Developing Future Interactive Systems

Maria-Isabel Sanchez-Segura

Developing Future Interactive Systems

Maria-Isabel Sánchez-Segura
Carlos III Technical University of Madrid, Spain

IDEA GROUP PUBLISHING
Hershey • London • Melbourne • Singapore

Acquisitions Editor: Mehdi Khosrow-Pour
Senior Managing Editor: Jan Travers
Managing Editor: Amanda Appicello
Development Editor: Michele Rossi
Copy Editor: Maria Boyer
Typesetter: Jennifer Wetzel
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: http://www.idea-group.com

and in the United Kingdom by
Idea Group Publishing (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 3313
Web site: http://www.eurospan.co.uk

Library of Congress Cataloging-in-Publication Data

Developing future interactive systems / Maria Isabel Sanchez-Segura, editor.
p. cm.
Includes bibliographical references and index.
ISBN 1-59140-411-8 (h/c) -- ISBN 1-59140-412-6 (s/c) -- ISBN 1-59140-413-4 (eisbn)
1. Interactive computer systems. I. Sanchez-Segura, Maria Isabel, 1971-
QA76.9.I58D438 2004
006.7--dc22
2004016384

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Developing Future Interactive Systems

Table of Contents

SECTION I: WHOLE VIRTUAL ENVIRONMENTS DEVELOPMENT METHODS

SECTION II: DESIGNING VIRTUAL ENVIRONMENTS

SECTION III: COLLABORATIVE VIRTUAL ENVIRONMENTS AND MIXED REALITY

Preface

> *"Cyberspace...a global artificial reality that can be visited simultaneously by many people via networked computers."*
> (Gibson, 1984)

One of the mass media communications with the most rapid growth in recent years is the Internet (McKay, Matuskey, Testani, et al., 1998). This increased importance had a major impact on society, as many people spend a lot of time on the Internet because of work, entertainment, an so forth (Welch, 1996; Damer, 1996, 1997; Bruckman, 1997). At the beginning, the use of the Internet was limited to chat, e-mail, file transfer, and so forth, but with time the Internet started to be used as the way to link people who were geographically dispersed. This marked the beginning of the first kind of virtual environments called MUDs—Multi-User Dungeons.

This book is intended to help in the understanding and use of virtual environments (VEs), starting with its beginnings and tracing their evolution, as well as providing in-depth information to develop them formally in order to guarantee a high degree of quality.

Motivation

The origin of MUDs can be traced back to 1978, thanks to the efforts of Roy Trubshaw and Richard Bartle who developed the first MUD (Carton, 1995). These kinds of applications were just textual and focused on entertainment. When the goal of these MUDs took a different path towards a more social focus, the term social-MUD was born (Dourish, 1998), and with it the first social-MUD, called Tiny-Mud, was developed in 1989 by Jim Aspen (1989).

MUDs continued to evolve and in parallel, taking advantage of technological advances. Some branches in the development of MUDs endowed these systems with a graphic interface; this was the birth of VEs. The first VE, called Habitat, was developed in 1985 by Lucas Film. The Habitat interface was based on two-dimensional graphics, and it was the first time the graphical representation (called "Avatar") of the user was included in the VE.

From this moment, a lot of VEs—some of which included three-dimensional representation, sound, capability to create new objects during the execution of the system, virtual reality devices, and agents—appeared (Sloman, 1999).

The term VE does not have a single and accepted definition (Damer, 1997; Eastgate, D'Cruz, & Wilson, 1997; Brand, Fanzen, Klintskog, & Haridi, 1998; Landauer & Bellman, 1998; Saraswat, 1997; Maher & Skow, 1999; Kulwinder, 1999). In general, we can affirm that VEs are software applications that can be executed in the network and allow the collaboration, learning, training, and simulation in environments such as medicine, culture, teaching, and architecture, based on their development goal.

Taking into account the evolution of VEs and cataloging them in a general way as interactive systems, we think the term *future interactive systems* seems to be appropriate for this new age of multi-sensorial systems where perception and interaction with the system are being developed widely, and open a lot of new possibilities to "feel" the software.

Developing Virtual Environments

In the last few years, there have been a lot of VE developments, due to the attraction and novelty of VEs. As a result of the speed in the evolution of these systems and their strong relation with technological evolution, the development of VEs was characterized by an absolute absence of rigor. This is not something strange, taking into account that these were developed for easy solutions and not to reuse or analyze the system properly. It is impossible to develop VEs from an engineering perspective without formalizing.

Next appears the list of areas where VE development efforts have been dedicated in order to highlight the features they focus on:

- Some researchers have dedicated their efforts to improving "social interaction" in VEs (Mantovani, 1996; Cherny, 1995; Saraswat, 1997).
- Others (Fahlén, Grant-Brown, Stáhl, & Carlsson, 1993; Benford, Snowdon, & Greenhalgh, 1995) have focused on "mutual awareness" or perception of the VE elements.

- The representation of the avatar in the VE as the way to involve the user in the VE has been studied in HANIM (1998) and VRML (1997).
- Use of techniques and algorithms in the actual building in VE constructions (Ingram, Bowers, & Benford, 1996; Bridges & Charitos, 1997).
- Definition of the hardware architecture to be designed to support a VE (Brand et al., 1998; De Oliveira, Todesco, & Araujo, 1999; Maher & Skow, 1999; Gabbard, Hix, & Swan, 1999).
- Definition of recommendations or suggestions to be put into practice in the development of a VE (Boyd, 1996; Saraswat, 1997).
- The importance that VEs have and will have in the future (Brown, Encarnaçao, & Shniderman, 1999).
- Computer graphic techniques, visualization, communication protocols, and execution time (Donath, 1997; Kulwinder, 1998; Gabbard et al., 1999; De Oliveira et al., 1998).
- VEs' usability improvement, focusing on interaction mechanisms, presence, and perception (Donath, 1997; Eastgate et al., 1997; Conkar, Noyes, & Kimple, 1999; Kulwinder, 1999; Fencott, 1999).
- ü Development of software tools to support the implementation phase of VEs (GVU, 2000; Bowman, Koller, & Hodges, 1998).
- User-centered design techniques that have been defined in the area of Human Computer Interaction and should be useful in VEs (Conkar et al., 1999; Gabbard et al., 1999).
- Usability engineering is emerging as a new wave in the development of VEs (Gabbard et al., 1999).

As can be seen from the above, there are many areas in which VEs are used as a test bed or a powerful tool to achieve experiments, and simulations. But in spite of this interest, the way in which VEs are being developed is at a very immature level and there are no specific techniques to be applied during the development of these systems. So the quality of these systems cannot be ensured.

The problem with the development of VEs was so important that at the end of 1998 the National Science Foundation (NSF) and the European Union (EU), in a joint meeting, decided that it was necessary to improve the way VEs were being developed. They provided a set of recommendations on the points VEs research should focus on (Brown et al., 1999):

1. The process of gathering the needs and requirements of the VE users must be improved.
2. The parameters related to the design and evaluation of new technologies must be researched in depth.

3. The description of mechanisms and procedures to facilitate a multidisciplinary development are necessary.

The use of software engineering techniques in the VE development process should be very interesting to answer the first point proposed by the NSF and the EU. Software engineering discipline solved the software crisis in the ’70s. This problem was related to the fact that most of the software cost was related to the maintenance of the existing software instead of new software development. Maintenance was very expensive because software was being developed without any quality requirement.

The design of VEs is a complex process in which a lot of different variables are involved (Eastgate et al., 1997). Nowadays, there is little knowledge of VE design; neither are there guides on how to develop them (Kulwinder, 1999). Also the development of VEs is especially critical because a lot of models from different levels must be integrated (class models, 3D models, architecture models, behavior models, etc.) (Landauer & Bellman, 1998). In addition, a VE must be endowed with enough credibility, something not taken into account in traditional software. Table 1 summarizes the main differences between traditional software and VEs (Bricken, 1990).

Due to the difficulties in designing VEs and the potential improvements from the formalization of their development process, this book provides an engineering vision of future interactive systems, as opposed to other texts based on VE graphic design. In the chapters included in this book, some researchers and developers show VEs as software systems developed by applying repeatable techniques that allow the development of different features of the VEs, ensuring quality at the same time.

Table 1. Table showing the differences between traditional software and VEs

TRADITIONAL SOFTWARE	**VIRTUAL ENVIRONMENT**
The interface offers functionality.	The interface allows the user to be included/involved in the VEs.
People learn to use computers through the mechanisms of these.	VE technology adapts computers to the tasks humans have to carry out.
Users use the software developed.	Users are active agents within the application itself since VEs are designed to increase and change with users’ actions.
Usually, only visual.	VEs can be multi-functional, that is, have 3D sound and image, mechanisms to improve the sensation of immersion, and so forth.
Metaphors are used to give users a clear mental picture of what the application offers.	In VEs, participants interact directly with objects as if they were real. Therefore, no metaphor is necessary.

