So Many Users — So Many Perspectives

Bastiaan Schönhage[1,2]*, Peter Paul Bakker*[1,2] *and Anton Eliëns*[1]

1 - Vrije Universiteit
Dep. of Mathematics and Computer Science
De Boelelaan 1081, 1081 HV Amsterdam
The Netherlands
email: {bastiaan, ppbakke, eliens}@cs.vu.nl

2 - ASZ Research & Development
Postbus 8300, 1005 CA Amsterdam
The Netherlands
email: schonhage@gak.nl

Abstract

Visualization — the transformation of data and information into multimedia including pictures, animation and 3D scenes — enables users to understand information more naturally. It reveals patterns and relations in the information which may otherwise remain hidden. As a consequence, it can provide a single user with enough *valuable* information to support decision making. In addition to this, visualization can also be used to explain information to other people. In this case, the results of visualization are deployed as arguments in collaborative decision making.

This paper discusses a distributed visualization architecture which supports collaborative decision making. The architecture is designed with the following consideration in mind: "multiple users, with different information needs, require multiple views or perspectives of the data." Additionally, in order to support the cooperation between users during the decision making process, we extend the architecture with collaborative aspects including session management, and the exchange of visualization perspectives.

Keywords

Visualization, Software Architecture, Collaboration, Decision Making

1 INTRODUCTION

Visualization is used to give better insight into data by showing a visual representation of the information. Visualization is becoming increasingly important because people are suffering from an information overload caused by enormous amounts of

data. By using visualization we can first explore a comprehensive overview of the information and later decide to zoom in on the details.

Currently, people are using visual representations of information for two different purposes. First, visualization is often used to understand information. A visualization gives quick insight into information using humans' remarkable perceptual abilities (Shneiderman 1998). Second, visual representations are used to show information to other people. Shneiderman (1998, p. 522) states that the bandwidth of information presentation is potentially higher in the visual domain than it is for media reaching any of the other senses. For example, news papers are full of graphs to show economic growth or the developments on the stock exchange market. In the first case, when using visualization to understand information, we often apply it individually (although it is surely useful to try to understand information in a group process). In the latter case we are communicating with other people because we try to illustrate something, or we want to convince them of our point of view.

In addition to static visualizations (2D images), current technology enables a new form of visualization: the interactive visualization of dynamic data. In recent years, the desktop computer has evolved from a text/picture based system to a fully multimedia-enabled workstation. This offers a great opportunity to deploy visualization on multimedia desktop computers. Visual representations consisting of interactive 2D or 3D animations enable the visualization of dynamic data coming from, for example, running simulations. Furthermore, multimedia PCs connected to fast networks allow desktop video conferencing, enabling direct user-to-user communication.

Structure The next section illustrates why visualization is useful to support collaborative* decision making in a business process re-design project. Section 3 briefly describes the *distributed visualization architecture* (DIVA), intended for multi-user visualization. After discussing issues in collaborative visualization in Section 4, we will describe our architecture extended with collaborative aspects in Section 5. The sixth section illustrates DIVA from a user's point of view by means of a sample visualization. Finally, in Section 7, we will end up with conclusions.

2 RE-DESIGNING BUSINESS PROCESSES AT THE GAK

Our example concerns the GAK (*Gemeenschappelijk Administratie Kantoor*), which is the largest provider of social security in the Netherlands. The GAK organization's main services are the registration and collection of insurance premiums, and the payment of social security benefits.

ASZ, which is the IT company of the GAK Group, builds and maintains the information systems that the GAK is using. Currently, the information infrastructure consists of several large databases, hundreds of separate applications and little inte-

*In this paper we will use the terms collaboration and cooperation interchangeably.

gration. To improve this, ASZ is investigating a software architecture that combines the databases, legacy software and new applications into a highly integrated system.

This development will certainly have an impact on the business processes at the GAK. The company will be able to improve current services and to offer new ones. However, deciding how the new business processes must be organized and what the consequences of some decisions will be is not at all trivial.

To assist the managers in studying the alternatives we create business-process simulations to *execute* the re-design alternatives (Eliëns, Niessink, Schönhage, van Ossenbruggen & Nash 1996). The managers are now able to run the simulations and experiment with the re-design alternatives themselves. To fully exploit the potential of business simulations we allow the managers to visualize and discuss both the results of the simulation, e.g. the costs and profits, and the running simulation itself, e.g. to illustrate the activities in the re-designed alternative.

