial disciplines, the CHIME architecture can be seen as an "assembly line" moving remote data into the Xanth Data Service (where their location information is stored), next into the VEM (which decides their eventual roles in the virtual environment) and finally to the CHIME Theme Service.

The CHIME Theme Service is responsible for all aspects of the virtual environment created for and inhabited by the system users. The Theme Service is broken into two components, Theme Plugins which run in a CHIME client and a MUD service which runs in the CHIME server. The MUD service is quite simple. It is responsible only for keeping track of the locations of the system users (i.e. what VEM container element are they currently located in) as well as relaying chat messages to all users of a particular room.

CHIME clients connect to the Theme Service and download available Theme Plugins. These Theme Plugins are then started in the client and are responsible for retrieving the XML document maintained by the Xanth Data Service. From this XML, the Theme Plugin lays out a virtual world according to the capabilities of the client. If the client has 3D capabilities, the theme plugin may build a 3D representation of the world and allow the user to walk through it. If the client is connected to the server via a very low

bandwidth connection, or is running on a slow system, the Theme Plugin may layout a textual world. In the CHIME architecture, all user interface decisions are left to the Theme Plugins at run time.

4 Implementation

Our initial implementation work on CHIME is complete. The architecture described above has been implemented and we have performed an initial experiment designed to test the system's scalability with data from a large software system, the Linux 2.0.36 kernel. We have loaded our experimental implementation with source code, build instructions (including Makefiles and more human-oriented instructions), documentation artifacts (both the informal documentation provided with the kernel source as well as well-known web pages providing tutorials and information about the Linux kernel), and web-based archives of the linux-kernel mailing list (used by the core developers and maintainers of the various kernel subsystems for technical discussions). All in all, this project includes over 1.2 million lines of source code and several hundred megabytes of documentation. Where possible, we have attempted to use CHIME's hypermedia capabilities to cross link (by hand) the external documentation artifacts with the source modules they deal with. The aim of this experiment was to simulate a large, complex, ongoing software effort using CHIME for its day-to-day work.

Our initial CHIME client and Theme Plugin build a simple 3D virtual environment from this data. Source code modules are rooms in the environment, individual files are rendered as furnishings in those rooms. Project members explore artifacts by walking around the virtual environment, and can interact with the artifacts and each other. As of this writing, we have integrated only a few tools into our prototype, notably video-conferencing software to allow participants to communicate with each other, web browsing software for accessing the numerous web-accessible artifacts related to the Linux kernel, and a text editor allowing participants to edit source code. We intend to reuse our Rivendell web-based tool server[24], originally developed for the OzWeb system, to provide more sophisticated tool launch and management facilities in CHIME.

5 Evaluation

The implementation presented here is actually CHIME 2.0. We previously developed another version as a prototype in which the clients used Virtual Reality Modelling Language (VRML) browsers to interact with the environment. It provided a similar immersive experience to users, but the conceptual model underneath was not as fully developed. Clients connected directly to the Groupspace; Groupview techniques were ignored.

The most significant limitation of CHIME 2.0 is that the existing implementation ignores versioning and transaction management issues in the environment. We are addressing this problem, as discussed in [22] and hope to incorporate an implementation quickly.

Another, less significant limitation is the absence of an easy mechanism for populating a CHIME instance with artifacts. The Xanth Data Service operates on XML, and in the current implementation is users are expected to provide XML describing new dataElements to be added. We intend to address this limitation by creating a simple GUI-based mechanism for new artifacts to be added inside a CHIME environment.

CHIME 2.0 is implemented entirely using Java 2 (aka JDK 1.2), and the initial Theme Plugin discussed here uses the SGI OpenInventor 2.1 api to provide 3D graphics capabilities. Our use of Java and OpenInventor allows CHIME to be portable; although our primary development platform is Windows NT, we have successfully used CHIME on both Sun and SGI unix workstations. We have attempted, where possible, to utilize existing technologies in constructing CHIME, including standard Java remote method invocation (RMI) for communications between CHIME server components, and Sun's XML processor [25] for all of CHIME's XML requirements.

6 Related Work

LambdaMOO [14] is prototypical of many Multi-User Domain systems, and many newer systems are still built around the original LambdaMOO implementation. LambdaMOO, through the use of an object oriented database and associated programming language, explored many of the ideas in Groupviews. We chose not to build on LambdaMOO, however, because the OODB underlying the system must contain all virtual environment components. This does not fit our Groupspace model for storage of the artifacts.