Book Structure and Use

This book is structured as follows. There are two initial chapters dedicated to the present and future of virtual environments in a general sense, Section I (Chapters III and IV), Section II (Chapters IV through VIII), and Section III (Chapters IX through XI).

- *Chapter I.* Based on the strengths and weaknesses of many current applications, this chapter discusses how to make virtual worlds (VWs) "real-world capable." With sufficiently realistic data and dynamic processing capabilities within VWs, we could work on analysis, engineering, invention, and design. This will require creating systems with sophisticated integration and analysis capabilities in order to suitably and continually scale up VWs with rich data sources, such as live data feeds and mobile sensors, and better computational and mechanical capabilities, such as multisensory interfaces and teleorobotics. Scaling VWs will require new strategies and capabilities for the numerousness and variety of resources.
- *Chapter II.* In this chapter, we look at some of the virtual reality technologies and their current effect on VEs. From here, we identify human technological drives and use this to highlight future technologies that will meld. Lastly, we look at how some of these changes will impact society and everyday life.

Section I focuses on the definition of two processes that improve the development of virtual environments, covering the whole software development lifecycle.

- *Chapter III.* This chapter undertakes a methodological study of virtual environments (VEs), a specific subset of such systems. It takes as a central theme the tension between the engineering and aesthetic notions of VE design. First of all, method is defined in terms of underlying model, language, process model, and heuristics. The underlying model is characterized as an integration of interaction machines and semiotics to make the design tension work to the designer's benefit rather than to eliminate it. The language is then developed as a juxtaposition of UML and the integration of a range of semiotics-based theories. This leads to a discussion of a process model and the activities that comprise it. The intention throughout is not to build a particular VE design method, but to investigate the methodological concerns and constraints such a method should address.
- *Chapter IV.* VEs can be seen as a special kind of information system, so they must be analyzed, designed, and implemented in this respect. This chapter presents a framework called SENDA, which defines a formal

process model to develop VEs from a software engineering point of view. As SENDA is a framework that covers the whole VE development lifecycle, this chapter defines processes and tasks for all the software phases. For each task, a set of techniques is mentioned and pointed out in the different chapters of the book where solutions for these techniques can be found, as well as external pointers on the books where specific techniques can be found.

Section II is dedicated to explaining in detail some design aspects of virtual environments development.

- *Chapter V.* Virtual worlds, construed in a broad enough sense to include text-based systems, as well as video games, new media, augmented reality, and user interfaces of all kinds, are increasingly important in scientific research, entertainment, communication, commerce, and art. However, we lack scientific theories that can adequately support the design of such virtual worlds, even in simple cases. Semiotics would seem a natural source for such theories, but this field lacks the precision needed for engineering applications, and also fails to addresses interaction and social issues, both of which are crucial for applications to communication and collaboration. This chapter suggests an approach called algebraic semiotics to help solve these and related problems, by providing precise application-oriented basic concepts such as sign, representation, and representation quality, and a calculus of representation that includes blending. This chapter also includes some theory for narrative and metaphor, and case studies on information visualization, proof presentation, humor, and user interaction.
- *Chapter VI.* Traditionally, the development of virtual environments has been tightly dependent on the programmer's skills to manage the available toolkits and authoring systems. In such a scenario, the discussion of different design alternatives, future changes and maintenance, interoperability, and software reuse are all costly and quite difficult. In order to overcome this unsystematic and technology-driven process, conceptual modeling has to be included just before the implementation phase to provide a shared representation language that facilitates the communication among the different team members, including stakeholders. Reuse and redesign for future requirements also have to be included since conceptual models hide implementation details and constraints, and are cheaper and easier to produce than prototypes. As a first attempt to attain these aims, this chapter presents the basis of a constructional approach for the VEs conceptual modeling through a set of complementary design views related to the VE components and functions. Moreover, we explore how these design issues might be addressed by hypermedia modeling techniques, given the similarities between these two kinds of interactive systems and the maturity reached in hypermedia development.

- *Chapter VII.* Virtual environments (VEs) have a set of characteristics that make them difficult to design and implement: distributed nature, high-level graphical design, technology novelty, and so forth. Besides, because of criticisms or the repetitiveness of some roles played in them, some of the characters of the VEs usually have to be automated. The risk is to pay too high a price, losing attractiveness, usability, or believability. The solution proposed in this chapter is to control the automated avatars associating them with software agents, becoming intelligent virtual agents (IVAs). With this aim, an architecture to manage the perception and cognition of the agent is described. On the one hand, the perceptual module of this architecture consists of a human-like model, based on one of the most successful awareness models in Computer Supported Cooperative Work (CSCW), called the Spatial Model of Interaction (SMI). On the other hand, the cognitive module proposes an easy-to-configure structure, providing it with the precise mechanisms to exhibit reactive, deliberative, or even more sophisticated social behaviors, incrementing the believability of the IVA in the VE.
- *Chapter VIII.* This chapter proposes an architecture for the development of Intelligent Virtual Environments for Training (IVETs), which is based on a collection of cooperative software agents. The first level of the architecture is defined as an extension of the classical Intelligent Tutoring System architecture that adds a new World Module. Several software agents are then identified within each module. They communicate among themselves directly via messages and indirectly via a common data structure, and are used for the collaborative development of plans. Some details are provided for the most remarkable interactions that will be established among agents during the system's execution. The proposed architecture, and its realization in a platform of generic and configurable agents, will facilitate the design and implementation of new IVETs, maximizing the reuse of existing components and the extensibility of the system to add new functionalities.

Section III is dedicated to specific developments of collaborative virtual environments and mixed reality applications.

- *Chapter IX.* This chapter gives an overview of some of the issues that face programmers and designers when building collaborative virtual environments (CVEs). This is done by highlighting three aspects of CVE system software: the environment model (data structures, behavior description) that the system provides, the data-sharing mechanism (how the model is shared), and the implementation framework (the structure of a typical client or platform in terms of the services it provides to the user). When a CVE system is designed, choices have to be made for each of these aspects, and this then constrains how the designers and programmers go about constructing the CVE worlds themselves. The main body of the overview presents examples that highlight many important differences between CVE systems. The authors also relate their discussion to the common topics of network topology and awareness management.

- *Chapter X.* This chapter addresses one of the challenges the collaborative virtual environments (CVEs) research community faces, which is the lack of a systematic approach to studying social interaction in CVEs, determining requirements for CVE systems design, and informing the CVE systems design. It does this by presenting a method for studying multi-user systems in the educational context. The method has been developed as part of the Senet project, which is investigating the use of virtual actors in CVEs for learning. Groupware prototypes are studied in order to identify requirements and design factors for CVEs. The method adopts a rigorous approach for organizing experimental settings, collecting and analyzing data, and informing CVE systems design. The analysis part of the method shares many of the interaction analysis foci and expands on it by providing a grid-based method of transforming rich qualitative data in a quantitative form. The outcome of this analysis is used for the derivation of design guidelines that can inform the construction of CVEs for learning. The method is described in the third phase of the Senet project.
- *Chapter XI.* This chapter introduces a component-oriented approach for developing “mixed reality” (MR) applications. After a short definition of mixed reality, the authors present two possible solutions for a component-oriented framework. Both solutions have been implemented in two different MR projects (SAVE and AMIRE). The first project, SAVE, is a safety training system for virtual environments, whereas the goal of the AMIRE project is to develop different authoring tools for mixed reality applications. A component-oriented solution allows developers to implement better-designed MR applications, and it fosters the reusability of existing MR software solutions (often called MR gems). Finally, it supports the implementation of adequate visual authoring tools that help end users with no programming skills to develop their own MR applications.

This book is intended to help readers with different interests in virtual environments. Readers should start with Chapters I and II for an introduction to these systems and their uses. Chapters III and IV will be useful for those interested in the development of these types of systems, which follow a definite and formal process to guarantee quality. Within the development of a VE, the design process involves the study of many details and elements to provide specific design tools applicable to different elements of the VE under development. Chapters V, VI, VII, and VIII provide this information. Chapters IX, X, and XI have been included as specific cases of VEs relating to collaborative virtual environments and mixed reality.

References

Aspen, J. (1989). TinyMUD. Retrieved from: *ftp.tcp.com/ftp/pub/mud/TunyMUD/tinymud-pc.1.0.tar.gz*

Benford, S., Snowdon D., & Greenhalgh, C. (1995). VR-VIBE: A virtual environment for cooperative information retrieval. *Computer Graphics Forum, 14*(3), 349-360.