Example: registration of new companies

As a concrete example, consider the process of registering the employees of a newly established company for social security. In the past, the employer had to go to a number of counters to fill in the required forms. When the client had forgotten something needed for the registration, he had to go back to get it and start the whole procedure again.

As a re-designed alternative we want to explore two options. First, all the paper forms could be combined into a single computer application. All forms could then be filled in at once, in dialogue with a single GAK employee. Second, a GAK employee might be able to go to the newly established company. There, using a laptop, she could fill in all of the needed information by asking it directly to the client on the spot.

To decide which alternative is preferable, we have to consider a number of aspects including the cost of the alternative, the satisfaction of the clients, and the time needed to register the company.

A visualization of the *business process flow* is useful in explaining the business process alternatives, . This illustrates who is performing which tasks and how the information flows through the model. Additionally, a *geographical visualization* shows how far and how often clients and employees of the GAK have to travel. The costs, waiting times and other statistical information of the re-designed alternatives can be presented using *statistical visualizations*, such as charts and histograms.

The decision makers, who are spread out over the country, plan to make the definitive decision at a meeting. However, before that, they want to prepare and discuss several alternatives. The above mentioned visualizations offer the decision makers (and other interested employees) a common ground for discussion.

Essentially, we want to support two forms of collaboration: *synchronous distributed* and *face to face* (Ellis, Gibbs & Rein 1991). In order to help the participants prepare for the meeting, we first support *synchronous distributed* collaboration where the

users cooperate at the same time but in different places. Secondly, at the meeting, where the decisions are made, the decision makers will discuss the selected alternatives *face to face*, i.e. same time, same place.

3 MULTI-USER VISUALIZATION

As the above example illustrates, it is useful to take visualization from the single user domain into the realm of distributed multi-user systems. This makes it possible to discuss shared information sources.

However, multiple users with different backgrounds have different information requirements. In the example above some managers might be interested in the resource allocation (who is using what) of the re-design alternative, while others might be more interested in the financial aspects. To support these different information requirements, multiple perspectives (or views) on the information are required. So, based on the same simulation, we distinguish alternative perspectives that visualize different aspects of the re-designed process.

The need to have multiple perspectives was the main motivation for designing the *distributed visualization architecture* (DIVA). Additional requirements were the support for interactive visualization to allow for experimentation, and visualization at the user's desktop by means of a networked or web-based architecture (Schönhage & Eliëns 1998).

We regard the process of visualization as a transition of data through a sequence of models, starting with the generation of data and ending with the presentation of a visualization (Schönhage & Eliëns 1997). To allow for multiple perspectives on the data, we introduce an intermediate model between the generation and presentation of information. This intermediate model contains information based on the originally generated data, adapted to the information requirements of its users.

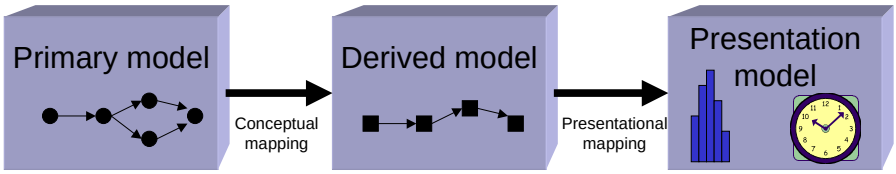

Figure 1 Conceptual architecture

Figure 1 depicts our architecture on a conceptual level. The primary model is the source of the information and contains explicitly or implicitly all information available. A conceptual mapping gives us the ability to adapt the raw information in the primary model to our information needs. Consequently, information in the derived model differs from data in the primary model in two ways. Primarily, only valuable information is selected to be present in the derived model and, secondly, information derived from primary data is added in the derived model.

How we present the derived information is specified in the presentational mapping. Here, information concepts in the derived model are mapped onto generic visualization primitives. The final presentation is the content of the presentation model. For example, when using DIVA for 3D visualization, the presentation component contains a 3D scene through which end users can navigate. For a more detailed description of DIVA see Schönhage & Eliëns (1997) and (1998).

