Chapter 2 CVE Critique: Technology, Design and Usability

> "Communication systems shape social organisations because they structure temporal and spatial relationships."
>
> Harold Innis, precursor of McLuhan, in Mattellart and Mattelart, (1998).

Chapter 2 CVE Critique: Technology, Design and Usability

This chapter consists of four distinct sections that critically review CVE technology, design, usability, and a number of existing CVEs respectively.

2.1 Introduction

This chapter places CVE technology, design, usability and existing CVEs, in context within the larger fields of Information Technology (IT), computer science design practise, human computer interface (HCI) design practise, and commercially available CVEs. It is by no means exhaustive in its description of all aspects of these fields, but rather considers in pertinent detail each of the issues that have an impact on the usability of CVEs. This approach has been adopted in order to clearly identify the state-of-the-art limitations of networked, graphical, communication technology, design approaches, usability evaluation approaches, and available CVEs, at the time this thesis was produced. Furthermore, it describes and discusses the ramifications and consequences of these limitations on the design of collaboration support for CVEs. The aim of this chapter is to identify the issues that are relevant to the support of real-time collaboration over distance as mediated by CVEs, describe the types of collaboration that are supported, identify the design choices that are shape CVEs, describe the evaluation issues that exist for CVEs, and discuss the effect of these limitations on the usability of CVE technology.

The next section presents the general field of IT, and places CVE technology in it (2.2), followed by a section that describes design practise and how this influences CVE design (2.3), a section that describes usability design practise and how this

impacts on usability design for CVEs (2.4), and a section that describes a number of existing CVEs (2.5). This is followed by some conclusions (2.6), which highlight and summarize the state-of-the-art limitations of CVE technology.

2.2 CVE Technology in Context

This section places CVE technology in context within the larger field of Information Technology (IT). It is by no means exhaustive in its description of all communication technologies, but rather considers in pertinent detail each of the headings and sub-headings under which CVE technology falls. This approach has been adopted in order to clearly identify the state-of-the-art limitations of networked, graphical, audio and video communication technology on the development of CVE technology, at the time this thesis was produced. Furthermore, it describes and discusses the ramifications and consequences of the computational bottlenecks on the design of collaboration support for CVEs. The aim of this section is to identify CSCW technologies that support real-time collaboration over distance, describe the types of collaboration that are supported, compare their success, and discuss the effect of the technical limitations on the usability of these communication technologies in general and CVE technology specifically.

The next section presents the general field of IT and places CVE technology in context (2.2.1), followed a section that describes other fields under which CVE technology may be classified (2.2.2). This is followed a section that describes and discusses the state-of-the-art limitations of networked, graphical, audio and video communication technology where they create bottlenecks on the design of collaboration support for CVEs (2.2.3). The discussion evaluates the previous sections

in order to summarise the technological framework that demarcates the issues under research in this study (2.2.4).

2.2.1 CVE Technology in Information Technology Context

CVE technology can be said to be part of the group of technologies called Information Technology (IT). The rapid development of IT is one of the focal points in the European research and development (R&D) market, and indeed in the world. The concept of the 'Information Society' has been officially formulated by the European commission to address issues of concern, such as the social implications of the large-scale deployment of new information and communication technologies (Chester, 1998). To this purpose, VR technology is specifically singled out (along with 'agent technology' - intelligent software that assists people and can act on their behalf) as an area for the pressing need for research and debate on issues of morality, privacy, personal identity, and possible fundamental changes to what makes up a human society. Obviously, the range of topics involved in such research and debate is too wide to be encompassed in this thesis, but where possible the author has pointed out issues raising from the research reported here, which could be pertinent to the concerns of the European Commission.

Within the general field of IT exists a group of technologies called Computer Mediated Communication (CMC). CMC addresses the use of an application computer to control interactive media and message-based communication to provide more effective ways of doing things. Whether or not a new technology provides better ways of doing things depends on an understanding of the old work practice, an understanding of the new work practice, and an understanding of how the new technology shapes and supports the new work practice. Such understanding does not

evolve overnight, but rather grows through multi-disciplinary debate and research, involving computer scientists (hardware and software engineers), sociologists, psychologists, ergonomists, artists and philosophers, users of the new technology, etc. CVE technology is specifically put forward as a means to improve our work practices. Firstly, because it supports real-time interaction over distance, which could reduce the need to physically travel to meetings. Secondly, it supports both formal and informal interaction within the same application, which should mimic the settings of a real meeting more closely than other teleconferencing technologies can provide. Thirdly, users can control their own viewpoint in the virtual world instead of having a static view. Finally, CVE users can directly interact on shared documents inside the same virtual world, thus mimicking real world settings for collaborative work through shared artefacts more closely than other teleconferencing technologies. However, it is by no means clear yet how well CVE technology provides these improvements, whether it allows sufficient and satisfactory collaboration over distance, nor how it influences and changes our work practice. This thesis addresses these issues from both a psychological and software engineering angle, in order to explore subjective user satisfaction and objective task improvements.

CVE technology is also part of a group of technologies called Computer Supported Cooperative Work (CSCW) technology. CSCW applications aim to support groups of users in their work as a group using one or more digitalized media as the communication platform, by taking into account all work patterns and the effect of the technology on the workplace, workflow, and the user and user-group as a whole within the workplace (Beaudouin-Lafon, 1999). There are notable exceptions to this classification; for instance, some multi-user VEs have been developed to support

collaborative and competitive gaming instead of work. However, it can be argued that even these types of VEs have to include support for interaction and coordination between users. There are two reasons why it is useful to deal with CVE technology as a particular instance of CSCW technology. Firstly, pigeonholing CVE technology within the larger field of CSCW should allow us to apply the CSCW knowledge base to the task of understanding collaboration for CVEs, because it is precisely the support for cooperative work in CVEs that is still poorly understood. And secondly, the technical underpinnings of both technologies are shaped by the same requirements: supporting distributed users to perform collaborative tasks through an understanding of the visible and invisible work practices.

CSCW applications can support three different general types of activities: communication, cooperation, and coordination (see table 2.1). Depending on the design of the application it can support one or more of these activities. Sometimes, support for these activities is built implicitly into more pragmatic applications, such as digital libraries, electronic commerce, knowledge management and distance learning, which are said to include CSCW principles within a larger context (Dix, et al., 1998). The same is true for CVE applications, which in principle can support all these activities and more, but only if the CVE has been designed to do so. In all cases, the development process for these new technologies should follow the same software engineering structure (requirements gathering, design, evaluation and deployment). In particular, the types of activities that need to be supported, the kind of work process into which they are to be incorporated, and the effect the new technology has on the work-flow, has to be carefully made more explicit through debate, research and development.

Type of Activity	Type of Support
Communication	Direct interpersonal communication by electronic mail, voice mail, audio conferencing, video conferencing and desktop conferencing.
Cooperation	Ideas generation and decision-making by means of message databases, document databases and meeting facilitation.
Coordination	Sharing computer objects to control business process modelling, workflow management, project management, calendaring, and scheduling.

Table 2.1: CSCW applications can support three different general types of activities: communication, cooperation, and coordination.

2.2.2 Groupware Technology

One major area in CSCW is the design of computer applications to support group working. These systems are often called groupware. Groupware is software designed to support the collaboration of several users. Groupware systems usually involve several receivers/senders connected by a network. Groupware technology can be divided into asynchronous and synchronous applications (see figure 2.1). Asynchronous systems facilitate delayed interaction on shared objects. Asynchronous groupware includes email, bulletin boards, non real time database sharing, fax and postal services. Synchronous systems allow remote users to interact at the same time. Synchronous groupware includes audio conferencing, video conferencing, desktop conferencing, and media spaces.

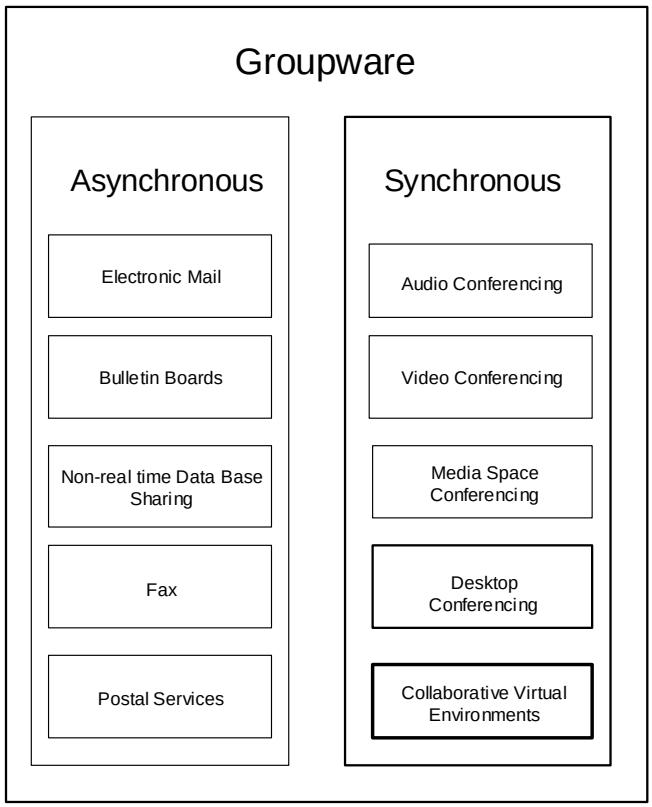


Figure 2.1: CVE technology supports synchronous interaction between users.

To be precise, groupware technology can be classified according to the location and timing of the collaborative activity:

- Where it happens: groupware users can be collocated (same place) or remote (other place).
- When it happens: groupware users can be using the system synchronously (at the same time) or asynchronously (at different times).

This classification can be made more obvious by using the time/space matrix illustration (see figure 2.2). The space axis of the time/space matrix refers to

geographical location of the users, and the time axis refers to whether or not the users' collaboration takes place at the same time.

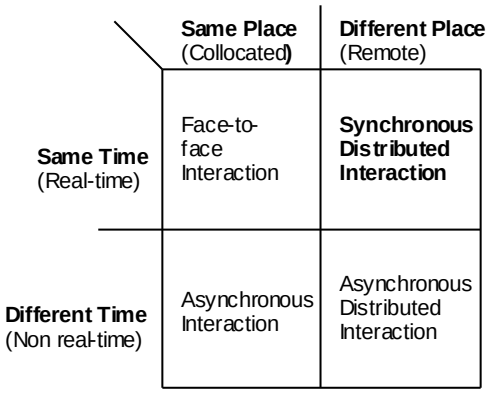


Figure 2.2: Time/space matrix. (adapted from Dix et al, 1998).

CVE applications specifically aim to support collaboration between geographically remote users, and thus belong to the 'Synchronous Distributed Interaction' cell in the time/space matrix. Although it could be argued that, theoretically, CVE users could be collocated, in practice this gives all kinds of problems because users sitting next to each other can hear each other in the real room before they can hear each other in the virtual space, and problems of auditory feedback, echo's and delay occur in such settings.

In addition to CVEs there are other types of systems which support synchronous distributed interaction. These include: several types of conferencing systems, such as audio conferencing, video conferencing, desktop conferencing, media spaces, where the primary focus is on communication; group editors, where the primary focus of the application is on shared document editing; shared drawing surfaces, where the

primary focus is on collaborative remote design; and group decision support systems, where the primary focus is on generation and recording of ideas and decisions.

Cooperation between synchronous distributed users takes place via the application and its success depends entirely upon the control and feedback it provides. Participants communicate with one another as they work and may need to use tools and work objects to support their collaboration. Users need to build and maintain a common awareness and understanding of the participants involved in the cooperation and the shared objects on which they work. These objects are the 'artefacts of work' and they constitute an important part of the collaboration process. The application needs to help the participants to identify who is communicating to whom, and in the case of shared artefacts how to control them and who is controlling the artefact at any particular time.