Bowman, D., Koller, D., & Hodges, H. (1998). A methodology for the evaluation of travel techniques for inmersive virtual environments. GVU. Georgia Institute of Technology. Retrieved from: *www.gvu.gatech.edu/gvu/research*

Boyd, S. (1996). The design of virtual environments with particular reference to VRML. Center for Electronic Arts. Middlesex University. Retrieved from: *www.man.ac.uk/MVC/SIMA/vrml-design/design.html*

Brand, P., Fanzen, N., Klintskog, E., & Haridi, S. (1998). A platform for constructing virtual spaces. *Proceedings of the Virtual Worlds and Simulation Conference (VMSIM'98)* (pp. 97-106), Society for Computer Simulation International, San Diego, California, USA.

Bricken, M. (1990). *Virtual worlds: No interface to design.* Technical Report R-90-2, Human Interface Technology Center, University of Washington.

Bridges, A., & Charitos, D. (1997). On architectural design in virtual environments. *Design Studies, 18*(2), 143-154.

Brown, J., Encarnaçao, J., & Shniderman, B. (1999). Human-centered computing, online communities, and virtual environments. *IEEE Computer Graphics and Applications, 19*(6), 70-74.

Bruckman, A. (1997). *MOOSE crossing: Construction community and learning in a networked virtual world for kids.* PhD dissertation, The Media Laboratory, Massachusetts Institute of Technology.

Capin, T., & Esmerado, J. (1998). EPFL-LIG's MPEG-4 body animation page. Retrieved December 1998 from: *ligwww.epfl.ch/mpeg4/*

Carton, S. (1995). *Internet virtual worlds quick tour: MUDs, MOOs & MUSHes: Interactive games, conferences & forums.* Ventana Press.

Cherny, L. (1995). *The MUD register: Conversational modes of action in a text-based virtual reality.* PhD dissertation, Stanford University.

Conkar, T., Noyes, J.M., & Kimple, C. (1999). CLIMATE: A framework for developing holistic requirements analysis in virtual environments, *11,* 387-402.

Damer, B. (1996). Inhabited virtual worlds. *ACM Interactions,* (September-October), 27-34.

Damer, B. (1997). Interacting and designing in virtual worlds on the Internet. *CHI 97*. Electronic Publications: Tutorials. Retrieved from: *www.acm.org/sigchi/chi97/proceedings/tutorials/bfdt.htm*

De Oliveira, M., Todesco, G., & Araujo, R. (1999). The limitations of interactive multi-user 3D environments in the WWW. *Proceedings of the Tenth International Workshop on Database and Expert Systems Applications* (pp. 279-283), IEEE Computer Society, Los Alamitos, California, USA.

Donath, J.S. (1997). *Inhabiting the virtual city: The design of social environments for electronic communities.* Doctoral thesis, Massachusetts Institute of Technology.

Dourish, P. (1998) Introduction: The state of play. Computer Supported Cooperative Work. *The Journal of Collaborative Computing,* 7, 1-7.

Eastgate, R.M., D'Cruz, M.D., & Wilson, J.R. (1997). *A strategy for the development of virtual environments applications.* Santa Clara, CA: Virtual Reality Worldwide.

Fahlén, L., Grant-Brown, C., Stáhl, O., & Carlsson, C. (1993). A space-based model for interaction in shared synthetic environments. *INTERCHI'93* (pp. 43-48).

Fencott, C. (1999). Towards a design methodology for virtual environments. *Workshop on User Centered Design and Implementation of Virtual Environments,* University of York.

Gabbard, J. (1997). *A taxonomy of usability characteristics in virtual environments.* Master thesis, Virginia Polytechnic Institute an Stage University.

Gabbard, J., Hix, D., & Swan, J. (1999). User-centered design and evaluation of virtual environments. *IEEE Computer Graphics and Applications, 19*(6), 51-69.

Gibson, W. (1984). *Neuromancer.* Ace Books.

GVU: Graphics Visualization Center. (n.d.). Retrieved from: *www.gvu.gatech.edu/gvu/research*

HANIM: Specification for a Standard VRML Humanoid. (1998). Retrieved from: *ece.uwaterloo.ca/h~anim*

Ingram, R., Bowers, J., & Benford, S. (1996). Building virtual cities: Applying urban planning principles to the design of virtual environments. *Proceedings of Symposium of Virtual Reality Software and Technology* (VRST'96), Hong Kong.

Kulwinder, K. (1998). *Designing virtual environments for usability.* Doctoral thesis, City University, London.

Kulwinder, K. (1999). Interacting with virtual environments: An evaluation of a model of interaction. *Interacting with Computers,* 11, 403-426.

Landauer, C., & Bellman, K. (1998). What is cyberspace? *Proceedings of the Virtual Worlds and Simulation Conference* (VMSIM'98) (pp. 16-21), Society for Computer Simulation International, San Diego, California, USA.

Maher, M.L., & Skow, B. (1999). Designing the virtual campus. *Design Studies, 20*(4), 319-324.

Mantovani, G. (1996). Social context in HCI: A new framework for mental models, cooperation, and communication. *Cognitive Science, 20,* 237-269.

McKay, D.P., Matuskey, P., Testani, S., et al. (1998). An architecture for training virtual worlds environments. *Proceedings of the Virtual Worlds and Simulation Conference* (VSIM'98) (p. 9). Society for Computer Simulation, San Diego, California, USA.

Saraswat, V. (1997). Design requirements for network spaces. *Proceedings of the Virtual Worlds and Simulation Conference* (VMSIM'98) (pp. 91-96). Society for Computer Simulation International, San Diego, California, USA.

Sloman, A. (1999). Evolvable architectures for human-like minds. *Proceedings of the 13th Toyota Conference on Affective Minds,* Nagoya, Japan.

VRML97 Specification. (1997). *International Standard ISO/IEC 14772-1:1997.* Retrieved from: *www.vrml.org/Specifications/VRML97*

Welch, D.J. Jr. (1996). Software engineering of VE: Integration and interconnection. *Informe Técnico, CS-TR-3720*, University of Maryland.

Acknowledgments

The editor would like to acknowledge the help of all involved in the gathering and review process of the book, without whose effort the project could not have been completed. Special thanks to all the staff at Idea Group Inc. that were always there to help in the production process. Special thanks to Jan Travers, who continually prodded me via e-mail to keep the project on schedule, and to Mehdi Khosrow-Pour, who motivated me to initially accept his invitation to take on this project.

Most of the authors of chapters included in this book also served as referees for chapters written by others. Thanks to them for providing constructive and comprehensive reviews.

I wish to thank all of the authors for their insights and excellent contributions to this book. Special thanks to Rose Sukhraj for her help, advice, and ideas. Finally, I want to thank my husband for his love and support throughout this project; my daughter, Natalia, who kept my mind active and stimulated my imagination; and, of course, my parents for giving me their valuable time and my brothers for their understanding and love.

Maribel Sanchez-Segura, PhD

Chapter I

Real Living with Virtual Worlds: The Challenge of Creating Future Interactive Systems

Kirstie L. Bellman
Aerospace Integration Science Center,
The Aerospace Corporation, USA

Abstract

Based on the strengths and weaknesses of many current applications, this chapter discusses how to make virtual worlds (VWs) "real-world capable." With sufficiently realistic data and dynamic processing capabilities within VWs, we could do medical interventions, analysis, engineering, invention, and design. This will require creating systems with sophisticated integration and analysis capabilities in order to suitably and continually scale up VWs with rich data sources, such as live data feeds and mobile sensors, and better computational and mechanical capabilities, such as multi-sensory interfaces and telerobotics. Scaling VWs will require new strategies and capabilities for the numerousness and variety of resources.