4 COLLABORATIVE VISUALIZATION

In DIVA, multiple users can have their own presentation model (perspective) while sharing a common resource. However, in this approach the different users have the feeling that they are the only user of the shared resource. There is not yet support that allows a user to be aware of other users or to interact with them. Our goal is to expand the architecture to support users to collaborate with each other. Here, 'to collaborate' means that the users are able to discuss visualized information in order to reach a decision.

The next section will address the issues that are involved in this restrictive notion of collaborative visualization. Then, we will discuss the requirements for an extended DIVA architecture in Section 4.2.

4.1 Issues

DIVA focuses on visualizing information from different perspectives. We can distinguish between two distinct phases of activities within this approach.

The first phase is to define and experiment with the perspectives. This activity is done mostly in solitude, although multiple users can share a primary model or a derived model. The purpose is to determine the information need and the relevant data for that need.

The second phase is that of multiple users collaborating by reviewing and discussing the different defined perspectives. This article focuses on the latter, the collaboration phase.

When a group of people collaborates, the group members must share a common workspace (Ellis et al. 1991). The task of a group of users is to interact with each other and present different views on shared information. Let us assume that the goal is to reach a decision, for instance, concerning which model to choose for a business process re-design project, as in the example of Section 2.

Sessions Collaboration normally takes place in some kind of meeting, which can differ in interaction protocols, group size, formality, etc. Each participant of the collaboration can have one or more *roles* depending on the sort of meeting. A role is a set of rights and obligations (Ellis et al. 1991). We distinguish the following roles: chair, listener, talker and interactor. The chair sets up the session, a listener is a passive par-

ticipant, a talker is actively explaining his arguments and, finally, an interactor is able to interact with shared resources. The rights and obligations of the different roles are determined by the *interaction protocol*. The possibility to switch roles dynamically is important, since a listener can change into a talker from one moment to the other.

Collaborative visualization in DIVA is a virtual meeting, where the participants are at different places and their desktops are connected by a network. We will call the event of such a virtual meeting a *session*. Session management should support several kinds of sessions and thus be able to handle changing numbers of participants, their roles and interaction protocols.

The notion of *subgroups* makes it feasible to split the total group of participants in (non disjoint) groups. These subgroups can communicate separately or perform subtasks. Subgroups can come into existence dynamically.

Sharing perspectives It is important that the cooperators can show their personal perspective or view to other participants, in order to support their arguments in a discussion. One way to share views is for one participant to *enforce* his perspective onto another user or group. Views can also be shared by means of a *perspective repository*, where participants can select a perspective they would like to consider. The perspectives they can choose from, must be deposited by other participants. This implies that not every participant should have to create her own perspective before joining a session. Obviously, there is a need to maintain meta-information, explaining what the perspectives are about.

Interference versus non-interference The common basis for the collaborators is the primary model, for instance, embodied by a simulation. Several derived models can depend on the primary model, and derived model could be related to a number of presentation models. When collaborating, the common basis should be the same for all the cooperators at every moment in time. To assure consistency, it is best to have the simulation act autonomously without the slightest interference. We will refer to this as *non-interference*. Non-interference does not restrict the possibility for each user to create his own perspective in any way. It does prevent somebody from rewinding, changing parameters or restarting the simulation while others do not want or expect this.

However, the need to stop, rewind or change parameters in a simulation is imminent. Considering multiple *what-if* situations, for example, is necessary when looking at different re-design alternatives, each with its own set of parameters. One way to meet this requirement, while upholding non-interference, is to store all past events in a database or to allow copies with different parameters of the simulation to be started.

While interaction with the primary model should be avoided as much as possible, derived models can be used in a more flexible manner. Several derived models can be created, all depending on one primary model. While the primary model can be considered a common basis that should not be interfered with, the derived model can be seen as a common workspace that permits *interference*. All participants of

a session could use the same derived model or multiple derived models could be created, depending on the need to share information concepts or to be independent of the other users.

Communications Some form of user-to-user communication is necessary to enable collaboration. These communications can range from a simple chat tool or whiteboard to sophisticated audio/video conferencing tools.

Tools of interest for use with DIVA include telepointers, to point out things of interest, raising hands, to indicate someone wants to speak, and voting tools, to support decision making (Ellis et al. 1991).

4.2 Requirements

Taking into account the issues for collaborative visualization mentioned in the previous section, we can summarize the following requirements.