CVEs are different from other synchronous distributed systems in that users are generally represented by a virtual embodiment which the user controls, instead of by a cursor or pointer. This embodiment makes visible their interaction on shared objects and direct communication with other users within a 'realistic' shared virtual space, which becomes their common medium of communication.

2.3.1 Synchronous Distributed Groupware Technology

Synchronous groupware technologies aim to allow multiple users to collaborate together in real-time on a certain task. Several distinctive kinds of synchronous groupware can be defined:

- Audio conferencing: voice only, group telephone lines.

- Video conferencing: visual and voice, group video lines.
- Media space conferencing: visual, voice, shared documents, group video lines, multiple viewpoints.
- Desktop conferencing: simple text, visual, voice and shared documents.
- Virtual conferencing (CVE technology): simple text, voice, computer generated collaborative virtual spaces, virtual embodiments, shared documents, multiple viewpoints.

Workplace studies have shown that communication and collaboration are dependent on the ability of personnel to use and refer to aspects of shared objects (Goodwin and Goodwin, 1996; Hindmarch and Heath, 2000a). Indeed there seems to be a trend to increasingly provide workers with enhanced access to each other' spaces and include objects for sharing during collaboration. (Gaver, et al, 1993; Kuzuoka, Kosuge, and Tanaka, 1994; Tang, Isaacs, and Rua, 1994; Heath et al, 1995, Benford et al, 1997, Reynard et al, 1998). Audio conferencing, groups working over telephone systems, is notoriously difficult to manage (Walters, 1995). In principle, large audio conferences can be established, but it becomes difficult to ascertain who is talking when more than four or five simultaneous users are involved. Video conferencing has been a natural extension of audio conferencing, allowing for the inclusion of facial expressions and a limited set of gestures to be part of the group interaction. However, for video conferencing, dialogues have been found to be significantly longer, with more interruptions, than for audio conferencing, particularly when transmission is delayed (O'Malley, Langton, Anderson, Doherty-Sneddon and Bruce, 1996). Also, sharing documents is still a problem because participants could not see when and where in the document was being pointed at (Heath and Luff, 1991). Research has shown that it is

important for participants involved in group work to be able to establish a general awareness of what others are doing, especially while they are working on individual focussed tasks (Hughes, et al., 1992). To provide support for this kind of peripheral awareness more video cameras have been added, this has led to the birth of yet another type of conferencing technology, referred to as media space technology. Media space technology is essentially video conferencing with additional video cameras aimed at documents and workspaces, which makes it possible to include shared document views, and give an increased awareness of the video conference participants' background to the other participants (Gaver et al., 1993). However, participants still have difficulties working together because the separate, fixed cameras leave gaps in the view offered of the remote space, which means that gaze direction, and gaze awareness which play a vital role in 'normal' communication are not sufficiently supported. For instance, participants found it difficult to ascertain which aspects of their own activities and workspace were visible to their colleagues (Heath, Luff and Sellen, 1995).

Desktop conferencing refers to systems that allow all users to synchronously interact with each other via their personal computer. It allows them to share information in one or more forms, from simple text, documents, audio, facial expressions and a limited set of gestures to simultaneous editing of visually displayed data. The exchange of simple text messages is achieved through the use of electronic chat programs, which provide typed, real-time communication between two or more people. All messages are recorded and displayed to all chat participants simultaneously. However, because it is possible for all participants to type at the same time, discussions often become confusing. The more participants that are actively

involved in the discussion, the harder it becomes to keep track of who is replying to whom and to check whether one has understood the messages as intended.

Desktop conferencing systems often include shared applications and artefacts, such as editors and drawing programs, which allow a team to display a common document and simultaneously work on the contents. The keystrokes and mouse movements done on one screen appear on the screens of all other participants. However, these systems have inherent problems in that the system may not include a direct communication channel between the participants. Also, it may be hard to attain a common understanding of what exactly is being looked at or referred to because participants may be looking at different parts of the document and refer to views the others may not be sharing at that moment. Some desktop conferencing systems provide video and audio links between all participants, who are all displayed in small windows on the screen and they are all able to talk to each other. However, sending full motion video information to multiple participants requires high bandwidth connections, which may not be available, and for each added user more bandwidth is required. Additionally, it is difficult to display more than a dozen video windows on a single desktop computer screen, thus effectively limiting the number of simultaneous users.

Virtual conferencing involves the use of CVE technology to create a 3D computer generated graphical multi-user meeting space. Sometimes virtual conferencing is referred to as virtual desktop conferencing, or even simply desktop conferencing, because some CVEs are developed for desktop use only, as opposed to immersively through the use of a head mounted display. Throughout this thesis ‘CVE technology’ and ‘virtual conferencing’ refer to desktop CVEs only, unless otherwise stated.

However, virtual conferencing is different from desktop conferencing in that all human-human interaction takes place inside a single application, whereas a desktop conferencing system can often consist of several different applications used simultaneously that are all controlled individually, from the computer desktop. However, sometimes CVEs make use of external applications to aid and supplement the collaboration support provided by the CVE, which makes it difficult to use as an immersive application, because head mounted displays have such low resolution that normal text, for example, is often difficult to read.

CVEs provide a shared environment, through which users can navigate and thus adjust and fine tune their own viewpoint. Their movements and gaze direction are visible to the other users, thus allowing cues for peripheral awareness to each other (Benford et.al, 1995). Finally, users should in principle be able to refer to objects spatially, thus allowing them to refer to shared objects quite naturally. In these respects CVEs provide advantages over video conferencing and media space systems. However, at present CVEs do suffer from some serious limitations. Desktop CVE systems generally offer a small field of view (50 degrees) compared to the size of our real field of view (120 degrees), thus making support for peripheral awareness more complicated than it would seem (Hindmarch et.al, 1998). Real world information, such as users' real gaze direction and real facial expressions, are not automatically displayed in the virtual world, thus losing the richness of interaction that video conferencing allows. Finally, the virtual world, the objects in it and the virtual embodiments are at present depicted using crude graphical representations with often limited functionality, in order to keep the computational demands of the CVE on the network as low as possible and the number of users it can support simultaneously as

high as possible. These scalability issues have repercussions on the usability of the CVE.

Mediating Technology	**Definition**	**Known Problems**
Audio conferencing	Group working over telephone systems	Difficult to ascertain who is talking. No way to share documents in real time.
Video conferencing	Group working via video connections. Inclusion of facial expressions and a limited set of gestures.	Dialogues significantly longer more interruptions. Problem sharing documents because of lack of detail.
Media space conferencing	Video conferencing with additional video cameras aimed at documents and workspaces.	Fixed cameras leave gaps in the view of the remote space. Difficult to make sense of colleagues' conduct.
Desktop conferencing	Group working via a hybrid of technologies incorporated into one or more software applications that the users can run from their computer desktops.	Users cannot control remote video cameras. Difficult to deal with multiple windows. Variable delays on interaction in the different applications. Users cannot point from one application to another because this is not visible to other users.
Virtual Conferencing *(CVE technology)*	Group working via a shared computer generated graphical space with avatars representing participants.	Field of view limited. Object interaction and navigation clumsy. Limited set of gestures. Lack of facial expressions.

Table 2.2: Historical overview of remote conferencing technologies.

In summary, all CSCW technologies discussed above suffer from limitations in the support for collaboration introduced by the technology itself. Hindmarch (1997) argues that the relative weakness of these systems to support synchronous remote

groupware derives from their inability to assist individuals in working flexibly with documents, models and other workplace artefacts. Table 2.2 summarizes the synchronous distributed technologies and their known usability problems. Now follows a section that identifies and discusses the technological bottlenecks imposed on synchronous distributed technologies in general and CVE technology in particular, in order to clearly point out the limitations they cause.

2.2.3 Bottle necks for distributed synchronous groupware

Synchronous groupware is particularly difficult to design due to limited bandwidth and delays on the network used to connect the receivers and senders (Chambers, Duce, and Jones, 1984). The response time between an action of a user and the reception of the action to all users runs the danger of being too long or too variable to be usable. The number of total simultaneous users runs the danger of being too high to be usable, and must be monitored to avoid problems.

Multi-user systems are more fragile because of the large numbers of hardware, software, and human operator components. The complexity of the algorithms increases with each added user, making groupware highly error prone. A single failure can propagate throughout the whole system. Interleaving of different user's actions associated with the unpredictable effects of network delays can easily create system errors and confusion amongst the users. Many faults will only be discovered when more users than predicted are using the system in ways that were not anticipated.

The limitations of currently available networked, graphical, audio and video communication technology that impact on the development of CVE technology (c.f. Macedonia and Zyda, 1997) are discussed below:

- Computational consistency problems (2.2.3.1);
- Concurrency control problems (2.2.3.2);
- Scalability problems (2.2.3.3);
- Robustness problems (2.2.3.4);
- Network delay problems (2.2.3.5).

These problems are frequently mentioned in CVE research and development literature (Greenhalgh, 1999), and cause considerable trouble for CVE developers, CVE usability testers, and CVE users. As such they are of great importance, because they shape and limit the design space and design choices for CVEs.

2.2.3.1 Computational Consistency Problems

Consistency maintenance refers to the requirement that multiple clients should be enabled to concurrently access and change shared data in the shared application, and the fact that all other clients have to be informed of these changes immediately, and in the same sequence as they occur, so that consistency of the shared data is maintained between all clients. There are several important types of consistency, such as update consistency, replication consistency, cache consistency, failure consistency, clock consistency, and user interface consistency. Consistency problems in CVEs occur when information about the positions and actions of user's virtual embodiments arrive in a different order, or with an unacceptable delay at other users computers. Typical

solutions for consistency maintenance are caching, and the employment of hypotheses of locality in the pattern of user references to data. Caching refers to a mechanism, implemented by software in the client computer, for the retention in the client's environment of a copy of the data values for subsequent reuse, avoiding the need to request them again when the same resource is accessed subsequently. Hypotheses of locality are used to define locality models, which attempt to predict the minimum amount of data that needs to be updated in order to maintain consistency for a particular user, with regard to the activities and position of that user.

2.2.3.2 Concurrency Control Problems

Typically, users of collaborative applications can input and output data concurrently. Problems occur when two or more users simultaneously access and modify the same data, because this would create a conflict about the actual state of the data. Therefore, the application must have concurrency control mechanisms to stop the different user's actions from interfering with that of others. Typical concurrency control mechanisms are locking, floor control, and to some extent social protocols. Locking refers to the principle that when one user starts to modify a certain data-set, no other user is allowed to modify the same data-set at the same time. Floor control policies determine which user can modify a data-set at any moment, and social protocol generally develops when users gain experience with negotiating control over data-sets, especially when locking or floor control mechanisms cannot be used. For concurrency control mechanisms to work effectively, the application has to be decomposed into all possible threads of action invoked by different users. Subsequently all threads which can potentially be executed concurrently need to be identified and enabled, while all others need to be locked. This is called the maximal-concurrent approach, and it is a

tedious and error prone task, which is also highly application dependent. As a result different architectures may take different approaches to concurrency control, with different final degrees of concurrency.