Introduction

Virtual worlds technologies underlie more and more of our critical human processes: how we entertain ourselves and socialize ourselves; how we teach and train; how we conduct ourselves in business, how we design and build our systems, how we deliver health care, how we negotiate and mediate with each other, even how we vote and conduct governmental affairs. As Howard Rheingold, an early commentator on virtual communities, wrote:

> *"In cyberspace, we chat and argue, engage in intellectual intercourse, perform acts of commerce, exchange knowledge, share emotional support, make plans, brainstorm, gossip, feud, fall in love, find friends and lose them, play games and metagames, flirt, create a little high art and a lot of idle talk. We do everything people do when people get together, but we do it with words on computer screens, leaving our bodies behind. Millions of us have already built communities where our identities commingle and interact electronically, independent of local time or location. The way a few of us live now might be the way a larger population will live, decades hence."* (Rheingold, 1992)

All our critical social processes are being altered by technology. Given this growing reality, how do we help technology developers and technology users make wise decisions on how they choose to incorporate technology into their critical human processes? How do we design reasonable systems and reasonable policies for those systems interlaced with new technology? How do we study the impacts of technology on ourselves and on our culture? How do we make both wise cultural and organizational decisions as well as wise technology decisions? How do we build understandable systems that incorporate ourselves as part of the system? The purpose of this chapter is to discuss how virtual worlds (VWs) could serve as a critical strategy for addressing all of these questions above. However, as discussed in the following sections, VWs can only do so by rising above their current engaging, but limited, applications. Current VW applications circumvent some of the limitations of time and space in the physical world, and they provide a forum for real, if limited, social interactions. However, real-world problems require not only real-world data, they require real ways of impacting the world with actions. This means that our challenge, as a research and development community, is to somehow keep the advantages of "virtuality," while intermixing VWs with realistic data, interfaces, and outputs as necessary for different application areas. As we will discuss, much progress has been made, but we have many formidable challenges to making VWs “real-world capable.”

We start the chapter with a brief definition and history of virtual worlds, followed by a discussion on some of the lessons learned from one of the most common virtual worlds application areas, education and training applications. The lessons learned from education applications form the basis for the discussion on the necessary enhancements needed to create more capable VWs. Specifically, we will comment on the need for integration and scalability at many different levels of the VW. In order to have “real-world capable” applications, we need sufficient details—both in data and in processes that act dynamically within the virtual worlds. With sufficient richness of details, we can use VWs for medical interventions, to do analysis and engineering, to invent, to design, and many other opportunities. Richness involves the inputs, the models underlying the world, and the outputs. Rich heterogeneous environments also require sophisticated integration at not only the physical levels (both hardware and network management), but also at the semantic levels and the little understood emotional, psychological, and social levels. In order to continually and incrementally build and understand such rich systems, we need highly flexible, highly analyzable and traceable testbeds that will allow us to develop, experiment with, and analyze the impacts of a wide range of VW resources ranging from different language and data structures to diverse sensors and telerobotics devices.

Background to Virtual Worlds

Virtual worlds (Landauer & Bellman, 1998c; Bellman, 1999), although drawing strongly from virtual reality technologies, differ from virtual reality (VR) in three ways. First, unlike most VR environments, virtual worlds are not necessarily homogenous simulation environments; rather they often have a large diversity of heterogeneous resources available through the environment. In some examples, users can access all of their computing resources—models, editors, Web sites, and so forth—from within the virtual world. Second, many of the “utilities” or “services” in the environment are embodied as agents that move and interact within the environment, communicate with users inside and outside the environment, and even modify the environment itself. Hence, in some of the VWs developed under the Department of Defense CAETI program (described in the next section) and in other VW applications (Gordon & Hall, 1998), agents in the VW serve as tutors, librarians, gossips, spies, playmates, evaluation agents, personal assistants helping students to manage and schedule their learning activities, and several types of guides (some of which specialized the tours of the

VW to the negotiated needs and interests of a given user). These agents will often come in through the same client-server protocol as a human user, and easily pass the Turing test in everyday conversation and activities.

Human and artificial agents are often represented by an *avatar* (a character representation in text or graphics, depending upon the virtual world). Lastly, like VR, virtual worlds are organized using a spatial metaphor: each of the separate places is called a "room." However, unlike most VR environments, these rooms are not just a picture or a description, but instead have rulesets or dynamical models determining the properties of the room or "setting," and constraining the behaviors of entities (objects, agents, and users) in that room. Hence, even in rudimentary form, these settings begin to act like little "eco-systems," with their own local physics and niche dynamics. One of the most important qualities of this for our discussion here is that these rooms are explicit models of the context within which we want the environment and its contents to interact and produce results, and within which evaluative processes present in the VWs can interpret these behaviors and interactions.

Virtual worlds rose from three major lines of development and experience: (1) role-playing, multi-user Internet games originally called MUDs (for Multi-User Dungeons and Dragons games (Bartle, 1990)), and now more recently, "MUVEs" (Multi-User Virtual Environments (Landauer & Bellman, 1996)); (2) virtual reality environments and advanced distributed simulation, especially those used in military training exercises; and (3) distributed computing environments, including the World Wide Web and the Internet.

Virtual reality and distributed simulations gave us experience with distributed simulation environments, especially for the applications in military training (Macedonia, 2002). They also provided us with some good examples of multimedia and multi-sensory worlds, with example worlds as diverse as military operations training (see Zyda, 2002, and for reviews of their leading work for these types of applications, Zyda et al. 2003; Zyda, 2002a, b, c; Schilling & Zyda, 2002); medical training; and NASA, Air Force, and Army flight simulators and tank trainers. Researchers in VR also produced high-end graphical environments and Avatars (for example, "Jack" from the University of Pennsylvania, Lewis Johnson's pedagogical agents, or Perlin's "dancers" (see Landauer & Bellman, 1998c)). VR research also developed the idea that one could use a spatial metaphor for working even in abstract spaces such as those useful for data analysis. However, because the text-based MUVEs have some very important properties for our discussions in this chapter and often are the least familiar to most readers, we will take a moment to describe them in more depth.

The first multi-user servers were originally developed in 1979 to allow users to play together an analog of Dungeons and Dragons, a role-playing game, (Bartle, 1990). Bartle, one of the original developers of MUVE technology, has recently written a good book (Bartle, 2003) describing in depth the development of some of the central technologies underlying MUVEs, from the early text-based MUDs up through the current massive, multi-player role-playing game environments.

Imagine reading a story set in some time and place. If the story is well written, it can feel like one is actually experiencing that situation or even becoming that character, regardless of whether the story is fiction or nonfiction. Stories can present information about a situation that is usually only learned through experience; they are particularly good at descriptions of complex settings that are very hard to construct (Dautenhahn, 1998, 1999a, b; Dautenhahn & Nehaniv, 1999; Raybourn, Kings, & Davies, 2003). If a multi-user virtual environment (MUVE) is designed well, it is like a well-written story in its power to transport the user to a different situation, but it has three other important features. First, for already created stories (often called "quests"), it is interactive, which means that the reader can affect the behavior and outcome of the story, so, in particular, the reader can explore the story in many different ways. Based on the reader's experiences in that world so far, they will also be exposed to certain characters, actions, and parts of the story. Second, it is also multi-user, so that the reader can work with, play against, or interact with other readers. All users within the same room can see each other's actions, character descriptions, and conversation with others (except when others "whisper," which is a private point-to-point communication not broadcasted to the rest of the room). Third, and most important of all, there is plenty of "room" (and much encouragement within this subculture) for users to create their own "stories," characters, and places. The act of creating (authoring) new rooms, objects, characters, and quests is known in the MUVE community as "building." Building is a critical way in which many users can contribute to an ongoing storyline, or in more serious applications, contribute knowledge and capabilities to the environment. Both humans and computer programs enter these worlds and act within them as distinct characters or objects with names, descriptions, and behaviors.

These MUVE stories are stored as databases on servers (that may also provide other services for managing the multi-user world) and accessed by the users (both human and computer-based processes) with client programs. A simple command language, provided by all the server programs for MUVEs, allows users to move around, act, and interact within the virtual world. There is also a simple construction language in many MUVEs that makes it easy for a player to immediately become a builder (an author and programmer) in that world. A MUVE implements a notion of *places* that *we* create, in which *we* interact with each other, and where *we use* our computing tools, together, instead of having all tool use and collaborative interactions mediated through tools used individually. Like other virtual reality environments, multi-user virtual environments employ the underlying spatial metaphor to take advantage of our seemingly innate ability to use spatial maps for many things. We are able to organize an enormous amount of data and information if we can place it in a spatial context. MUVEs elicit a surprisingly powerful sense of space using only text. Characters may gather in the same location for conversations and other group activities, where their interactions are not restricted (or interpreted) by the servers, and because the servers do not get in the way, it is as if they have become almost transparent (Gordon & Hall, 1998).

The sense of "being there" can be quite strong, and in fact, the emotional "reality" of human users comes across surprisingly well, and this in turn greatly enhances the sense of being there, making MUVE experiences very compelling (Schwartz, 1994; Turkle, 1995; Landauer & Polichar, 1998a). Although MUVEs are VR programs, the human interactions are real; only the physical ones are not (Riner & Clodius, 1995; Clodius, 1995). One reason for being interested in MUVEs is because there may be hundreds of people in the MUVE at any given time, moving around separately and independently, creating objects in real-time, and interacting with each other. From just the standpoint of social science or cultural studies, MUVEs are clearly an important new phenomenon (Lawley, 1994; Reid, 1994; Rheingold, 1993). There are now thousands of internationally populated text-based MUVEs, some with as many as 10,000 active players; as we will discuss in a moment, some graphical role-playing VWs now have several million players. These players are not simply visitors as to a Web site, but rather users who spend often several hours a day within that world building up that world.