Promo[26] builds a virtual environment interface from a software process definition. Rooms in the environment are mapped to subcomponents of the process. In the environment, artifacts are located in the rooms in which the process will utilize them (i.e. a room for compiling code will contain the code, etc.) This is the first work the authors are aware of which attempts to marry virtual environment techniques with software development environments.

A number of Software Development Environments (SDEs) have been created over the years in both research and industry (see [6], [7], [8], [9], [5] for example). These differ from our conceptual model in that they assume that all artifacts will be managed by the development environment itself (or through tools specially "wrapped" to be called from the environment). In addition, existing SDEs do not provide the Software Immersion inherent in CHIME.

Many research and commercial Groupware systems might at first glance appear to fulfill our Groupspace model. Systems like Orbit[12], TeamRooms[11], eRoom[10], and Lotus Notes[27] do have much in common with CHIME Groupspaces, but these systems store artifacts inside their servers. When they do allow reference to external data, it is often limited to a web link to external data. A number of such system have explored similar awareness mechanisms as our Groupviews.

Microsoft NetMeeting[28] and other real-time collaboration tools provide a form of temporary Groupspace. While in a meeting, users can share applications (effectively making single-user COTS tools multiuser), use these tools to bring in data from external sources, and have some awareness of others' actions. These workspaces, however are

not persistent; when the last participant leaves a meeting, the workspace disappears. In addition, these tools do not scale well beyond a few users and a relatively small number of artifacts - bandwidth and memory issues in real time collaboration make this difficult.

Research into Software Visualization (and the related area of Algorithm Animation) looks at the design and development of techniques to show program code, algorithms, and data structures by using typography, graphics, and animation. The Software Immersion in our conceptual model for CHIME can be seen as a form of Software Visualization, as we are displaying the organization of software artifacts through the design of a virtual environment. [19] contains a good overview of research in this area.

A number of research and commercial systems (see, for instance [29] and [30] utilize metadata style architectures to provide access to back-end, remote data. These systems are typically focused on query optimization and similar database research issues over remote data sources. This work is quite relevant to the Groupspace concept, as a powerful query facility optimized for use in a metadata-based system would be a useful addition to the Groupspace model.

An ongoing conference series discusses the use of Multi-User Domains for "real" work [31]. Research results from this community demonstrate many examples of the use of MUDs and virtual environment systems in engineering disciplines as well as other work areas.

7 Conclusions and Future Work

We have designed a conceptual model, feasible architecture, and performed an initial implementation of a metadata-based software development environment utilizing virtual environment techniques. The model is made up of three separate components: Groupspaces, which provide a persistent organization of software artifacts, Groupviews, multiuser user interfaces including awareness mechanisms, and Software Immersion, which creates an immersive environment from the artifacts populating a Groupspace.

One area for future work on CHIME is support for easy population of a Groupspace with a set of objects from existing development efforts, as well as automatically generating links between artifacts which seem to be related. A number of efforts in the Rationale Capture and Reverse Engineering communities have explored this, and we intend to look into utilizing existing systems of this sort with CHIME.

We plan to incorporate the Rivendell tool management system we have previously developed[24] to provide tool launch and sharing capabilities to CHIME. Although our experiment with the Linux kernel involved a small number of tools, we envision much richer tool support for future CHIME incarnations.

As mentioned in our evaluation of CHIME, we intend to incorporate research into versioning and transaction support for virtual environments from[22]. This is the most glaring omission from the current implementation, and must be addressed quickly.

It has been the custom of our research lab to use our own tools to support our work. In this regard, we intend to use CHIME for our own development work, involving a small team of programmers (5-10) working on a number of separate but interrelated projects.

8 Acknowledgements

We thank Scott Gross, Janak Parekh, Dan Port, Adam Stone, Chris Vaill, and Jack Yang for useful technical discussions and feedback. The Programming Systems Lab is sponsored in part by the Defense Advanced Research Projects Agency, and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-2-0022, and in part by an IBM University Partnership Program Award. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, the Air Force, the U.S. Government, or IBM.

9 References

[1] S. K. Card, George G. Robertson, and Jock D. Mackinlay. The Information Visualizer, an information workspace. In *Human Factors in Computing Systems Conference*, 1991 page 181.

[2] D. F. Jerding and J. T. Stasko. Using information murals in visualization applications. In *Proceedings of 8th ACM Symposium on User Interface and Software Technology*, 1995, pp. 73-82.

[3] F. Halasz and Mayer Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30-39, February 1994.