2.2.3.3 Scalability Problems

Scalability refers to the continued ability of the application to service its users while the number of users and/or the user activity increases. Scalability problems typically occur because with the addition of each new user to the system, the amount of data that needs to be sent to all users increases, thus rapidly creating larger and larger processing, bandwidth and memory demands. Scalability problems can only be detected when system development and testing includes a carefully controlled increase in the number of users and user activities. Solutions to scalability problems for CVEs include the notion that users need only be updated about changes in the virtual environment that are in the scope of their awareness and/or interaction, thus decreasing the amount of data that has to be send to all users. Other solutions include decreasing the level of detail with which the virtual environment and the objects in it appear to the user, based on the distance between them and the user, which lowers the amount of data that has to be sent to each user.

2.2.3.4 Robustness Problems

Robustness refers to the ability of the application to handle computational errors and network delays without crashing. Four potential sources of problems are: failures in the network, workstation or operating system; errors in programming the shared application; unforeseen sequences of events, inability of the system to scale as the number of users or rate of activity increases (Dix, et al., 1998). There are factors that

make multi-user systems more fragile. For instance, multi-user systems typically employ a larger number of hardware and software components than single-user systems, the algorithms used for groupware are more complicated, and network delays can have unpredictable effects on the applications. Errors may not become apparent until the application is used by large amounts of users in realistic settings, and the consequences of a failure can propagate throughout the whole system, destroying the work of a whole group of people. The typical solutions for robustness problems are too far into the field of computer science to be of relevance here, but perhaps needless to say, the robustness of an application needs to be systematically and thoroughly tested before commencing any usability test in order to avoid running into high costs and frustration due to the number of experimental subjects which have to be kept standing by and coordinated until the system is up and running again.

2.2.3.5 Network Delay Problems

Groupware systems generally involve several computers connected by a network. The results of actions of one user have to be fed back to that user and fed through to the other users. This feedback and feedthrough includes transmission over the network. Network traffic is subject to unpredictable delays, because the available network bandwidth is shared by multiple users. Limited bandwidth and delays of the network can cause unacceptable delays on feedback and feedthrough of user actions. Unless the input of a user is processed and the changes transmitted quickly enough to give the user the impression of an instantaneous change, the user becomes aware of the 'lag' or network delay. Measurements show that a delay of less than 0.1 second must be achieved in order to produce the impression of continuous interaction when using modern graphical interfaces (Coulouris et al, 1995). When editing text, a delay of

more than a fraction of a second between typing and the appearance of characters is unacceptable (Dix et al, 1998). Rapid feedback to the user who initiated an action is necessary, but the feedthrough to the other users also suffers from lag. Lags on feedthrough of more than a few seconds can be disastrous for conversations. A delay of 200 msec on the auditory feedback of the user's own speech creates impossible problems for the speaker (van Leyden Sr., 1984). The typical solutions for network delay problems in CVEs are to make use of a combination of peer-to-peer communication between the machines of each user, client-server communication that distributes the computational tasks over several machines, and multi-server architecture that allows a number of servers to cooperate to accommodate a larger number of clients. A more detailed description of these solutions will carry too far into the field of computer science to be of relevance here, but clearly any design for CVEs must take potential network delays very seriously.

2.2.4 Discussion

The computational bottlenecks for CVE technology have been identified above. A summary is provided in table 2.3. The main focus of the discussion has been on how these bottlenecks affect application behaviour and ultimately the users of CVE systems.

Technological Challenges	**Definition**	**Known Solutions**
Computational consistency problems	Information about the positions and actions of CVE users arrive delayed, and or in a different order at other user's computers.	Caching; locality of reference
Concurrency control	Two or more users simultaneously	Locking, floor control, social

problems	access and modify the same data, creating computational conflicts about the actual state of the data	protocol
Scalability problems	The amount of data that needs to be send to all users demands more processing power, bandwidth and/or memory than available.	Take future scaling of the system into account at the beginning of the project. Scoping of awareness and interaction. Level of detail management.
Robustness problems	Crashing of the whole system due to inability of the application to handle computational errors and network delays.	Employ modular and defensive programming. Logging of application communications to trace and recreate errors. Systematic and thorough testing before commencing any usability test.
Network delay problems	Limited bandwidth and delays of the network cause unacceptable delays on feedback and feedthrough of user actions.	Combination of peer-to-peer communication, client-server communication, and multi-server architectures.

Table 2.3: Summary of computational bottlenecks for CVE technology.

There are two issues, which arise from the description of the computational bottlenecks that warrant further discussion: solutions for scalability problems and solutions for reciprocity problems. Solutions to scalability problems are generally sought by reducing the polygon count of the graphics, and by optimising the efficiency with which the data stream is sent across the network. The design choices that are made to arrive at these solutions have a large impact on the way a CVE

behaves and thus on the final usability of the CVE system. In the next section (section two) an analysis is made of CVE design practise and the consequences of these practises for usability measurements, and usability as such. Solutions for reciprocity problems push the design of CVEs towards a faithful reproduction of our real world, in order to allow us to use our everyday skills to conduct collaboration in a virtual environment. This push also has consequences for the current CVE design practise discussed in the next section (2.3). Additionally, reciprocity problems will be alleviated through the development of user experience. An experienced user has a better understanding of the limitations of a CVE interface, and will be able to understand the interface struggles of inexperienced users more quickly than vice versa.

2.3 CVE Design Practise in Context

This section places CVE design in context within the larger field of computer science design practise, and the general field of Human-Computer Interface design (HCI). It is by no means exhaustive in its description of all design activities, but rather considers in pertinent detail each of the issues that seem to shape CVE design practise. This approach has been adopted in order to clearly identify the assumptions that are made about how CVEs should be designed, at the time this thesis was produced. Furthermore, it describes and discusses the ramifications and consequences of these design choices for the directions of development for CVEs. The aim of this section is to identify all the design choices that shape the CVEs of today, describe the type of choices that have to be made, and discuss the effect of these choices on the usability of CVE technology.

The next part presents the direct manipulation interaction paradigm (2.3.1), followed by parts that describe how data simplification affects design (2.3.2), how the general trend to prototype CVEs affects design choice (2.3.3), and how increases in bandwidth increase expectations of the technology even though the design still takes place within performance constraints dictated by the low-end users (2.3.4). This is followed by a discussion of the impact of current design practise on CVE design (2.3.5).

2.3.1 Direct Manipulation

Interface design is, amongst other things, concerned with making computer applications easy to use for humans. As computer technology has developed, and the interface design paradigm evolved from command line interfaces to icon and menu-based windows systems. The notion of 'direct manipulation' became the central interface design principle (Norman, 1988, Shneiderman, 1992; Preece et. al., 1994; Dix et. al., 1998). Direct manipulation refers to the idea that by representing data as visible, recognisable objects, it will make it easier for the user to recognise, understand and interact with that data than by applying more or less abstract commands. Whether the objects created for direct manipulation are simple 2.5D icons or 3D animated agents, their effectiveness relies on their resemblance to real-world objects, actions and newly developed metaphors. Consequently, interaction design in the 1990s has taken place within a generally accepted interaction framework based on real-world metaphors. Typically, the interface is thought of as a world in itself, a 'world-in-miniature' where the user can act and which changes state in response to user actions, thus relying on what is known as the model-world metaphor (Dix et al. 1998). It comes as no surprise that most CVE designs are based on this model-world metaphor too. Where CVEs are to support real world interaction, CVEs are designed

to look and behave like the real world. Realistic representation is expected to allow the transfer of conceptual real-world knowledge to the interface. However obvious a solution this may seem, there are some important issues that are easily overlooked. One of the first problems is that it is not known whether realistic virtual environments allow effective transfer of real-world knowledge. The visual representation may be misleading; users may grasp the analogical representation rapidly, but draw incorrect conclusions about permissible actions. A second problem is that it is not known whether realistic representations provide the easiest way of interacting with the CVE. The interaction required to achieve the correct results may be more complicated to perform in a CVE than in the real world. A third problem is that it is possible to create objects and actions in a CVE that have no real-world counterpart. Choosing the right metaphors to represent such objects and actions is not an easy task (Benyon and Imaz, 1999).

2.3.1.1 Hidden Assumptions: Creating Realistic CVEs

Hidden assumptions are not uncommon in any design tradition (Hollnagel, 1993). Implicit in the documentation of VR development projects is the claim that the added value of CVEs is that they are more intuitive to use (c.f. Bricken, 1991). If these implicit assumptions are made explicit they can be formulated as the following deduction to reveal the prevailing design fallacy (COVEN, Del. 3.5, 1997):

a. CVEs are realistic by making use of interactive 3D representations updated in real time, and
b. Realistic interfaces are intuitive.

From a) and b) follows:

c. CVEs are intuitive.

CVEs will not automatically be intuitive, unless they have been specifically designed to be intuitive; a CVE is not in and of itself intuitive. The expectation that CVEs will be inherently intuitive, may have something to do with the slow advance of the development of guidelines for CVE evaluation and design.

The task of designing realistic CVEs is affected in a number of ways. A first problem is that as 3D representations become more realistic, they demand more bandwidth and processing power. However, current CVE technology is developed within limited bandwidth and processing specifications. A second problem is that realistic representations lead the user to expect to be able to use all permissible actions that are available in the real world. However, to design all permissible real-world actions in a CVE involves a lot of time and effort; often more than is available in the CVE development life-cycle or project goals. This often means that realistic interfaces become less than intuitive to use. A third problem is that because CVEs are assumed to be intrinsically intuitive to use, any user problems are assumed to be solved by user training, user experience and the development of new social protocols. However, this resulting process of deferring the task of finding solutions to user problems to the users does not seem to be commonly acknowledged so far. It may well be that in some cases a much better interface design could be found by rethinking the original assumption of realism leading to intuitiveness. Inventing new interaction paradigms surely will not benefit from development based on hidden or uncertain assumptions about what is good, and this is why the author has tried to exemplify how the current interaction paradigm a priori affects CVE development.

2.3.1.2 Disciplinary Matrix

For the scientific research community a paradigm is a collection of theories, descriptions of successful research, assumptions, values and norms, which govern the style and the type of research and development that is conducted. It is a source of examples for junior researchers, of successful solutions found by applying the paradigm to problems. It could be said to be the disciplinary matrix, which contains all the knowledge, the generalisations, specific explanations, theories and complementary concepts needed to address these issues (Groot, ibid). The disciplinary matrix, firstly consists of the terminology and basic concepts that are used to describe and reason about the topic of research. This terminology and basic concepts are then used to formulate hypotheses, which can be put to the test. Developing a new technology involves the invention of new words and concepts, new working models, prototypes and theories. In fact it was noted in an American research report into the scientific research challenges of virtual reality (Durlach and Mavor, ibid) that there were no precise and generally accepted definitions of the terms used in VR technology development yet, and this is still largely true to date. It may seem obvious that such a task involves the integration of different kinds of knowledge and expertise. Traditionally, large software applications are developed in design teams consisting minimally of an application programmer, a domain expert, a usability researcher, and increasingly more often some representative users. HCI expertise and ultimately interface design guidelines are based on an integration of knowledge from such different disciplines as computer science, artificial intelligence, linguistics, philosophy, sociology, anthropology, design, engineering, ergonomics and human factors, social and organisational psychology, and cognitive psychology. Similarly, CVE research and development ought to benefit from this interdisciplinary approach.

However, it has to be noted that there has been a lack in thorough usability testing of CVE design so far, and partly due to this, a lack in the availability of integrated systematic knowledge about CVE design (Wilson, Eastgate, and D.Cruz, 1998; Johnson, 1998; Hix, Swan, Gabbard, McGee, Durban and King, 1999).