Because of the large number and diverse types of virtual communities, they have come under increasing study (Hummel & Lechner, 2002; Rheingold, 1993). Some of these virtual communities have now been in existence as long as 15 years. They have elected town officials in some places; the users walk around their towns, have their own places that they build, and describe themselves as "living" there. They have imbued these virtual places with meaning. They have roles and functions that they play within those communities. As scientists, we want to understand more about why some of these virtual communities flourish over years and why others vanish within a month. Certainly for anyone interested in collaborative technologies and the future of network and Web applications, we need to know what they are doing right that they are able to live, work, and build together in these virtual communities.

There has been a growing interest in trying to understand the phenomena that occur in virtual communities and how they differ or not from physically co-located communities.

Many researchers have started to characterize and study the impact of virtual places on social processes (Gallager, 1993); some of the issues raised are discussed further on in the context of the challenges to creating better and better virtual worlds. Ironically, one of the strengths of the early text-based MUDs was that despite their relatively primitive technology, they provided a surprisingly effective basis for realistic social interactions (Rheingold, 1993; Riner & Clodius, 1995; Hughes, Walters, & Kort, 1994). The ability of humans to take the limited text-based MUDs and the available graphical environments, with their increasingly engaging but limited avatars, and form meaningful human relationships is a credit to human social capabilities. It serves very well in terms of some types of meetings, some types of community building and friendship. However, they are still too limited for real-world applications such as a doctor assessing the qualities of a gait to diagnose a brain tumor, or a psychotherapist gauging the emotion on a patient's face, or even apparently, a potential business partner offering a firm handshake to close a multi-million dollar deal for e-commerce. Much more realistic detail is required in the VWs both for realistic objects and behavior, and for realistic social interactions. Hence, the appropriate detail in the appearance and behavior of avatar movements, objects, and the VW settings is critical to realistic and appropriate responses in such demanding applications as crisis management, health care, psychotherapy, and military operations. However, supporting the social interactions necessary to such applications is as highly demanding: The nuances of facial expression, body language, and tone are critical to communication and critical to the responses of the humans involved in the VW and their ability to carry out 'real work'. Also, we believe that getting the social reality correct is critical not only to being able to conduct more realistic social transactions as needed for real-world capable applications, but also for eventually creating new types of human and group, even inter-cultural collaboration beyond anything we have imagined so far.

One of the most important qualities of MUVEs is that text-based MUVEs allow people the freedom of and richness of word pictures, something that we cannot imitate yet with any graphical environments. Text-based MUVEs have a much richer and more dynamic visual imagery than, say, movies or games, because it is within and customized to each player's imagination. Even with the simplest of construction languages, people experience a deep sense of being present within these virtual environments, partly because they have built those environment descriptions from their own imagination. This sense of real presence and real interactions leads to real emotions and real social interactions, even though they are mediated through text displayed on a computer screen.

Sometimes researchers in VR have assumed naively that the graphical VWs somehow make less use of imagination than a text-based one; this is not supportable from the viewpoint of cognitive science and psychology. The role of interpretation, personal projection, and cognition matter as much in the impact of perceiving and making use of a graphic representation as a text one. Certainly there are differences, and we need deep cognitive studies to characterize the differences not only between text and graphical images, but the increasingly rich world of haptic, auditory, and other new types of interfaces. In all cases, we know very little still about how to somehow separate out the contribution of imagination and personal projection from perception of what has been presented within the VW…or how to make best use of the differences caused by different representations and interfaces on the different types of tasks one wants to accomplish within the virtual world. We will return to this issue in the next section.

Multi-user virtual environments are also great equalizers: All people—not just what we call “technocrats”—can become authors or builders in a short amount of time. It is not the computer technology that makes MUVEs work (although one of the things we seek in our research is to develop better technology for supporting them). It is the writers and artists who create the world and the people who live in it. The MUVEs with better poets seem to last longer than the ones with better computer scientists. In fact, we’ve seen examples of eight- and nine-year-old children, who were raised in inner cities and were nearly illiterate, become, within a short amount of time, able to build up worlds (Hughes, 1995; Landauer & Bellman, 1998c). Their teachers, in several different projects, have reported the children’s enormous motivation to be part of these environments which had a noticeable impact on their efforts to read and to write well. One little girl reportedly built a 30-room mansion with gardens and pools. Another, an equally shy little boy, showed the author his gardens, where, when you looked at the flowers, they blossomed. This easy entrée to MUVEs extends across not only age, as just discussed, but across disabilities and gender. One of the most articulate people on one of the author’s favorite MUVEs is profoundly deaf; he is much more comfortable speaking to people online (and vice versa, other people are more comfortable speaking to him online). Some MUVEs have a near gender balance; one of the largest MUVEs of all has an ongoing culture of role-playing, and actually has a slight majority of female players (Leong, Web site).

Aside from being able to become authors, not just players, of the worlds, another aspect of these environments is that they have very simple client-server architectures, which means that people in these environments who only have teletypes can still participate. Others have speech-generation boxes because they cannot see. Lipner (1999), who speaks as a computer scientist and a blind user, has pointed out that the text makes it easier for many people with disabilities to use such interaction systems, because much of their current software was developed to handle text only and cannot yet annotate correctly other multimedia. For such users, they are much more blind in a graphically rendered MUVE or a CAVE than in a text-based MUVE. These environments are worldwide. You do not need sophisticated equipment or programming experience to become a player or participant. This low cost of entry to MUVEs has made these environments popular with a wide range of non-technical people. Many other researchers in both VW technology and other types of collaborative environments have been working on how to maintain a low cost of entry for participants, mostly through the development of tools that can be widely and inexpensively used. See There (www.there.com) for an example in the area of massive games, Eikemeier and Lechner (2003) from Bremen, Germany, with their iK^{now} tool based on Peer-to-Peer software for an example geared towards commerce, and Das et al. (1997), based in Singapore, for an example geared towards both entertainment and education with HistoryCity based on NetEffect (1997).

Different kinds of MUVEs use different construction languages. Usually, the variant of the word MUVE reflects the choice of language available. These different languages allow different classes of behaviors to be specified for the objects created by the users. Some of these objects can be created and used in real-time. Pedagogically, this can be very powerful. At one meeting of mathematicians on a MUVE, some colleagues were joking with the author about an “infinitely parallel quantum computer.” While the others joked, the author quickly created an object labeled thus with a few simple attributes, and threw it across the room to one of the others. Although this was done as a joke, think about the ability to make—even at a primitive level—a new idea active and visible, something that others can pick up, modify, duplicate, and walk out of the room with. One of the colleagues present that day still has the “quantum machine” (in his virtual pocket naturally).

Another important quality about these environments is that every object is a state machine. Therefore, the objects that you are holding in your hand, the rooms you have walked through, the things you’ve accomplished in that environment, can all contribute to determining what you see, what objects do to you as you walk through this environment, and sometimes even where you go when you walk through a door or perform some action. These properties allow authors to set up “quests” or interactive stories that have game or logical features that must be accomplished to succeed. They are also easy ways of structuring learning material. One of our colleagues, a good amateur Egyptologist, set up a quest that requires one to learn some middle Egyptian, both vocabulary and grammar rules. If you do not tell the boatman to take you across the river in proper Egyptian, you cannot cross, nor can you talk to the idols that give you other clues for finding the treasure and solving the puzzles. This particular quest is implemented in a virtual world that uses one of the simplest of MUD servers of all, a TinyMUD. For our purposes here, these properties also allow one to set up rules that define an initial “ecology” or “physics” for each room. Although not as sophisticated as the modeling provided in other virtual worlds, even the simple TinyMUD can help one to start describing the contexts for and the constraints on the interactions between avatars, agents, and objects.