- Session management is needed to control the virtual meetings, including the participants, their roles and interaction protocols.
- The participants must be able to share their perspectives by enforcement and via perspective repositories.
- The primary model should be interfered with as little as possible.
- Additional communication support is necessary, but falls outside the scope of this paper.

5 COLLABORATIVE MULTI-USER VISUALIZATION ARCHITECTURE

The DIVA architecture is intended as a *framework* and should be flexible enough to incorporate new components. The components used in DIVA are generic software components which interact with each other. The architecture is distributed, meaning that its components can reside on different hosts on the network. Related collaborative architectures can be found in Bentley, Rodden, Sawyer & Sommerville (1994) and Reinhard, Schweizer & Völksen (1994).

5.1 Software components

Figure 2 shows the main components of the architecture. In the following list, the three components that were already present in an earlier version of the DIVA architecture (Schönhage & Eliëns 1997, Schönhage & Eliëns 1998), are listed first. The last three components in the list extend DIVA.

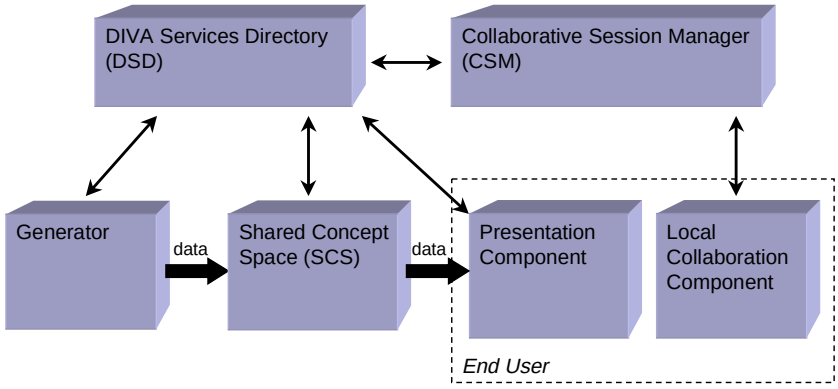

Figure 2 The main components of DIVA

- generator — embodies the primary model
- shared concept space — information store that contains the derived model
- presentation component — the actual information visualization
- DIVA services directory — central registering of components
- collaborative session manager — overall coordination of virtual meetings
- local collaboration component — local collaboration support

In a normal situation, one DIVA services directory and more than one of each of the other components could exist. A presentation component and a local collaboration component are present at the desktop of each participant. A short description is given for each of the components.

The *generator* embodies the primary model and generates all data needed for the visualization. Examples of generators are simulations of business processes. The generated raw data is transferred to the shared concept space.

The *shared concept space* stores information in an expressive and adaptive way. The information is contained in the form of concepts that are stored in a hierarchical manner. Each concept has one or more data properties that represent the information. The data properties are updated with data coming from the generator. The received data can be stored directly or can be used to compute and store derived information.

The *presentation component* actually visualizes concepts from the shared concept space. It makes use of gadgets, which are generic visualization primitives that present certain types of information. As an example, cone trees are primitives (gadgets) to visualize hierarchical information (Robertson, Card & Mackinlay 1993).

Examples of gadgets are a rotating object which indicates a certain action or a histogram which displays data.

The DIVA *services directory (DSD)* is the central directory component. DIVA components (or services) can register here, identifying themselves and giving their location. Once they are registered, the DSD can inform other objects about the availability and whereabouts of these services.