Amongst the reasons why usability testing of CVE design has been less than desirable considering the present HCI traditions are that the CVE developers of today have no more knowledge of interface design than is available through their course curriculum which is based on traditional software engineering techniques (Newman and Lemming, 1995; Sutcliffe, 1995; Wickens, Gordan and Liu, 1998). However, a lack of usage of these techniques has been observed in general (Landauer, 1995) and amongst VR developers (Wilson et al, 1998). Interviews, carried out by the author and others (see Chapter 6), have revealed that VR designers do not employ the readily available HCI knowledge that does exist today (Kaur, 1998). Another reason why usability testing of CVE design has been less than desirable, considering the present HCI tradition, is that the development of CVE technology is still in its early stages. The development of prototypes has to be based on a certain type of testing, but the assumptions on which CVE research and development are based, seem to be derived from the push of technology and not so much, as yet, from the pull of the users. A third reason why usability testing of CVE design has been less than desirable considering the present HCI tradition is that the different disciplines involved in building CVE applications each have their own terminology and concepts. In order for multi disciplinary teams to work together, they have to bridge this 'multi disciplinary gap' (Grudin, J., 1993). This gap is partly caused by the different disciplinary matrices with their complementary terminologies, partly by the ignorance

about the limitations of each respective discipline by members of the other disciplines, and partly by uncontrollable factors such as the social impact of interdisciplinary work, the social impact of the new technology when introduced into society, and the social impact of the challenge this new technology places on current understanding of technology and design. Uncontrollable factors need to be observed, documented and analysed for us to have a better understanding of them. Sociologists have made this their field of expertise within CVE research and development. Ignorance about the limitations of each respective discipline by members of the other disciplines is solved by synergistic teamwork and the application of a systematic development methodology such as the traditional software engineering methods advocated by the HCI literature. Finally, acknowledging and analysing the multi disciplinary language gap could be used to stimulate instead of confuse the debate between disciplines, a process deliberately started at CVE'96, followed by CVE'98 and CVE2000.

2.3.2 Simplifying Data-Exchange

Any design process is governed by trade-off decision making that should satisfy certain requirements (Norman, 1988; Howard, 1977, Smets and Overbeeke, 1995). This is also true for CVE interface design. The author conducted interviews with five CVE designers about their design practice during the COVEN project (the interviews are presented in Chapter 6). A basic form of analytical induction was used to draw conclusions from these interviews. This led the author to formulate their design practise as a continuous trade-off between the following two constraints:

i) A human constraint: the CVE has to be effective and intuitive.

ii) A machine constraint: the CVE has to utilise minimum computational load and network traffic.

The solutions to satisfy these two constraints seem to be respectively:

i) Use of realistic representations to allow users to transfer their everyday knowledge to the CVE.

ii) Simplification of representations and functionality to stay within performance parameters.

These solutions are conflicting in that by making objects realistic this also introduces the suggestion of available actions on objects that are probably not supported in the CVE, because the objects have been simplified in either their representation or functionality or both. This conflict introduces a potential for usability-breakdown. However, to make things worse, synchronous distributed groupware has to satisfy the performance constraints of the technology it is based on so stringently, that these trade-off design choices are very often made first in favour of the machine constraint and only then on the human constraint. Especially because CVE development is still in its early stages, the basic technology has to be developed first in order to have a test application for CVE usability. It has to be noted that these early design choices shape and obscure our thinking and expectations of what a CVE interface can or should do. For every CVE development project those early design choices ought to be carefully documented in order to be able to rethink these choices after usability testing.

During the longitudinal usage evaluation of the COVEN platform design activities the author deduced four general decision trade-off areas. Table 2.4 structures this information and lists for each category a number of specific usability problems that were found during the usability evaluations.

Trade-off Decision Area	**Category of CVE Usability Problem**	**Usability Problems**
Prototype development vs. Demonstrable applications	Hardware/Network/Software Problems	Lack of functionality. Latency in performance. Poor display quality.
Run-time performance vs. User performance	System Problems	Usability solutions not automatically device independent. Users with slow connection quickly judged to be uncooperative, users with a fast connection quickly judged to be uncollaborative. 'High-end users' judged as higher in status, competence and trustworthiness than 'low-end users'.
Object representation vs. Affordance representation	Application Problems	Meaning of objects within the environment not clear. Apparent or unapparent availability of actions not obvious. Realism vs. simplification choices based on performance constraints only.
Presence & co-presence vs. Minimalist design	Interface Problems	Interaction struggles in 3D space, such as: navigating in 3D space; picking of 3D objects; positioning precisely.

Table 2.4: Decision trade-off dimensions for CVE design and associated usability problems.

To summarise, there are CVE specific problems caused by technological limitations, latency, 3D interaction, and realism that affect both machine constraints and human

constraints in all kinds of different ways. As a result, the real challenge for CVE researchers is to "prioritise specific user and application needs and then to find ways of supporting them within a limited computing resource" (Bowers, Pycock, and O'Brien, 1996).

2.3.3 Prototyping

Requirements for an interactive system cannot be completely specified from the beginning of the lifecycle (Dix et al.,1998). The only way to be sure about features of the potential design is to build them and test them out on real users in a realistic setting. This is not a problem unique to CVE design, but the issue is raised here specifically because CVE design suffers greatly from this problem due to its novel 3D interaction paradigm. Instead, it is a general software engineering problem, which is tackled by applying a systematic usability engineering technique called iterative design, and an accompanying software engineering technique called prototyping. Iterative design is a purposeful design process that tries to overcome the inherent problems of incomplete requirement specification by cycling through several designs, incrementally improving on the final product with each pass. Prototyping is an engineering tradition used to demonstrate the iterative design choices. CVE prototyping is characterised by its evolutionary style, by which is meant that it is often a compromise between the production of a demonstrable application and a throw-away design exercise. The final system evolves from a very limited initial version to its final release as a demonstrator. The initial versions are limited in that they tend to concentrate on some aspects of the interactive system and ignore others, thus only providing partial functionality. CVE systems today tend to be vertical prototypes, in that they contain all of the high level and low level functionality, but for a restricted

part of the system, in order to be a demonstrator of novel basic functionality. Generally, the interaction is so free and flexible that CVE systems tend to be intrinsically unsupportive. CVE prototypes are often created incrementally, allowing large systems to be installed in phases. Early on the design team agrees on the core features, and a skeleton system is developed as soon as possible, or an underlying partially developed CVE engine and programming environment are used as a starting point for the iterative design process.

So far, rapid prototyping of CVEs is a rather informal process, and testing takes place amongst colleagues, often of the same professional background, or by virtue of the designers trying things out themselves. This has a number of consequences. Firstly, the effectiveness of the iterative design process is influenced, because the design is being modelled within a closed professional culture, on other professionals of the same discipline. This makes it more difficult for outsiders to gain a detailed understanding, and to share the available usability knowledge, and thus might contribute to the discipline gap introduced in the paragraphs about the disciplinary matrix (2.8.2). Indeed, VR programmers have been found to make little use of usability manuals because they do not readily apply to their design problems (Kaur, 1998). Another consequence is that this type of rapid development and manipulation is mistaken for rushed evaluation, which might lead to erroneous results and invalidate the only advantage of using a prototype in the first place (Dix, ibid). A third consequence is that it is likely that wrong design decisions are made at the beginning, which remain part of the design because designers tend to dislike discarding work in which they have invested time (Dix et al., 1998). Finally, there are CVE specific problems for the usability testing process. A certain number and type of interaction

features have to be sacrificed during the rapid development of a prototype. There is often limited attention as to which features are left out why, and how this influences the overall usability of the system.

2.3.4 Increased Bandwidth over Time

As computers become faster, either model complexity (the number of polygons used) or the update rates, can increase, but rarely both. For instance, at the start of CVE technology development most applications would only run on Silicon Graphics machines, but as time has gone on, more and more CVE systems run on low-end PCs. It is imperative to develop CVE applications that will run on a wide range of operating systems if CVE technology is to be accepted by consumers and businesses alike. However, in order to create usable, scaleable CVEs one has to work from the premise that there will always be a limit to available computing communications resources. There is a great potential demand for CVE technology, judged by the popularity of multi-user games accessible via the Internet, and the emerging interactive TV CVEs developed for interactive multi user cyber dramas (Murray, 1997).

2.3.5 Discussion

Over the past years CVE interface design has become a separate academic subject within the field of HCI. It has been suggested that possibly new interaction paradigms are needed to design effective CVE interaction. However, it is by no means clear yet what these should be. New design technologies and philosophies introduce new problems and hypotheses. Solutions of these problems or the acquisition of new knowledge sometimes lead to adaptations of a current or standard disciplinary matrix,

other times it leads to the invention of a new disciplinary matrix and subsequently a shift in paradigm. Whether CVE technology demands a paradigm shift is open to debate, but since CVE technology allows it users to do new things, in new ways, researchers have to be aware that possibly not all CVE interaction design problems can be solved using the current interaction design paradigm.

2.4 CVE Usability in Context

This section places CVE usability in context within the general field of HCI design. It is by no means exhaustive in its description of all usability models, methods or techniques, but rather considers in detail each of the relevant issues that shape CVE usability activities. This approach has been adopted in order to clearly identify the current conceptual model that governs how CVEs should be tested for usability. Furthermore, it describes and discusses the application of usability engineering methods to CVEs. The aim of this section is to give an overview of the evaluation issues that exist for the CVEs of today, describe the type of design and evaluation answers that are needed, and discuss the open issues as regards usability for CVE technology.

The next section presents the current usability paradigm (section 2.4.1), followed by sections that describe which traditional usability engineering methods apply to CVE development and how these usability engineering methods structure the choice of validation criteria with which to measure usability for CVEs (section 2.4.2), how a structured scientific approach guides the formulation of validation criteria and conceptual models help to structure the usability evaluation in an empirical fashion

(2.4.3), and a conclusion describing what kind of validation criteria and conceptual model could apply to CVEs (section 2.4.4).

2.4.1 Technological Change: Changing Design Needs

The term Human-Computer Interaction was adopted in the mid 1980s as a means of describing a new field of study concerned with the design, evaluation and implementation of interactive computing systems for human use and with the investigation of major phenomena surrounding them (ACM SIGCHI, 1992). During the past 20 years, HCI knowledge and methods have been expanded, refined and adapted to keep up with the rapid changes in the underlying technology.

Computers are becoming smaller, faster, cheaper, more popular and more often interconnected. These new and improved hardware and software technologies open up new challenges for HCI. VR technology, and particularly that supporting CVEs, poses such a challenge (Stone, 1993). Never before were humans able to share the same virtual space, move about freely, interact with shared virtual objects, and 'hear' and 'see' each other in real-time. The current analogies employed to think about this new technology are paradigms from the realms of theatre, movie making, comic design, architecture, mathematics, interactive storytelling, role-playing, and audio-video conferencing. It is also based on the collective experiences of the early virtual communities based at certain fixed sites using bulletin board style communication, and multi user dungeons (Rheingold, 1994). Instead of addressing the interface between human and computer, it would seem that HCI now also needs to address, what the author would like to call the "inter-space", the space between users in the virtual environment, or the interaction that is supposed to take place inside that space

and how to represent those actions. Complex social interactions are represented through an impoverished medium, which may have serious consequences for the usability of the designs (Brown and Duguid, 1994). However, it has been found that CVE users perceived the CVE as a social space and interpret the occurrences in this virtual social space similar to those encountered in real spaces (Jeffrey and Mark, 1998). Although this thesis focuses on the social interactions in the virtual space, it does not exclude the objects in the virtual environment and the CVE itself from the analysis of those social interactions.