Finally, the simple client-server architecture means that computer programs called “robots” can also be users, coming into the environment with the same interaction mechanisms that a human uses. They use the same commands that a human uses to move around the environment or construct new objects (see Foner, 1997; Johnson, 1999). Foner (1997) was one of the first to discuss such robots—Julia, a TinyMUD robot of the Maas-Neotek family—and to emphasize the "sociology of such agents" and why it was important to consider sociology for agent-oriented programming. In our experience, we could not tell one player was a robot until it was in a group situation, where its responses became less coherent, because its underlying pattern matcher could not keep track of multiple parallel threads of conversation. These robots give us many interesting ideas about the kinds of intelligent support that agents could do for people within virtual environments. At present, there are prototype robots that take notes for people; tell them stories about the area, room, objects, and people in the MUVE; and play games with them. They can follow people around, help them find things, and do errands for them. They can tutor them, help them find digital material, and give them tailored presentations on the computer programs or other objects available in the virtual world. Lastly, some robots helped us to monitor and evaluate the behavior of others in the MUVE (Bellman, 1997). In the Computer-Aided Education and Training Initiative (CAETI) project, a large educational technology research program sponsored by the author at the U.S.'s Defense Advanced Research Projects Agency (DARPA) from 1994 through 1998, these robots were also used for computer-based tutors and other evaluation agents (see Johnson, 1999, for an example). In the DARPA CAETI program, we added to the basic MUVE capabilities in several ways: we developed more advanced MUVE architectures, especially ones with the ability to keep a text, 2D, and 3D version of the world in sync. This was important especially because it always allowed users several ways of sharing in the world, even if at times in a limited way. The advanced architectures also allowed us to distribute the functionality underlying these worlds in more powerful ways. A good example of this was the better-distributed database management. We also made it possible to have many more types of heterogeneous tools available from within the environment. Some of these tools were new types of embodied intelligent utilities and agents that helped individual users (librarians, guides, and tutors) or conducted support activities across the world (evaluation agents). Some of these new tools also helped tailor resources to an individual user (Bellman, 2001). The key integration issues addressed in CAETI were the integration of heterogeneous resources (Bellman, 1994), interoperability of virtual objects among different worlds, and distribution of VW servers and databases (Rowley, 1997).

Since CAETI, there has been useful exploration and development in scaling VWs, experimenting with the underlying software and hardware architectures and networking strategies for distributing VWs and agent capabilities (Mamei, Zambonelli, & Leonardi, 2003; Cabri, Leonardi, & Zambonelli, 2000; Welch & Purtillo, 1997), and exploring interoperability strategies (Soto & Allongue, 1997, Greenhalgh & Benford, 1997). Some of the integration issues are explored in depth in this volume and are crucial to the development of future robust VWs. However in other sections of this paper, we will be describing some needs and strategies for intersecting the necessary integration and scaling advancements in the underlying software, hardware, and network architectures with the integration and scaling that must be accomplished at what we are calling here the necessary 'psychological scalability'. VWs have many levels that must be integrated and scaled up: the means by which information and diverse processes are brought into the VW, the means by which information and action is understood inside the VW (e.g., the needed traceability, analysis, and evaluation capabilities), and then again the integration that allows the real result to be effected from inside the VW. This has always been the challenge for virtual worlds. They depend not only on computer science and engineering, but increasingly on sensors and physiology for new interfaces, on cognitive science and social science, on mathematics and modeling essential for reasoning about the systems and evaluating the actions within the VW. In other words, to scale to real-world-capable VWs, we must open up and integrate with the real world. We must open up VWs and break with closed-system programming paradigm so common in computer science and move to open systems that bustle with diverse people, processes, sensors, and effectors.

One way to distinguish the different types of scaling issues is to compare the advances in the online multi-player games to the earlier text-based MUDs. Clearly, the graphical environments associated with online games have surpassed the number of people in text-based VWs, and with that triumph has come many new developments in building an infrastructure that can support the sheer numbers of players. Herz in her seminal work (1997) cites games with millions of active players (although subdivided in duplicate servers). For example, she reported that the *Lineage* online game has three million players in Korea (out of a population of roughly 48 million), with even college scholarships offered for outstanding players (Herz, 1997). *Everquest* now supports a real-world economy on eBay selling objects acquired from playing for a worldwide group of players numbering in the millions. Guilds within *Everquest* often have 5,000 participants, with leaders, governing boards, and so forth.

Furthermore, little by little the graphical environments are catching up with the text-based MUVEs in their social richness, their sense of commitment and social communities, even the ability of a user to author or build meaningfully within the worlds (see the Second Life Web site at *secondlife.com* and There at *www.there.com*). We need research on how these new games compare to different kinds of MUVEs—not only because of the differences in size, but because of the different goals for participating in that community and the structure of participation. We have much to learn from all the forms of virtual worlds: from text, graphical VWs and caves; from forms where the behavior of participants is highly scripted and limited (many games, many training uses of

VWs), to the open and creative MUVEs composed of thousands of users. But we will learn different things. In highly scripted games we can experiment on the choices, reactions, and learning of populations within pre-determined scripts or scenarios; in the non-games open VWs, we can learn more about the perceptions, potential behaviors, attitudes of populations in general (and in such a way as to inform our future scenarios). Part of this difference is very much the difference between psychological approaches to experiments, with its emphasis on carefully controlled settings and alternatives in order to narrow down contributing factors, and ethological approaches to experiments, with its emphasis on observation of minimally constrained individuals within their natural habitat. Both are important and potentially complementary approaches. (For a stimulating review, see the special issue of the *Scandanavian Journal of Information Sciences* on the intersection of Ethnography and Intervention, 2002.)

It is apparent that we still need much more development in traditional computer science, software engineering, and networking to meet the challenges of scaling up to the sheer numbers: the millions of users on diverse platforms distributed over thousands of servers with eventually billions of objects. Especially, we will need much more work on integrating the new types of data and interfaces necessary to the mobile, sensor-rich, real-world applications. Nonetheless, we believe that eventually these environments will not be limited by the availability of the environments or on the growing databases, but rather on the ability of the future online analytic capabilities. That is, what is not currently available is the "psychological scalability"—the analytic capabilities and other supporting capabilities—that would make sense of and ensure the use of the rapidly growing populations of users, databases, and tools based on massive VW games for real-world functions.

Moving Out from Games to Real-World Applications

VWs have moved out of being just game environments into many other application areas. Just in the last few decades, they have moved out of the "game" arena into educational and corporate environments for distance learning, collaborative learning, literacy support (at all grade levels, including adult), corporate meeting support, professional organizations, and even technical conferences (Bellman, 1994, 1997; Landauer & Bellman, 1996, 1998a, b; Polichar, 1996, 1997). For example, one of the early uses of MUVEs were as meeting places for small scientific groups. Pavel Curtis helped create two of the first projects (at Xerox PARC) using MUVEs for scientific computing (Curtis, 1992; Curtis & Nichols 1994): Astro-VR was geared towards the professional astronomy community, and Jupiter was targeted for use by researchers within Xerox. In these projects, Curtis and others attempted to keep the powerful world metaphor while adding audio, video, and interactive windows.

Some of the work that is relevant to eventually using virtual worlds for e-commerce relies heavily on current work on agents, because whether the user is inside a virtual world or allowing an agent to buy for them, there are similar issues of "providing support, trust, and legitimacy" in both cases. A good example of recent work occurred at the University of Linz, where Gabriele Kotsis and her group examined some of the issues in electronic commerce. They describe two approaches for designing and modeling multi-agent systems as they act on behalf of human organizations.

Medical applications are one of the most active domains for VW and agent research and development. A recent paper by Swiss researchers Nadia and Daniel Thalmann (TECFA Web site) gives an excellent overview of the different types of medical applications potentially available with the use of VWs and virtual humans, citing potential uses in a number of medical areas:

> *"For example, it is possible to simulate the effect of deficiencies on tasks like walking and grasping. For plastic surgery and facial deformations, we may simulate the effects on facial motion including speech. In surgery, use of a graphics database of organs and the impact of Virtual Reality may lead to surgical interventions in a virtual world. Psychiatry research may also find new important tools in the research in behavioral animation and knowledge-based animation."* (TECFA Web site)

Although computers have played the roles of patients for psychiatric training before, virtual worlds provide the opportunity to place the patient in a vivid context. Hence Thalmann cites some earlier work in the use of virtual reality and virtual humans in psychotherapies, "Using this new technique, it will be possible to recreate situations in a virtual world, immersing the real patient into virtual scenes. For example, it will be possible to re-unite the patient with a deceased parent, or to simulate the patient as a child allowing him or her to re-live situations with familiar surroundings and people" (p. 5). Thalmann and Thalmann emphasize that the applications here are not just for the training and education of both medical practitioners and patients, but rather provide opportunities for continual active analysis on the part of the medical practitioner of the current patient, the impact of interventions, and lead to new understanding on the part of the medical personnel. Also, it has been emphasized by many that such devices as optical see-through displays (Feiner, 2002; Milgram et al., 1995; Drascic & Milgram, 1996; Azuma, 1997) and other devices will allow surgeons and others to perform procedures on a real human while being visually supported by virtual displays overlaid on the real scenes. For example, Henry Fuchs heads a leading group at the University of North Carolina which seeks to:

> *"…develop and operate a system that allows a physician to see directly inside a patient, using augmented reality (AR). AR combines computer graphics with images of the real world. This project uses ultrasound echography imaging, laparoscopic range imaging, a video see-through head-mounted display (HMD), and a high-performance graphics computer to create live images that combine computer-generated imagery with the live video image of a patient. An AR system displaying live ultrasound data or laparoscopic range data in real time and properly registered to the part of the patient that is being scanned could be a powerful and intuitive tool that could be used to assist and to guide the physician during various types of ultrasound-guided and laparoscopic procedures."* (Bajura, Henry, & Ryutarou, 1992; UNC Web site)

There are now numerous conferences and Web sites describing the explosion of applications being developed. One Web site runs a monthly report for an organization called Virtual Medical Worlds to keep the virtual medical community informed. Part of the goal of Veersweyveld (1997) and many U.S. and European researchers is to improve the overall level of medical practice, for example, "there is the newly emerging structure of the medical world consisting of specialized clinics, general hospitals, and local doctors which can collaborate and facilitate a uniform level of medical practice." This dream is that all doctors, no matter how familiar with a particular procedure or with a particular condition, will be able to administer the best possible health care because of the availability of databases, VR support (showing the inside of the human, allowing one to rehearse a procedure on the real patient being able to follow the best path computer for one, etc.).

But it is not just to raise the standard of practice for doctors and medical staff. It is also to allow a new level of medical knowledge available to patients and to everyday people who could better help in an emergency. Hence, Silverman et al. (2002) describe a computer-based training game called "Heart-sense" to help individuals improve their recognition of heart attack symptoms and therefore hopefully seek help earlier and thereby reduce myocardial infarction mortality. They have created a virtual village in which:

> *"…users encounter and help convince synthetic personas to deal appropriately with a variety of heart attack scenarios and delay issues. Innovations made here are: (1) a design for a generic simulator package for promoting health behavior shifts, and (2) algorithms for animated pedagogical agents to reason about how their emotional state ties to patient condition and user progress. Initial results show that users of the game exhibit a significant shift in intention to call 911 and avoid delay, that multimedia versions of the game foster vividness and memory retention as well as a better understanding of both symptoms and of the need to manage time during a heart attack event."*

As we will discuss, the challenges of serving applications that need to have high fidelity, validation, traceability, and performance lead to formidable challenges for the current technologies. Before we tackle these challenges, it is worthwhile to consider the lessons learned from one of the earliest uses of VWs for real-world applications—education and training.

Summary and Critique of Educational Applications

There has been a great deal of research on the use of technology for education (Soloway, 1993; Forbus & Feltovich, 2001; Psotka, Massey, & Mutter, 1998); the purpose here is to simply highlight some of the work on education in virtual worlds in order to illustrate where virtual worlds must develop. The educational tools community has always been interested in collaborative technology. As Wolfgang Gerteis and Joachim Schaper state in a paper addressing issues in e-learning, it is the "powerful combination of instructional design and collaborative elements [that] offers new capabilities to build the bases for new content and services" (Gerteis & Schaper, 2003). Most of the initial VW educational applications were geared towards older elementary through high school students and revolve around two uses: enhancing literacy skills (Viau, 1998a, 1998b, 1998c, 1998d) and analytic skills through simulation in the same environment (Viau, 1998a; O'Day et al., 1998; see also Spohrer, EOE Web site). An early example of educational MUDs was SolSys, the Solar System Simulation, which originated at CONTACT VI in 1987 and was further developed as an intercollegiate curriculum by Reed Riner (Riner & Clodius, 1995) at Northern Arizona University as an honors course in Anthropology and Engineering. Since 1990, it has included student teams from many colleges and universities around the globe.

Solsys allows students in teams to build their own colonies in a simulated future human community in space. Teams communicate via Web sites, Internet e-mail, and a Multiple User Domain (a TinyMUD), under the direction of local faculty advisors and a board of professional consultants in varying fields ranging from the social to space sciences. By building the colonies in VWs, the students not only must draw on all disciplines of knowledge, but also demonstrate this knowledge to teachers and other visitors to their sites. There the visitors can chat with the creators about the social, biological, artistic, and physical ideas represented in the VW, tour their cities, and even "talk to Martians" (Solsys Web site).

Barry Kort was another pioneer in the early use of MUVEs for education (Hughes, Walters, & Kort, 1994). Kort created MuseNet; the Multi-User Science Education Network, is a system of computers in the domain musenet.org providing access to *Educational MUSEs (multi-user simulated environments),* such as MicroMUSE and MariMUSE. MicroMUSE is a multi-user simulation environment developed in the 1990s at Bolt, Beranek, and Newman (BBN). The system features explorations, adventures, and puzzles with an engaging mix of social, cultural, recreational, and educational content. It won a 1996 NII Award for "Pioneering innovations in children's education via the Internet" (Kort Web site). In Kort's words:

> *"MUSEs are multi-user text-based virtual communities accessible via the Internet. They derive from popular text-based adventure games such as Adventure and Zork. But MUSEs support real-time interaction among many participants who collaborate to build their own world. Thus they support the constructivist model of learning, in the spirit of Dewey and Montessori. More than just multi-user programming environments, MUSEs foster a strong sense of community among participants."*

There are a large number of educational MUVEs with wide-ranging topics (see TECFA Web site for a good starting point).

During 1993-1997, the author made the development of VW educational and training applications one of the major thrusts for the very large government program, DARPA Computer-Aided Education and Training Initiative (CAETI), which involved more than 300 U.S. private companies, universities, and research institutes. Some of the educational tools developed under CAETI (Bellman, 2001) that were of special interest to virtual worlds were new types of embodied intelligent utilities and agents that helped individual users (librarians, guides, and tutors) or conducted support activities across the world (evaluation agents).

Some of these new tools also helped tailor resources to an individual user. Saraswat (who contributed to CAETI while at Xerox PARC) points out a number of educationally interesting MUVEs and features of them. One of the author's dreams for CAETI was that eventually learners will have wonderfully rich places where they can have both the known advantages of one-on-one tutoring (see also Bellman, 2001), via intelligent companion tutors, and the benefits of the collaborative and social interactions within these places. In order to realize this dream, we would have to develop tutors who could "co-experience" (Bellman, 1994) the VW with the learner, and dynamically adjust its pedagogy and content to adjust to these experiences. This would mean that the tutor, rather than being an oracle with all knowledge built in beforehand, would instead have to have the sophisticated capability of knowing enough about its type of knowledge and pedagogy so as to recognize new instances of it or to generalize it to the circumstances at hand. It would also have to be able to integrate such experience into its stores of pedagogical examples, content matter, and so forth.

Although project-based and collaborative technologies were of deep interest to the teachers in the CAETI K-12 testbeds, there was little curriculum material developed to support it. Hence most of the educational MUVEs were used to support literary skills. The original virtual world applications' focus on literacy and programming skills remain the most enduring ones and are still active today. For a good example of this, see Viau's "world building" courses (Viau, 1998a, 1998b, 1998c, 1998d, 1996.) Because building (which in the early VWs includes both authoring and programming skills) can provide such a powerful incentive for children to participate in constructivist, participatory, and collaborative educational strategies, it remains an excellent focus for educational applications. However, hopefully with the development of more and more environments with significant simulation capabilities, VW educational applications will continue to broaden and VWs will become a learning space where children can not only show their understanding in the descriptions of the worlds they create, but also in the behaviors that occur within those worlds. Many of the CAETI VW projects were geared towards enhancing VWs in that fashion.

Some of the projects for collaborative "learning spaces" included a team (Intermetrics, Yale, University of Illinois at Chicago) developing a multimedia math/science world called "Wyndhaven." Another team (Xerox PARC, Phoenix College) created several impressive virtual communities (including ones that combined school children with senior citizens) and focused on inserting into these environments better simulation, construction, experimentation, and reflection capabilities for the learners. For example, in one of Xerox PARC's projects, (1998):

> *"The context for discussion is Pueblo, a MOO-based, cross-generation network learning community centered around a K-6 elementary school. The development of practice in Pueblo draws upon teachers' and students' experience with semi-structured classroom participation frameworks—informal structures of social interaction which foster certain ways of thinking, doing, and learning through guided activities and conversations. We have translated several familiar frameworks into the Pueblo setting, using the classroom versions as models to be adapted and transformed as they are aligned with the affordances of the MOO. We identify four design dimensions that have emerged as particularly interesting and important in this process: audience, asynchrony and synchrony, attention and awareness, and prompts for reflection."*

In their paper, they further discussed "the relation between MOO affordances and design choices and provide examples of successful and unsuccessful alignment between them." Particularly important has been Xerox PARC's emphasis on the role of self-reflection in a constructionist environment, for example, it is not enough to get a child to do something; to learn best they need to reflect on what they have understood.