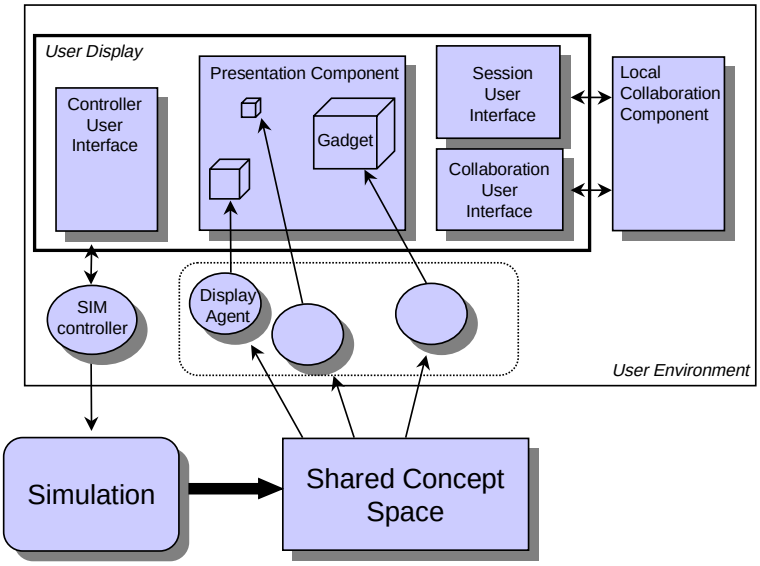

Figure 3 The user environment

The *collaboration session manager (CSM)* coordinates components. It deals with interaction protocols, which means it knows about the participants and their roles, sharing perspectives, user to user communication and consistency.

The *local collaboration component* is directly connected to the session manager. It is present at each participants desktop and handles interactions and information related to a collaborative session. It may for example display a list of participants and offer communication facilities.

5.2 User Environment

Figure 3 shows a typical user environment. Outside of the user environment, two components are shown. A generator, which is a simulation in this figure, feeds data into the shared concept space. This is depicted by the fat arrow. These two components can be situated anywhere on the network. Most of the components in a user environment use the display. The local collaboration component displays information about the collaborative session. The presentation component displays the visualization and the controller displays its user interface.

Display agents From the shared concept space, information is being sent to *display agents*. These agents are present at the user environment and each of them maintains one gadget. The information they receive is transformed into a visualization. As an example, consider the visualization in Figure 4 on page 12. The line of three

puppets in front of the desk is a visualization gadget depicting a queue. When the display agent senses that the length of the queue increases by one, it accordingly places a fourth puppet on the screen.

The display agents also reside in the perspective repository. When a user requests a certain view, the agents that represent that view are cloned and moved to the user environment to build the perspective in the VRML world. Enforcing a perspective onto another user is accomplished by cloning and moving the display agents from one user to another.

Now why can we call these entities agents? There are a lot of definitions of agents (Franklin & Graesser 1996), and the question can be raised why a display agent is an agent. In other words, how does it differ from a standard program or software component? For one, the agents are autonomous, which means they execute on their own. Second, they react on certain input and subsequently act on their environment, namely, they sense information from the shared concept space and act on the VRML world. They have some goals they need to accomplish. Third, the display agents can communicate with each other, for instance, to discuss how to place the gadgets each of them represents on the screen (this is a future feature). Fourth, they have a domain which they have knowledge of. This domain is the visualization of information. Fifth, they act on behalf of a user. Users can give their preferences to a display agent, and the agent will take care of it. All in all, the display agent fits quite a number of definitions of autonomous agents given by Franklin & Graesser (1996).

Controllers Every DIVA component can have a separate mobile *controller*. It can be moved from one environment to another, so it can be shared by several participants. The ability to use a controller depends on the role of the user. Participants can request a controller or the chair could appoint it to one of them.

Controllers can have several functions. For example, a simulation can be controlled by starting and stopping the simulation and changing certain parameters. A controller for a shared concept space can be used to create new computed concepts or to decide which data from the generator is selected.

5.3 CORBA and the Web

DIVA is designed as a distributed object oriented system. The DIVA components are written in C++ and Java, and can run on different platforms. We use the Common Object Request Broker Architecture (CORBA) to let our distributed objects communicate with each other. CORBA (Siegel 1996, Orfali & Harkey 1997) abstracts from hardware, operating systems and programming languages. By using the interface definition language (IDL) to describe the interfaces between components and by making use of the object request broker (ORB), distributed components are able to communicate.

Voyager (ObjectSpace 1997) is an agent ORB written purely in Java, which supports CORBA. Voyager allows us to use mobile objects, a feature which CORBA

does not have. We use Voyager to construct the mobile controller components. These components are able to ”dock” at a user environment and can subsequently show their user interface on the screen to let the user interact with it.

We use VRML (ISO 1997) as the main visualization tool. The users are able to navigate through the VRML worlds by using a VRML-browser. The External Authoring Interface (EAI) makes it possible to control the VRML worlds dynamically via the Java and Javascript languages.