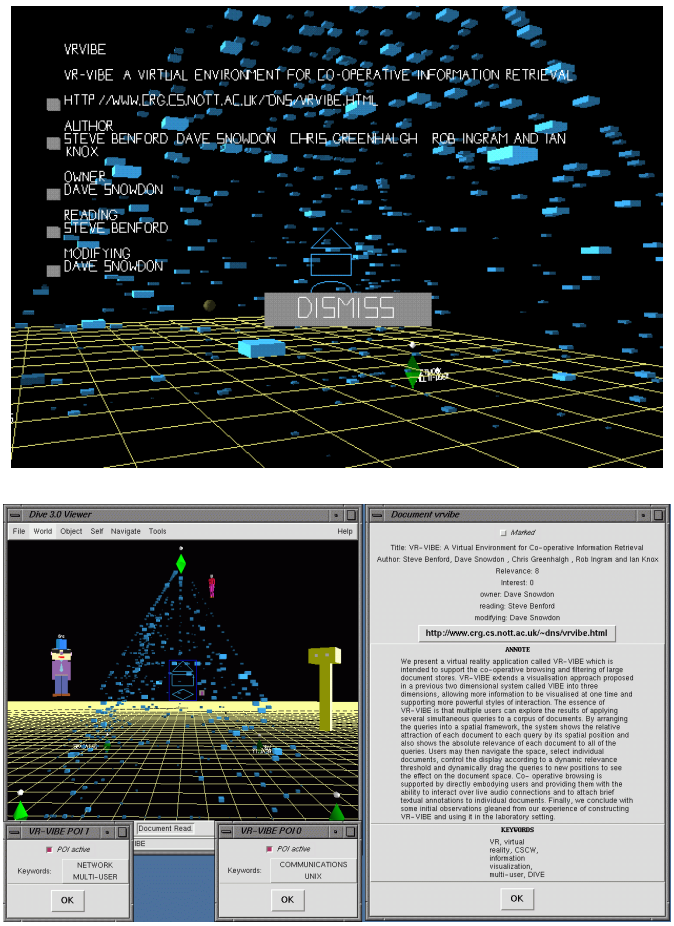


Figure 2.3: The fastness of CVE space (pictures from Snowdon, 1996).

Under focus is the idea of a social interspace, and the artefacts through which social interaction in a CVE is achieved. Illustrations such as figure 2.3, show the vast virtual

space, with a number of virtual embodiments in it representing other active users, and different types of data objects, representing something in a structured way to the users. The users can manipulate the objects and discuss them whilst observing the objects and each other. Thus, we are concerned here with the social space that is created inside the CVE.

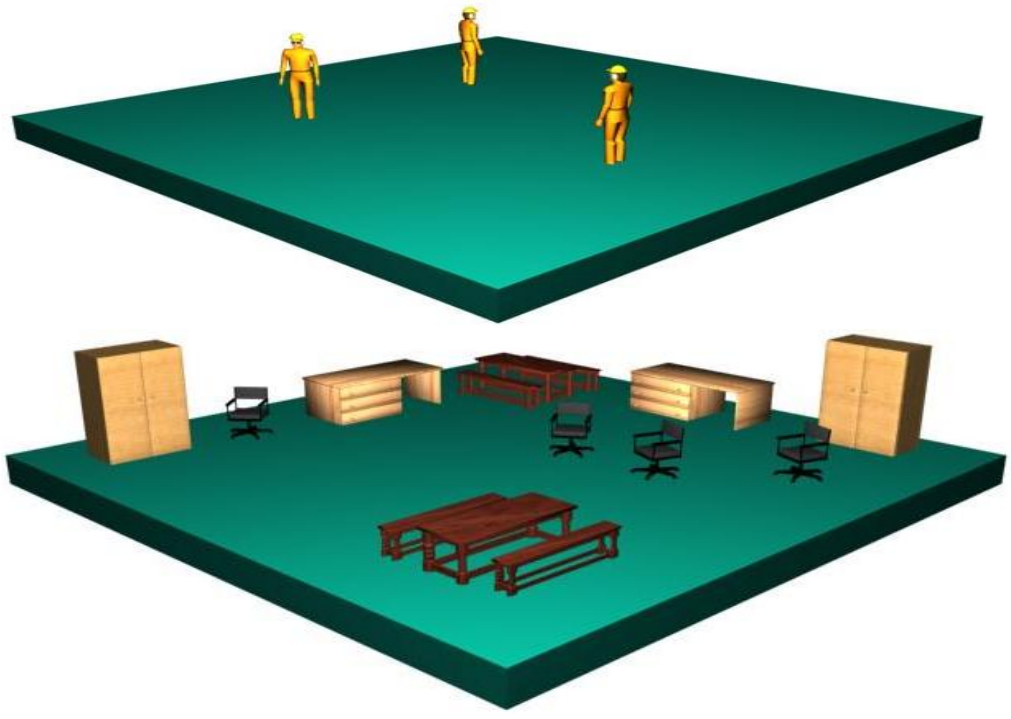
This means that this thesis is not directly concerned with the interface to the virtual environment, nor with the input/output technologies, nor with the presentation of new usability methods for CVE, although during the COVEN project the author of this thesis was involved in the development of guideline documents discussed in more detail in Chapter 9, Section 9.3.4. Rather this thesis concerns itself with the application of HCI principles and methods to the design and evaluation of social interaction and collaboration in CVEs.

All HCI takes place within a social and organizational context and this tradition is followed in the research reported in this thesis. Findings contributing to knowledge of the social and organisational impact of developing and using CVEs, in terms of practical CVE design activities are clearly noted in the text, as their collection and dissemination are one of the goals of this thesis.

Another goal of HCI is to carefully allocate tasks between humans and machines, making sure that those activities that are creative and non-routine are given to people and those that are repetitive and routine are allocated to machines (Preece et al., 1994). At the time of writing there is no particular consensus as to precisely how this allocution is accomplished for CVEs, or why, and many tasks that could be automated

are still left to the user. As CVE technology progresses more tasks will be automated, and one of the goals that this thesis aims to satisfy is to find such tasks suitable for automation, and make suggestions as to how this could be accomplished. This thesis aims to identify heuristics that could help CVE designers decide how to allocate CVE tasks between users and system in a systematic informed manner.

The author proposes a model of CVEs as consisting of several functional space-time layers of types of data (see figure 2.4). Firstly, there is an architectural layer, the actual lay-out of the space and adjoining or connected spaces. Secondly, there is a semantic layer; the actual meaning of the spaces, objects and actions in the VE, which the designers are trying to convey to the users. Thirdly, there is the social layer; this is the ability of the VE to connect one user with the other users occupying the same VE, by means of text, audio, visual, and other information cues. Finally, there is a temporal aspect to the whole CVE experience, which refers to the fact that the CVE is experienced over time.



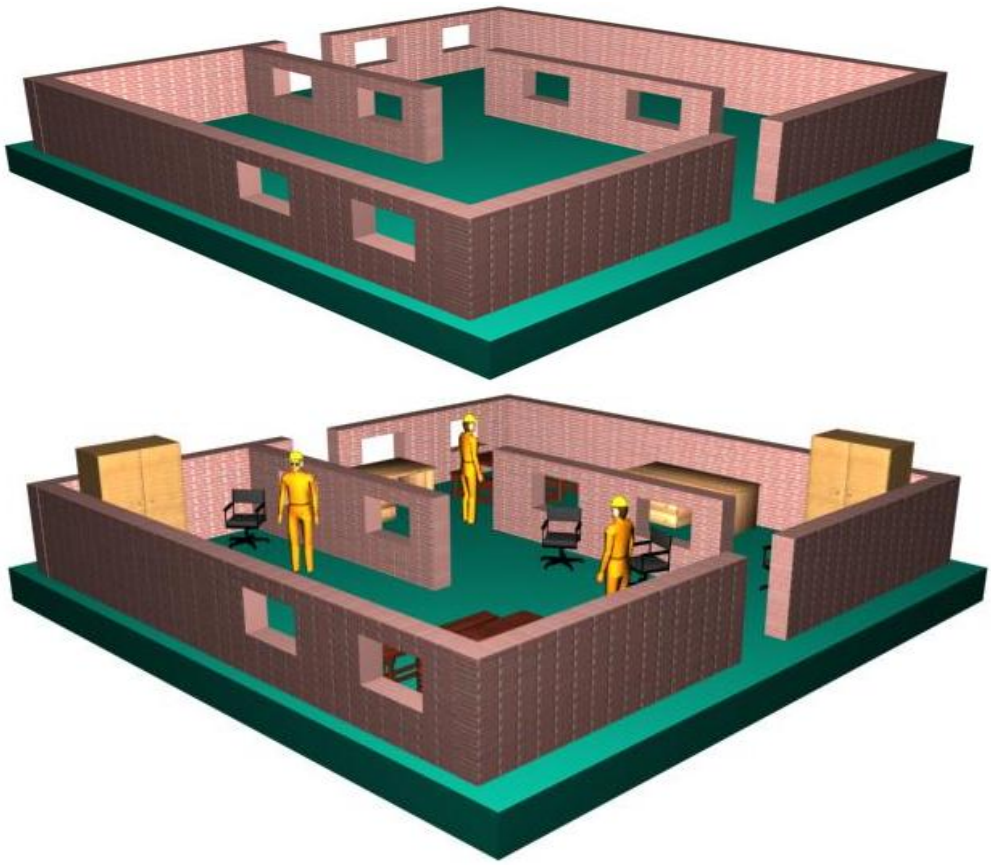


Figure 2.4: Functional layers of generic CVE[1].

In combination, these layers make up the CVE space as presented to the user. However, the total CVE interaction experience is also influenced by:

- The input/output media;
- The interface controls to the input/output media;
- The fidelity of the computational machinery and its communication network;
- The other users and the effectiveness of the interactions with them;
- The organisational settings or physically surrounding space and goings-on of each user;
- The social background and expectations of each user.

[1] Pictures draw by Damian Schofield.

This thesis is specifically focussed on the social layer, and the collaboration activities, which can be observed in this space.

2.4.2 Usability Engineering in General

Usability engineering takes place alongside systems engineering in a general process of software development called the software engineering life cycle. The traditional software engineering life cycle arose out of a need in the 1960s and 1970s to provide structure to the development of large software systems. The waterfall life cycle for development is depicted in figure 2.5.

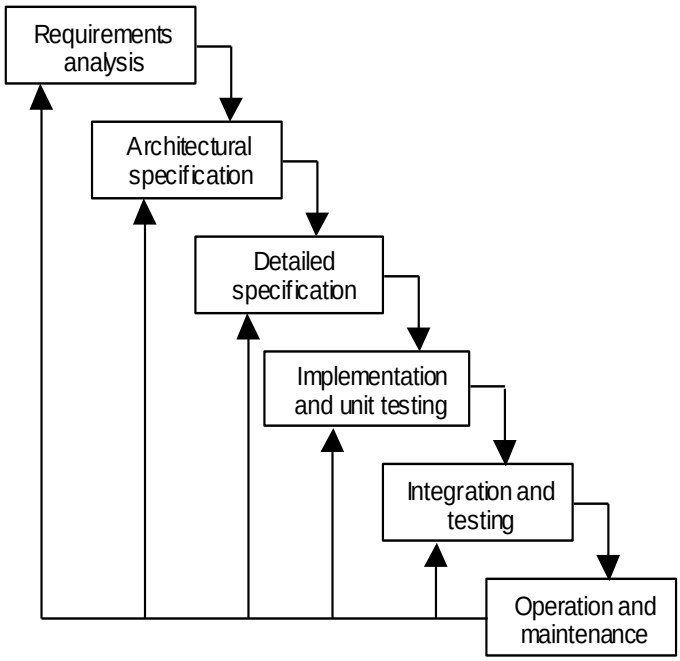


Figure 2.5: Activities in the software lifecycle (Dix, 1998).