[4] G. E. Kaiser, S. E. Dossick, W. Jiang, J. J. Yang. An architecture for WWW-based hypercode environments. In *Proceedings of 1997 International Conference on Software Engineering: Pulling Together*, May 1997, pp. 3-12.

[5] I. Ben-Shaul and G. E. Kaiser. *A Paradigm for Decentralized Process Modeling*. Kluwer, 1995.

[6] G. E. Kaiser and P. H. Feiler. Intelligent assistance without artificial intelligence. In *Proceedings of 32nd IEEE Computer Society International Conference*, February 1987, pp. 236-241.

[7] S. P. Reiss. Software visualization in the Desert environment. *ACM SIGPLAN Notices*, 33(7):59-66, July 1998.

[8] Sun Microsystems, Inc. *Introduction to the Networked Software Environment*. March, 1988.

[9] Microsoft Corp. Visual C++. http://msdn.microsoft.com/visualc.

[10] Instinctive Corp. eRoom. http://www.instinctive.com/html/eroom.html.

[11] TeamWave Software, Ltd. TeamWave Workplace. http://www.teamwave.com.

[12] T. Mansfield, S. Kaplan, G. Fitzpatrick, T. Phelps, M. Fitzpatrick, R. Taylor. Evolving Orbit: a progress report on building locales. In *Proceedings of Group97*, ACM Press, Phoenix, AZ, Nov 1997.

[13] B. Sarwar, J. Konstan, A. Borchers, J. Herlocker, B. Miller, and J.Riedl. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *Proceedings of the 1998 Conference on Computer Supported Cooperative Work*, Seattle, November 1998.

[14] P. Curtis. MUDs Grow Up: Social Virtual Reality in the Real World. *Proceedings of 1992 conference on Directions and Implications of Advanced Computing*.

[15] MIRC, Inc. Introduction to IRC. http://www.mirc.com/irc.html.

[16] Electric Communities, Inc. The Palace. http://www.thepalace.com.

[17] Id Software, Inc. DOOM. http://www.idsoftware.com

[18] L. Feijs and R. de Jong. 3D visualization of software architectures. Communications of the ACM, 41(12):73-78.

[19] J. Stasko et al., editor, *Software Visualization: Programming as a Multimedia Experience*, MIT Press, 1998.

[20] Marc Brown and Marc A. Najork. Algorithm animation using 3D interactive graphics. In *Proceedings ACM Symposium on User Interface Software and Technology*, pp. 93-100, November 1993.

[21] D. B. Hogan. *Modeling construction cost performance: a comprehensive approach using statistical, artificial neural network and simulation methods*. Columbia University Dept. of Civil Engineering PhD. Thesis, 1998.

[22] J. J. Yang. *An Approach to Cooperative Transaction Services on the WWW*. Columbia University Dept. of Computer Science Technical Report CUCS-008-99, 1999.

[23] P. J. Nurnberg. *Brief description of the Open Hypermedia Systems Working Group*. http://www.csdl.tamu.edu/ohs/intro/preface.html.

[24] G. E. Kaiser, S. E. Dossick, W. Jiang, J. J. Yang and S. X. Ye. WWW-based Collaboration Environments with Distributed Tool Services. *World Wide Web*, Baltzer Science Publishers, 1:3-25, January 1998.

[25] Sun Microsystems, Inc. Java Project X XML Parser. http://developer.java.sun.com/developer/earlyAccess/xml/index.html

[26] J. C. Doppke, D. Heimbigner, and A. L. Wolf. Software Process Modeling and Execution within Virtual Environments. *ACM Transactions on Software Engineering and Methodology*, vol. 7, no. 1, January 1998, pp. 1-40.

[27] Lotus Development Corp. Lotus Notes. http://www.lotus.com.

[28] Microsoft Corp. NetMeeting. http://www.microsoft.com/windows/IE.

[29] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom. "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS". In *Proceedings of the AAAI Symposium on Information Gathering*, pp. 61-64, Stanford, California, March 1995.

[30] Novera Corp. Novera jBusiness. http://www.novera.com.

[31] Collaborative Virtual Environments 98. Online information site. http://www.crg.cs.nott.ac.uk/events/CVE98/index.html

[32] Kraut, R. Patterson, M., Lundmark, V., Kiesler, S, Mukophadhyay,T & Scherlis, W. (1998). Internet paradox: A social technology that reduces social involvement and psychological well-being? *American Psychologist*, Vol. 53, No. 9, 10171031.