The visualization gadgets in the presentation component are represented by mobile display agents. These agents are constructed using Voyager. Display agents can also ”dock” in a user environment and, in addition, get access to the local VRML world. They collect the needed information from shared concept spaces to build and maintain the 3D visualization.

The combination of CORBA and the Web enables access to information resources by means of HTML, Java and VRML (see also Rohrer & Swing 1997). For example, the simulation and shared concept space can be hosted on a Unix server while the presentation components are executed in a Web-browser on Windows client machines.

6 APPLICATION

Figure 4 presents a screenshot of the desktop of a decision maker participating in a collaborative session as described in the example of Section 2. We describe a scenario of how the user gets to this display.

The decision maker starts a Java and VRML enabled Web-browser and follows a link pointing to a DIVA server. The resulting HTML file will setup the user environment. First, the user has to log in, making available her name and network address, and after that she can choose from one or more sessions to join. Once the user enters a session, she will be assigned a role and then gets a default or enforced perspective. The VRML world showing her view is embedded in the Web page. Two Java applets will contain the session interface and the collaboration interface (these are not shown in the screenshot).

With a push on a button she is able to request a remote control, which will arrive at her environment (assuming that she is allowed to do so). Once the remote control arrives in the form of a mobile object, this object pops up a remote control user interface on the display. The user is then able to control the simulation that is associated with the remote control. In Figure 4 the remote control can be used to run, stop and reset the simulation. In addition, the speed of the simulation as well as three parameters can be changed. Changes to the simulation will accordingly appear in the visualization in the browser window.

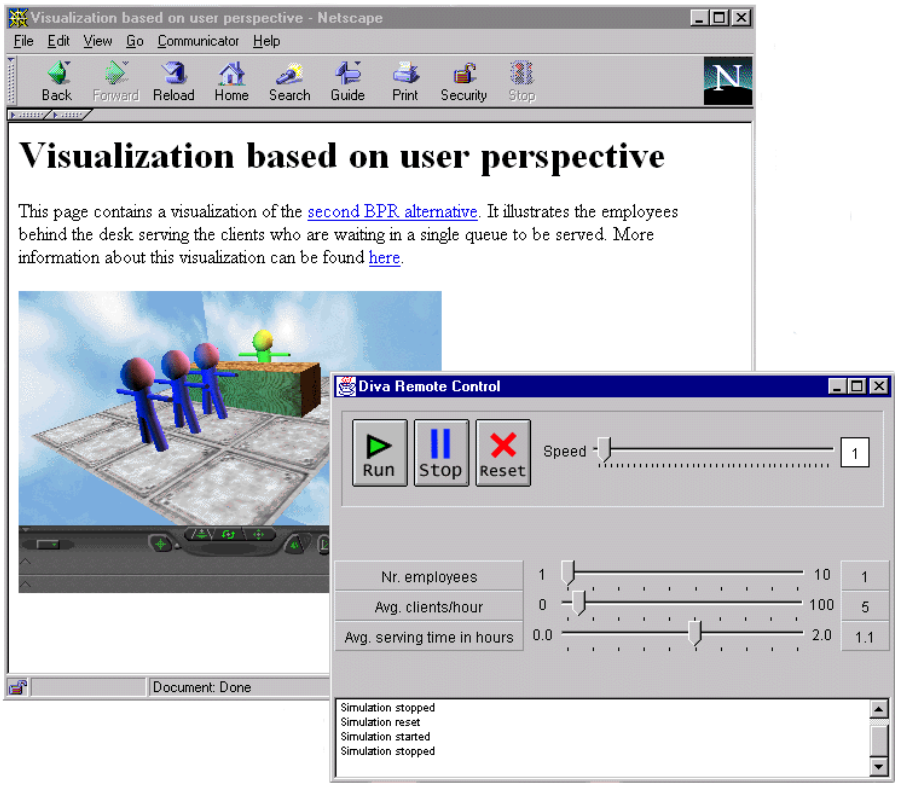

Figure 4 A perspective with a remote control

7 CONCLUSIONS

This paper is based on our belief that (interactive) visualizations are useful arguments in decision making because they provide such quick insight into information by using the human perceptual abilities. By means of collaborative visualization, decision makers are able to discuss a shared information source, such as a business process simulation, to convince other users of their point of view.

Based on a discussion of issues in collaborative visualization, we have concluded that the following requirements are needed for our visualization architecture.