The software life cycle identifies the activities that occur in software development. These activities are ordered in time and appropriate techniques are adopted to carry them through. A requirements analysis produces a requirements specification of what

the final system will be expected to provide. The architectural specification is a high-level decomposition of the system into components that are either developed from scratch or brought in from existing software. The decomposition allows for isolated development of separate components that are later integrated into the final system. The detailed specification is a refinement of the original description derived from the architectural specification. Usually, there will be more than one way to design these refinements within the behavioural constraints identified in the requirements analysis. Choosing the best refinement is a trade-off decision-making process based on an attempt to satisfy as many of the non-functional system requirements as possible. The detailed design is then implemented in an executable programming language. All the components are then tested to verify that they perform correctly and effectively. Additionally they are tested against HCI principles to validate that they satisfy the high-level requirements, which were in the requirements specification. This validation against HCI criteria is usually referred to as 'usability evaluation', or simply 'evaluation'. Once enough components have been implemented and individually tested, they are integrated as described in the architectural design. The integrated system is then subjected to verification and validation testing again. Once the system has been finalised, all work on it is considered under the category of maintenance, until a new version demands redesign (Newman and Lamming, 1995; Dix, 1998).

Usability evaluation is an engineering process (c.f. ISO DIS 9241-11). This process consists of stages that take place alongside the software development stages, and feeds back into the software design process as an iterative design-test-redesign activity until the goals of the system have been satisfied. Whatever the stage within

the software life cycle, the ideal usability evaluation consists of many team-based creative instances followed by

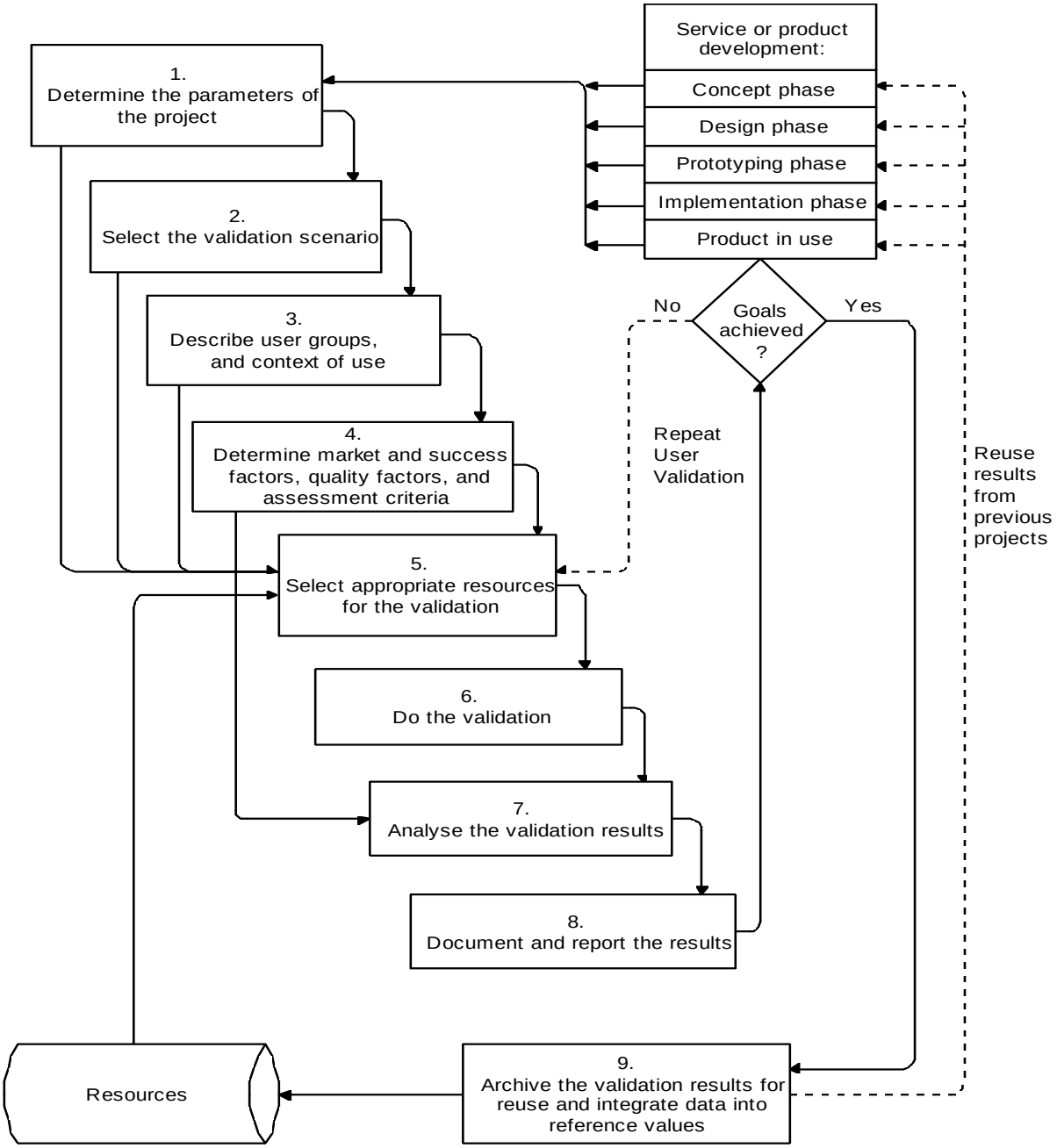


Figure 2.6: Structure of a Standard Usability Evaluation Process in: Melchior at al., 1995).

a progressive improvement and retesting of the design. This process needs to be structured and well documented, to ensure the quality, utility and pertinence of the

collected results. This documentation also allows for comparisons over several stages of evaluations and pooling of the results for future reference, and finally, it can be used to control the general cost-effectiveness of the usability activities themselves. Figure 2.6 illustrates the structure of an ideal usability evaluation process (Melchior et al., 1995).

Figure 2.6 shows only the most important connections between stages. The thick lines indicate the routine sequence between the nine stages of the user validation process. If, at the end of a validation process (stage 8), it is obvious that the results from user validation do not meet the validation criteria, then the system ought to be redesigned based on the reported validation results, and the improved system ought to be validated again (repeating stage 5 to 8) and the previously produced report extended with the new validation results. It has to be noted that many alternative paths through the process model can be assumed and are possible. The user validation process model does not describe a static process, but rather a flexible process. It is incorporated at any time in the development process, where information about user validation is needed. User validation is a demonstration of user acceptance of the application and a demonstration of a positive cost/benefit ratio of the application for the user compared to other solutions. Especially for the development of basic new technologies, such as CVEs, the demonstration of user acceptance and superiority of the pilot over alternative solutions is essential for decisions on further and future development, and correspondingly funding and investments. The objective is to demonstrate as convincingly as possible that the application does indeed do what it promised to do, and that it offers advantages as compared to relevant other, and

possibly competitive solutions. This requires a number of activities that can be summarised as follows:

- Identifying the quality factors to be measured (validation criteria).
- Defining the quality factors in terms of measurable quantities (acceptable limits of validation criteria).
- Selecting corresponding measurement methods (design of the experiment).
- Establishing realistic conditions for use (correct control of experimental conditions).
- Selecting subjects from the intended user group (random choice from the group of representative end-users, controlling for background and gender).
- Designing an appropriate test plan (complete strategy, documented).

The choice and use of validation criteria for interactive systems are much debated topics, because it is generally preferable to use methods that yield precise and clearly defined data, so that generalisable, replicable and comparable knowledge is gained, This knowledge is than summarised for greater understanding of the precise nature of the validation criteria, and as reference values for the evolution of the application and for further new development in the future. Thus, in order to be able to collect precise and clearly defined data the quality factors, or validation criteria must be carefully chosen. Especially for new technologies, such as CVEs, it will be difficult to predict which validation criteria will measure usability. In order to establish the validation criteria for a pioneering system, it is advisable to clarify:

- The goal of the usability evaluation (2.4.2.1);

- The place of usability within the overall criteria of acceptability of a system (2.4.2.2);
- How the validation criteria to test usability are to be decided on (2.4.2.3).

Next, the goal of the usability evaluation for CVEs is described (section 2.4.2.1), the place of usability in the overall acceptance of CVE systems is discussed (section 2.4.2.2), and finally is it critically examined how this information applies to the choice of validation criteria for CVEs (section 2.4.2.3). Creating new validation criteria is subject to systematic scientific methodology, which is further explained in section 2.15 (Systematic Scientific Approach to Usability Engineering for CVEs).

2.4.2.1 Goal of Usability Evaluation for CVEs

As a general rule, the role of usability evaluation is to test the system, to ensure that it actually behaves as expected, meets the requirements of the user, and usability breakdown problems are rectified. Traditionally, usability evaluation has a number of main goals (Preece, 1994; Newman and Lamming, 1995; Dix, 1998, Buie, 1999):

- Support the development process;
- Assess the extent of the system's functionality;
- Assess the effect the interface has on the user;
- Identify any specific problems with the system.

CVE usability evaluation would typically look to satisfy the same goals. The usability evaluation of a CVE ideally follows the standard usability engineering process, as evoked in section 2.1.4. The standard usability engineering process is designed to

support the development process, using the user requirements specification to assess the extent of the system's functionality, the effect the interface has on the user, and identifies any specific problems with the system. Although each development project is different, the approaches, methods, techniques and activities used during the standard usability engineering process to achieve usability, are clearly defined and well documented. It is important to note that usability evaluation of CVE technology challenges these standard practises, because of the changes of the interaction paradigm from 2D to 3D, as evoked in section 2.7.

2.4.2.2 Place of Usability in Overall Acceptability of CVEs

Traditionally, usability is seen as one element in the overall acceptability of the system (Nielsen, 1993). A systems' acceptability can be subdivided into social acceptability and practical acceptability (see figure 2.7).

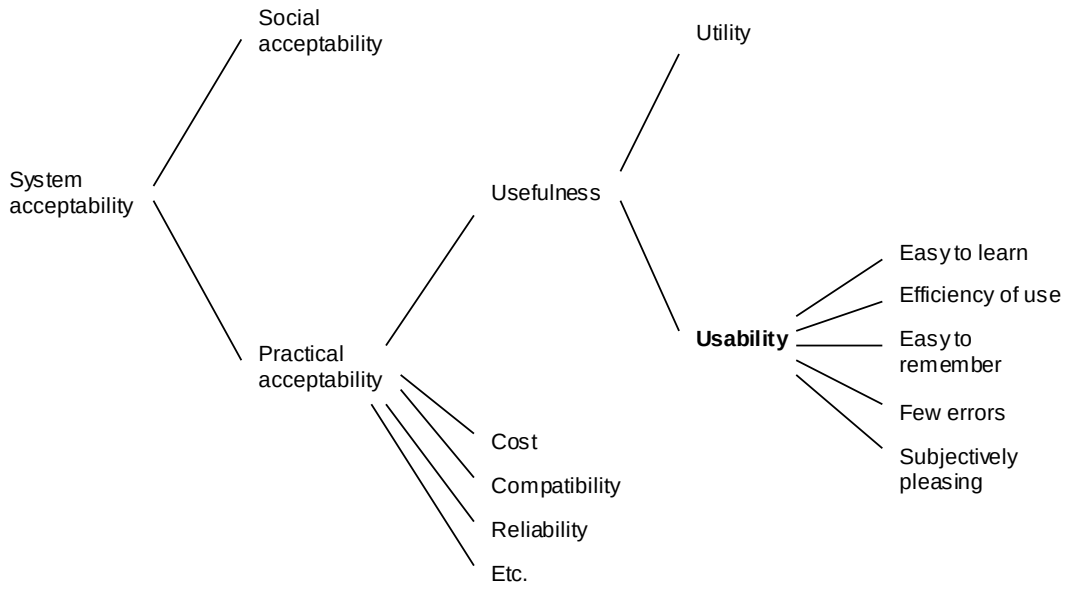


Figure 2.7: The place of usability in the overall acceptability of a system (Nielsen, 1993).

Social acceptability refers to how well a system is received by the users, including society, and if and how society as a whole can benefit from it. The latter being one of the direct concerns of the EC (Chester, 1998), as evoked in section 2.2. Looking at the diagram in figure 2.7, it may be obvious that social acceptability is not automatically a topic that will be evaluated during a standard "Usability Evaluation". For this reason the COVEN project (funded by the EC) included three social acceptability tests; one at the beginning of the project, to establish the needs and requirements for a travel information CVE from the Travel Agents' point of view; and the other two at the end of the project, from the consumer point of view (one in the laboratory, one in situ in a tourist agency).

Usability is generally considered to belong in the category of practical acceptability, which has to do with usefulness, cost considerations, compatibility, reliability, etc. Usefulness refers to whether the system can be used to achieve its desired goals. Usefulness can be divided into utility and usability. Utility refers to how well a system allows the user to fulfil their task(s), a concern traditionally addressed using a variety of methods, including ethno-methodological observations of work practise.

Usability refers to how well users can interact with the system to reach their goals. Thus, precisely defining and documenting how a user is to be supported, to reach what particular goal, is the first task in establishing validation criteria, before the usability of any particular system can be measured. A user requirements analysis is aimed at facilitating this stage, by guiding the researcher through a process of defining and documenting the user context, the user's goals with the application, the scenario

of use, and type of support needed for the user from the application in order to reach their goals satisfactorily.

2.4.2.3 Standard Usability Validation Criteria

A number of general factors have been consistently found to influence the usability of interactive systems (Nielsen, 1993). These can be divided into five categories:

- Easy to learn;
- Efficiency of use;
- Easy to remember;
- Subjectively pleasing;
- Low error rate.

These five categories typically form the basis for the validation criteria against which the usability of interactive systems is tested. For instance, it is calculated how many errors users make, or how much time it takes for them to complete their task. In such a case ‘error rate’ and ‘time taken’ are the validation criteria used to measure the usability of the system. Reduction of error rate and time taken to perform the task, is thus expected to show an improvement of the usability of a system.

It is to be expected that CVEs would also suffer from low usability if the above five factors are not taken into account in the design of the CVE. However, there may well be additional validation criteria for CVEs, or they might be slightly different from those found for previous interactive systems, since we are now dealing with the usability of three dimensional, multi-user collaboration systems. For instance, one of

the first new concepts to emerge from VR technology developments has been the issues surrounding cognitive immersion and the sense of presence in the virtual world. In order to be able to answer the question of how much the sense of presence is essential to the usability of a VE system, presence first needed to be defined, explored and expressed in terms of validation criteria to be able to measure the degree of presence. Only after being able to measure the degree of presence is it possible to systematically assess the effect of presence on usability. Validation criteria for open issues are traditionally created using a systematic scientific approach (Groot, 1969). This approach is discussed in the next section (2.4.3).

2.4.3 Systematic Scientific Approach to Usability Engineering for CVEs

There are two important reasons to adopt a systematic scientific approach to CVE usability:

- To support the usability engineering process in defining the precise measurement criteria for CVE usability.
- To better understand what constitutes CVE usability and how this related to standard usability knowledge.

How good any system is, depends partly on the quality of the system hardware, partly on the functionality of the software, partly on the user interface of the system, and partly on the user's experience, preferences, and specific needs. This creates a potentially confusing set of possible factors influencing usability, which can only be analysed effectively if a systematic, scientific approach is used. Additionally, if in the process of evaluation a potential usability problem is diagnosed, it is important to

understand the reason for the problem and not just detect the symptom (Dix, 1998). For this reason we need to empirically test hypotheses about the usability of CVE design features. By comparing differences in usability between different CVE design solutions, we can refine our understanding of what constitutes usability for CVEs. The empirical cycle of scientific inquiry consists of five phases (Groot, 1969):

Phase 1: 'Observation'. Collection and grouping of empirical materials.

Phase 2: 'Induction'. Formulation of (tentative) hypotheses.

Phase 3: 'Deduction'. Derivation of specific consequences from the hypotheses, in the form of testable predictions.

Phase 4: 'Testing'. Testing of the hypotheses against new empirical materials, by way of checking whether or not the predictions are fulfilled.

Phase 5: 'Evaluation'. Evaluation of the outcome of the testing procedure with respect to the hypotheses or theories stated, as well as with a view to subsequent, continued, or related investigations.

Our understanding of the usability validation criteria for CVEs is so limited that it could be said that CVE research and development is still in phase 1 of the empirical cycle. This means that we need to observe users in action, in order to achieve an understanding of their real task as it takes place. Based on this understanding we can than proceed to formulate tentative hypotheses about what usability for CVEs specifically means. These tentative hypotheses then, can be tested and refined until they allow the precise measurement of CVE usability and establish ideal usability solutions for each particular type of CVE. The observations should not leave anything out, because the aim is to create a true overview of "how the new technology influences what". A complete, or ecologically valid, picture of the whole technology

and not just the hardware/software, takes into account the designers, the design task, the evaluation task and the evaluators, as well as the end users using the technology as part of a set of tools to get their work done.

To define a validation criterion a precise specification is created of measurable behavioural objectives that must be met in order to pass judgment. What has to be specified is, how the software is to be judged as 'good' or 'bad'. This typically refers to the goal of the system: Can the user reach their goal using the system?; How do users reach their goal using the system?; How well can users reach their goal using the system?; How can the system be improved to help users reach their goal more satisfactorily?; etc. This type of information is typically expressed in the user requirements specification. The author presents the general goal of CVE applications below (see Table 2.5). Although the specific goal for particular VE applications may differ, there are a few general goals all (C)VEs have in common, because they all aim at creating a usable, credible virtual world for the user.

Goal of VE Application	**Description**
Presence	All VEs try to convey the illusion of a place where users feel present.
Navigation in 3D	All VEs must allow the user to navigate through the virtual spaces.
3D Object Interaction	All VEs must allow the users to interact with the objects in the virtual space.
Coordinating Multiple Tasks	All VEs must allow users to coordinate multiple tasks inside the VE and outside the VE.
Collaboration in CVEs	In the case of multi-user VEs or CVEs, users must be able to collaborate with each other.

Tab. 2.5: General goals for VE applications which can be used to define usability criteria.

Numerous researchers have defined validation criteria to measure the success of VE applications to satisfy one or more of the general goals mentioned in table 2.5, and several researchers have attempted to summarize these findings in order to enter stage 4 and 5 of the empirical cycle.

Thus, the most general goal for a user using a collaborative virtual system to receive support for usability, can be defined as follows:

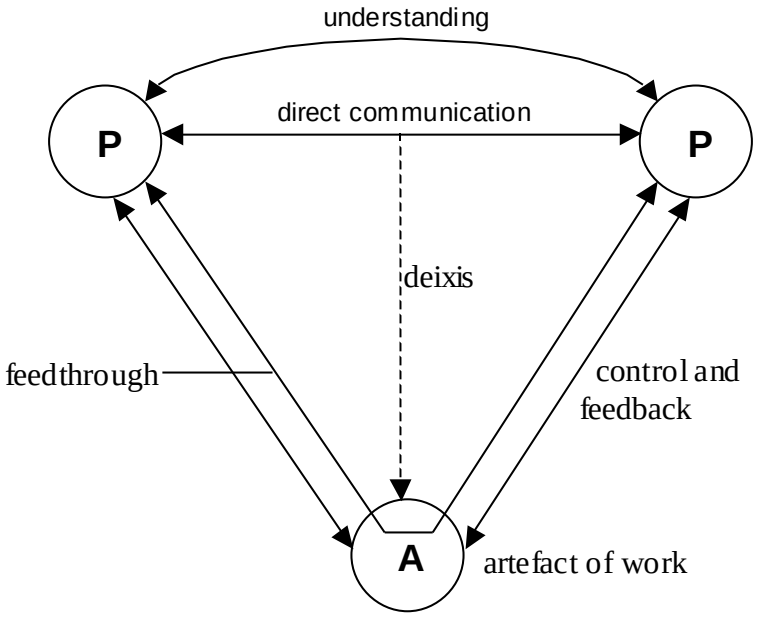
- To be provided with information about who does what, where and to whom, sufficient to understand and contribute to the goings on, compared to real life collaboration.

Chapter 3 critically examines human behaviour during collaboration, to assess the type of support users need from the CVE application more precisely. Additionally, there is some well-documented understanding on how users perceive their goals and the means to satisfy these goals, discussed in the next section (2.4.3.1).

2.4.3.1 Conceptual Models for CVEs

How well a user can reach their goal depends on how well they can perform their task, which in turn depends on how well they understand their task and on how well they can perform this task within the constraints from design of the system. Models help to formulate the constraints within which design takes place. As Donald Norman put it: “The problem is to design the system so that, first it follows a consistent, coherent conceptualisation – a design model – and, second, so that the user can develop a mental model of that system – a user model – consistent with the design

model" (Norman and Draper, 1986). Users have conceptual models of their task and of the application, and equally, designers use conceptual models to define the task that needs to be supported by the system, and models to conceptualise and view the system that they are creating. Figure 2.8 shows a general cooperative work model, which could also be said to depict CVE mediated user-user interaction.



P = participant

Figure 2.8: Model of cooperative work (from: Dix, 1998).

According to this model (Dix, 1998), the elements of cooperative work are defined as two or more participants and the things upon which they work. They are engaged in a common task, and interact with various objects, some of which are physically shared (being manipulated by two or more participants at the same time), and all objects are additionally seen as shared in the sense that they contribute to the cooperative purpose. The participants communicate with each other during the work, denoted by the arrow between them. Part of the purpose of the communication is to establish a

common understanding of the task and its progress between the participants. Participants might be using shared artefacts during their work; the two-way arrows between the participants and their artefacts of work denote this. The two-way flow represents the flow of control activities from the participants upon the artefacts, and the feedback, which should come from the artefacts to the participants. Additionally, the feedback of one participants' manipulation of shared objects as observed by the other participants is represented as feedthrough. Finally, a distinction is made between direct communication between the participants, and communication about the task, which refers to the artefacts used as part of that task; denoted by the dotted line representing the type of communication; speech elements called deictic or indexical (further discussed in Chapter3) common to cooperative work.

Based on this model (figure 2.8), the definition of the most general goal of a CVE to support usability can be stated as follows:

- The CVE should afford the perception of the functionality of the spaces, the objects, ones own virtual embodiment, and that of other participants, sufficiently for a typical user to achieve effective interaction and communication between themselves and the other CVE participants, and with the CVE.

This definition has helped to state the roles of different elements of the CVE experience, refining the first definition in the beginning of section 2.15. Analysing the cooperative work model and applying it to CVE has shown that he CVE participants should be enabled to conduct deictic communication about the shared objects between

participants, perceive feedback from object to interacting participant, and perceive feedthrough from manipulated object to the other participants. In short, CVE usability needs to take into account the user dealing with the immediate CVE interface, the user dealing with other users and objects inside the CVE, and the feedback of their own, other users actions and system actions on the objects and users in and with the CVE.