First, different perspectives are necessary because multiple users, with different information needs, require different views on the data. These perspectives can be created by means of shared concept spaces and presentation components that make use of display agents.

Consequently, an important requirement is the exchange of visualization perspectives, for example, by enforcement or by a repository of perspectives. The exchange

is achieved by cloning and transporting display agents, which in turn define how and what available information is presented to the users.

To manage the cooperative sessions, we have defined two collaboration components that handle the rights and obligations belonging to the roles of the participants.

As a final requirement, we have stated that interference should be avoided as much as possible because other participants are involved in the actions taken. On the other hand, interaction with a running simulation to evaluate some *what-if* situations is very powerful. Therefore, we have created remote control components to interact with simulations and other data generators.

In further research, we are investigating an extension of the shared concept space to store sessions that can be replayed at a later time. Additionally, we are planning a case study to determine for useful visualization primitives to represent business information.

REFERENCES

Bentley, R., Rodden, T., Sawyer, P. & Sommerville, I. (1994), 'Architectural support for cooperative multiuser interfaces', *IEEE Computer* **27**(5), 37–46.

Eliëns, A., Niessink, F., Schönhage, S., van Ossenbruggen, J. & Nash, P. (1996), Support for Business Process Redesign: Simulation, Hypermedia and the Web, *in* 'Euromedia 96: Telematics in a Multimedia Environment, London, United Kingdom', The Society for Computer Simulation International, pp. 193–200.

Ellis, C., Gibbs, S. & Rein, G. (1991), 'Groupware: some issues and experiences', *Communications of the ACM* **34**(1), 680–689.

Franklin, S. & Graesser, A. (1996), Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, *in* 'Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages'.
URL: *www.msci.memphis.edu/∼franklin/AgentProg.html*

ISO (1997), *The Virtual Reality Modeling Language*. International Standard ISO/IEC IS 14772-1:1997.

ObjectSpace (1997), *Voyager Core Technology User Guide (Version 2.0 beta1)*.
URL: *www.objectspace.com/voyager/documentation.html*

Orfali, R. & Harkey, D. (1997), *Client Server Programming with JAVA and CORBA*, John Wiley & Sons.

Reinhard, W., Schweizer, J. & Völksen, G. (1994), 'CSCW Tools: concepts and architectures', *IEEE Computer* **27**(5), 28–36.

Robertson, G., Card, S. & Mackinlay, J. (1993), 'Information Visualization using 3D Interactive Animation', *Communications of the ACM* **36**(4), 57–71.

Rohrer, R. & Swing, E. (1997), 'Web-based Information Visualization', *IEEE Computer Graphics & Applications* **17**(4), 52–59.

Schönhage, S. & Eliëns, A. (1997), A flexible architecture for user-adaptable visualization, *in* D. S. Ebert & C. K. Nicholas, eds, 'Workshop on New Paradigms in Information Visualization and Manipulation '97, Conference on Information and Knowledge Management, 10 - 14 November 1997, Las Vegas,

USA', ACM Press.

Schönhage, S. & Eliëns, A. (1998), Multi-user Visualization: a CORBA/Web-based approach, *in* 'Proceedings of "Digital Convergence: the Future of the Internet and WWW", 20-23 April 1998, Bradford, United Kingdom', British Computer Society.

Shneiderman, B. (1998), *Designing the User-Interface, Strategies for Effective Human-Computer Interaction*, 3rd edn, Addison-Wesley Publishing Company.

Siegel, J. (1996), *CORBA Fundamentals and Programming*, John Wiley & Sons.

8 BIOGRAPHY

Bastiaan Schönhage is PhD-student at the Vrije Universiteit in Amsterdam and ASZ Research & Development. His research comprises the design and exploration of a flexible software architecture for dynamic information visualization on the Web (DIVA). His research interests include information visualization, and distributed OO software architectures.

Peter Paul Bakker completed his master thesis on the collaborative aspects of the DIVA-project. He is a pre-graduate student in Software Engineering at the Vrije Universiteit and he is doing an internship at ASZ R&D. His research interests include collaboration, VRML and agent-based ORBs.

Anton Eliëns is lecturer at the Software Engineering section of the Computer Science Department of the Vrije Universiteit, Amsterdam. His research interests include object orientation, hypermedia and distributed logic programming.