2.4.4 Discussion

Usability research for CVEs needs to take place on a very broad scale. It is not quite known what the usability factors are for CVEs specifically. In order to establish which factors are elements of usability for CVEs we need to make predictions about the outcome of design solutions for the usability of the system. These predictions are then rephrased as hypothesis so that they can be tested in an empirical manner. These theories allow us to predict will happen when people use a system and they allow us to guide the design; subsequently the design can be tested again, the theories refined, and the design updated.

2.5 CVE Applications in Context

This section consists of a review of three existing CVEs: MASSIVE, dVS, and DIVE, at the time this thesis was written. It has to be noted that there are other CVEs in existence as well as these, but these are the three major ones that the author is most familiar with. This review is by no means exhaustive in its description of all interactive features of each CVE, however it provides a short general description of each CVE, accompanied by images that provide impressions of the CVE space, CVE objects, and virtual user embodiments. Finally, a discussion is provided on the impact on CVE usability of the general limitations of embodiments, and the general

limitations of 2D mouse vs. 3D interaction that currently seem common to CVEs. The aim of this section is to give an impression of the type of CVEs designs that exist today, describe the type of embodiments and interaction solutions that are employed, and discuss the open issues as regards this state-of-the-art design for the usability evaluation of CVE technology.

The next section presents a framework developed during the COVEN project to compare CVE product features (2.5.1), followed by sections that describe which Massive (2.5.2), dVS (2.5.3), and DIVE (2.5.4), and a discussion in which it is described what the limitations are within which current CVE usability evaluation takes place (2.5.5).

2.5.1 Functional Comparison

During the COVEN project a rough comparison was made of the CVEs that were used during the project (DIVE and dVS) and other products (however, MASSIVE was not incorporated in this comparison). Nineteen CVEs are compared to each other by using a list of 12 features; each feature is given a score on a scale form 0 (absent or null) to 5 (very good). Although the numbers are not based on actual measurements, they do represent the collective, subjective opinion of the COVEN partners as expressed at the time. These results are shown in table 2.6. The COVEN partners scored each CVE on set of CVE functional features (see table 2.7). These features are: subjective views; high-level behaviour; rendering; rendering scalability; network scalability; web interface; audio communication; video communication; virtual humans; simulated crowds; spoken language; and usability guidelines (table 2.7 provides the definition of the terms).

Product	Origin	Platform			Features											
		Unix	PC	Mac	Subjective Views	High-Level Behavior	Rendering	Rendering Scalability	Network Scalability	Web Interface	Audio Comm	Video Comm	Virtual Humans	Simulated Crowds	Spoken Language	Usability Guidelines
MASSIVE	Uni of Nottingham	X	X		-	-	-	-	-	-	-	-	-	-	-	-
DIVE	SICS	X	X		5	5	4	5	5	5	5	5	5	5	4	5
dVS	Division	X	X		3	5	5	3	5	3	5	1	5	0	0	5
ActiveWorlds	ActiveWorlds		X	X	2	3	4	2	4	1	0	0	5	0	0	1
Blaxxun	Blaxxun Interactives		X		2	5	4	1	3	5	5	5	4	0	0	1
Community Place	Sony		X		0	4	4		4	5	0	0	3	0	0	
Interspace	NTT Software		X		2	5	3	1	2	5	5	5	2	0	0	1
TalkWorld	Etchinghill Studios		X		2	3	3	2	4	1	5	0	3	0	0	0
ParaWorld	ParaWorld		X		0	3	2	1	2	0	0	0	2	0	0	0
V-Chat	Microsoft		X		0	0	2	0	1	0	5	0	1	0	0	0
Worlds Chat	Worlds Inc.		X		4	5	4	1	4	3	0	0	5	0	0	0
2nd World	Canal Plus		X		2	5	4	1	3	5	5	5	4	0	0	1
Pueblo	Chaco		X		0	0	1	0	4	0	0	0	0	0	0	0
VNET	S. White				**Unable to load the software**											
CoSpace	AT&T				**Web site temporarily closed**											
TeCo3D	Mannheim U./Siemens T.C.	X	X	X	3	5	4	?	4	5	5	5	?	0	0	0
JavaMoo	Bang Space Inc.	X	X		**Unable to load the software**											
DeepMatrix	Geometrek		X		2	5	4	1	3	5	5	5	4	2	0	1
SCOL	Cryo OnLine		X	X	5	3	2	2	4	3	5	0	1	1	0	0

Table 2.6: Comparison of CVE platforms that were used during the project (DIVE and dVS) vs. other products.

Not every feature has a very obvious direct bearing on the usability of CVEs, however effectively each of the features plays a role in creating a credible, usable CVE.

Feature	Definition
Subjective Views	Ability to manage a separate view adapted to each participant.
High-level behaviour	Ability to produce complex interactions with the world components.
Rendering	Quality of rendering with reference to the hardware facilities available.
Rendering scalability	Ability to support environments that are geometrically both large in

	extent and deep in details.
Network scalability	Ability to support a large number of active processes in a highly interactive environment and over network with varying latency and bandwidth configuration.
Web interface	Level of integration with the web (URLs, documents, etc.).
Audio communication	Ability to provide an audio channel to participants and quality of the communication.
Video communication	Ability to provide an video channel to participants and quality of the communication.
Virtual humans	Precision and rendering of human avatars.
Simulated crowds	Precision and rendering of crowds.
Spoken language	Ability to provide a natural spoken language interface and width of covered speech.
Usability guidelines	Presence and quality of usability guidelines for CVEs.

Table 2.7: Definitions of the list of features that was used by COVEN partners to score the CVE platforms.

2.5.2 Massive

MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments) (Greenhalgh and Benford, 1995; Greenhalgh, 1999), is a CVE developed to run across the Internet. Its graphics have therefore been kept as simple as possible. It has a 3D graphical window, a 2D icon based interface control window, a plan view in which users can find distant participants and other objects, and users are able to communicate via an audio link. The primary goal of MASSIVE was to create and demonstrate scalability for CVEs, introducing the "Spatial Model of Interaction". The spatial model of interaction governs participants' awareness of the virtual world and its contents. It controls the degree of visible detail and audio volume

of distant sources. MASSIVE consists of multiple worlds, linked by portals that enable participants to move between worlds.

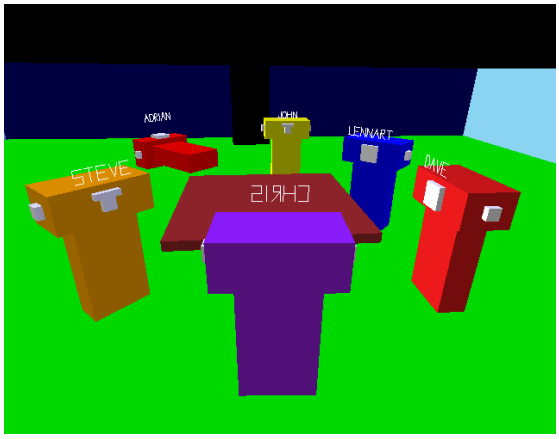


Figure 2.9: MASSIVE-1 screenshot showing "blockie" embodiments around a meeting table.

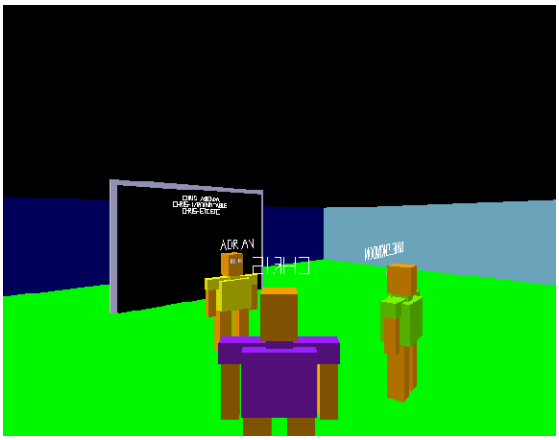


Figure 2.10: MASSIVE-2 screenshot showing more humanoid virtual embodiments in front of a "white-board".

Users have a virtual embodiment that allows for the expression of a few gestures by changing the position of the arms, and a few cartoon style expressions such as a question mark above the head. Users can put the embodiment into a sleep position to signify the fact that they are not controlling their embodiment because they are elsewhere engaged. Users can change their viewpoint from normal view, to various out of body views and birds eye views. Users are able to interact with objects within

the CVE, by clicking on them and moving them. See figure 2.9 and 2.10 for images of MASSIVE, and two different stages in the development of virtual embodiments.

2.5.3 DVS

DVS/dVISE Virtual reality System (Rygol, Ghee, Naughton-Green, and Harvey, 1996), is a CVE developed to run across high-performance networks, such as Ethernet and ISDN. Consequently the design of this CVE has not been particularly driven by concerns about limiting bandwidth usage. This stance may possibly have had support from the original design choice of the system, which is based on distributed architecture technology (multiple processors supporting a single system). The application consists of a graphical window (dVS), an interface control window to select different modes of navigation and interaction, and several toolboxes (dVISE) which allow users to add to and change the contents of the VE.

Figure 2.11: dVS.

The system does not include an audio or text communication medium and users typically used the telephone or separate, external text or audio-networked program to

communicate, although the system has audio built in to broadcast VE sounds. User representation ranges from a virtual hand representation to a full body representation in the virtual space and they can select and interact with objects. Objects that are interacted with typically 'highlight' their outlines in response to being selected by a user, but no other feedback on who is the user interacting on the object is available. The system provides three modes of interaction: Desktop non-immersive (using a 2D mouse), Semi-immersive (using 3D peripherals such as a Spaceball, or tracked 3D mouse and polarising spectacles to get a 3D desktop view), and Immersive mode (using a tracked Head Mounted Display (HMD) and 3D peripherals to interact with the VE).

2.5.4 DIVE

DIVE has been developed by the Swedish Institute of Computer Science (SICS) (Frecon and Stenius, 1999).

Figure 2.12: DIVE.

Users can individualise their embodiments, and they can fly, rotate, and wander around. They navigate with the mouse. Worlds are connected via portals. Users can select objects, and manipulate them, and collaboration tools are provided, such as whiteboards and web-browsers. DIVE has many 2D pull-down menu's full of commands that help the users in their tasks, such as "find other users" automatically, set point of origin to automatically get back to, etc. DIVE has gone through many iterations of development and has subsequently evolved over the years. The source code for DIVE is freely available, and DIVE users can in principle extend the world using a programming language called tcl/Tk.

2.5.5 Discussion

Typical virtual embodiments are limited in their interactive abilities. For instance apart from the hand and head no other body part can be moved independently. Typical objects in CVEs are limited in their functionality; some are there for decoration purposes only; some are there but do not possess all the functions their real world counterparts possess, or are used differently. Typical CVE spaces are unbounded, unless walls, a floor and ceiling have been specifically designed.

2.6 Conclusions

There are a number of general limitations to the current design of CVEs, which will most likely lower the usability scores on any evaluation immediately. The most obvious limitations are caused by the CVE spaces, objects, including the virtual embodiments, and the 3D interaction. The CVE spaces, objects, embodiments and 3D interaction are all represented in a simplified manner, which means that there are fewer cues for understanding the constraints and affordances for action. For instance,

way finding is notoriously difficult in a VE due to the limited field of view and other factors. Selection and manipulation are often difficult due to the fact that the input device does not afford the same manipulation as a real hand or tool would, and the output device does not provide the same feedback as its real world counterpart would do. It is by no means clear yet what type and how to provide feedback and feedthrough to all participants in a shared 3D environment about actions taken by the participants. The delay caused by the network on the data exchange between each geographically remote CVE participant is another aspect that will affect the design choices and the usability evaluation outcome. It is more than likely that usability problems of this nature will be uncovered by usability testing. This thesis attempts to document these types of problems alongside the other findings.