The SUMMIT program at Stanford (headed by Parvati Dev) created a number of distributed multimedia MUVEs geared towards both medical education and support groups related to health. SUMMIT also had methods for authoring multimedia content that passed one of the more difficult tests, for example, the doctors actually used them to create materials for their courses. The ExploreNet project (University of Central Florida) used older children to help create the educational worlds for younger children, thereby benefiting both age groups. Their projects emphasized both social science and literary curricula. GMU taught programming courses in a C++ MUVE. The SAIC corporation and University of California, San Diego worked on developing intelligent agents for virtual worlds, including a "librarian" that interacts with students to help them find information. During the course of the program, gradually more and more tutors and agents were introduced into the MUVEs, one of the successful examples of cross-program integration among CAETI projects (Suthers, 1998).

In CAETI, a number of educational simulations were developed, although many of them were not yet available within the VWs. These efforts are particularly important in deepening the level of content in educational technology in general and in VWs specifically. Examples include GMU and Shodor that provided several impressive simulations on a number of topics, including galaxy formation and the mathematics of fractals, AMPHION, developed by the University of

Wyoming and NASA-Ames to visualize space objects for collaborative use, and "Function Machines," a graphic programming language for math/science by BBN. Similarly, several of the projects developed impressive multimedia content with associated analysis tools. A good example of this is the Intelligent Multimedia/Thinking Skills project (GMU) that built instructional modules for social studies, with an online coach/tutor and tools that support higher-order thinking skills and excellent source materials for a module on slavery in the U.S. Other collaborative projects deserve, like all these projects, a chapter in themselves. Hands on Universe (Lawrence Berkeley Laboratories) in collaboration with the Lawrence Hall of Science, TERC Inc., Adler Planetarium, and Yerkes Observatory, as well as an international network of educators and astronomers, allowed high school students to conduct real science in an apprentice role and to direct the use of large telescopes accessed over the Internet. Other projects, such as UNC's collaborative Web applications, and Guzdial and Kolodner's important work at Georgia Tech on learning through design and complex problem solving, were developing both fundamental new methods (including formal mathematical methods) for supporting collaboration and new theory for understanding the collaboration of small groups of learners (Bellman, 2001).

Strengths and Weaknesses of Educational VWs

We can summarize the advantages of VWs for education as follows: First and foremost a VW can be an excellent constructivist learning environment (Papert, 1980). Part of what makes the MUVE better than other constructivist environments is that we potentially have better control over the learning context (how it is situated), we can observe and record all behavior for further analysis by the teacher and by the student on all constructions and learning exercises. We can add community as Bruckman emphasizes with MOOSE Crossing, have persistent learning environments available to the student over years, and provide not only peers and interested adults, but also distant experts.

Second a VW can support collaborative learning and project-based paradigms, including ones that persist over years intermingling older and younger students, and bringing in community and professional participants. Further one can study team building for children and adults. For example, at the Franklin Institute (Testani, Wagner, & Wehden, 1999), a VW was developed and tested to provide training in team building for small businesses in the wider Pennsylvanian area. Team training and collaborative skills have been increasingly asked for and emphasized by businesses and professional groups as necessary in the modern work world where complex projects demand collaboration across groups, companies and countries, and cultures.

Third, as hoped for with all educational technology, VWs potentially allow a large number of students individual attention and support doing work geared towards their individual interests and talents at a potentially reasonable (and supportable cost for poor schools) and in a scalable fashion. In addition, as emphasized by the military for many years, it also allows individuals to learn about and train on systems that would be too costly, dangerous, or unwieldy in a school setting.

Fourth, VWs provide a computer-mediated environment that can be easily instrumented to collect all behaviors and interactions in their context (e.g., at what time, doing what with what reasons, and interacting with whom.) This new source of comprehensive data can potentially be analyzed in order to evaluate new practices; or the concept of operations (CONOPS)—how people really work with different tools and policies—can be studied. It can be used to allow new ways of evaluating learners who are engaged in a constructivist paradigm, addressing a way of having an individual exhibit their own contributions while still be in a collaborative context. It can lead to early publication and evaluation, a chance for students to participate in apprenticeships with mentors and experts in science and other professional areas.

Lastly, one can potentially be in an educational environment with an unlimited ability to expand in knowledge—both in range, amount of content, and depth of content. Unlike a textbook, a VW has no limit to its expansion. Not only can formally validated course content be presented, but also peers and professionals can contribute informally and formally to the knowledge available in the environment. With the Web and mobile agents, it is easy to envision how VWs in the future will dynamically create rooms and objects based on source information found on the Web and other networked resources. As noted above, these worlds would also be enriched with tutors who co-experience the world with a learner and help them reason about and reflect on the content they are exposed to.

As exciting as this potential for VWs is, the reality of the current applications show how far we need to go. The depth of content and pedagogy remains largely shallow and spottily available within a given class curriculum. This results in a lack of integration into the curriculum over the school year, much less through the student's educational years. This means that although one may have excitingly suggestive applications that demonstrate some of the potential for educational VWs, one will not see as good or lasting educational benefits. One of the reasons that the VWs are so limited within the curriculum is partly the lack of authoring tools allowing content to be developed more quickly—by experts and teachers and most importantly by students. Another reason is that there is too little integration between the VWs and real-world devices, such as sensors in chemical lab beakers and monitors on physical devices in a physics lab. Nor is the virtual world integrated into the real-life classroom; it tends to be an after school or separate student activity. This is partly due to the lack of computers, but it also mirrors all the challenges we discuss in the next section for bridging the gap between virtual worlds and real worlds. Again, there are several good examples of computer-based support integrated with experimental apparatus and real classroom practice, but it is largely spottily available when available at all.

It therefore is no surprise that the psychological integration of the user's actions and experiences in the real world and the virtual world is poor. Basically it is up to the learner, with support from mentors, teachers, and peers, to make the most of the VW and deal with their real world at the same time. Part of the problem is the lack of task analysis and clear functional definitions for the educational VWs. We need to understand the educational goals of a VW when we design it or when we bring it into the classroom. Lastly it is difficult to generate the educational material for VWs. It is difficult to design and author the scenarios that guide or constrain a learner in the VW, and to generate and to parameterize qualitative aspects of both the scenarios and the environments. Along with this, we need much better authoring capabilities for non-text environments and for new multi-sensory interfaces and systems. Again there is a lack of scientific research that would help inform what educational goals should be accomplished with what types of visual, haptic, or auditory interfaces. Also, we have experience and intuition, but not theory to inform how we distribute learning capabilities and information in the VWs. This is a continuing issue for all educational technologies, first pondered and addressed in intelligent tutors work. We have learned how to write a book and organize the materials for a text, but what are the best ways of organizing materials in an interactive system? This is a question of pedagogy and epistemology as well as computer science.

To make the above weaknesses of current systems clear, let us look at some of the lessons learned from the constructivist applications in educational VWs, partly because some of these educational applications have had some evaluation studies.

The creative building permitted in text-based MUVEs comes to the fore educationally, feeding very much into participatory and constructive educational paradigms. The early educational applications of VWs were characterized by an enormous creativity and enthusiasm, typified by, for example Bruckman, then at MIT (Bruckman & De Bonte, 1997), in her work on MOOSE Crossing:

> *"MOOSE Crossing is a text-based virtual reality environment (or 'MUD') designed to give children eight to thirteen years old a meaningful context for learning reading, writing, and computer programming. It is used from home, in after-school programs, and increasingly as an in-school activity. To date, it has been*

> *used in five classrooms. This paper compares its use in three of those classrooms, and analyzes factors that made use of MOOSE Crossing more and less successful in each of these contexts. Issues highlighted include access to computers, existence of peer experts, free-form versus structured activity, and school atmosphere."*

However, as many have noted:

> *"CSCL environments can help to foster and support collaborative learning in schools. However, our observations indicate that a computer-supported cooperative learning tool cannot on its own cause a fundamental cultural shift. Factors that affect the success of MOOSE Crossing in the classrooms we observed include the accessibility of computers, school atmosphere, and teacher attitudes towards collaborative learning."*

However, it has been difficult to develop the full potential of VWs for education partly because the educational value of VWs still needs to be rigorously evaluated. Bruckman, one of the early developers of educational VW applications, has been one of the few to consistently study how well her educational applications of VWs met educational goals in the classroom.

Tuman (1992, pp. 41-43) makes a similar point, although what he discussed at that time was the impact of hypertext on literacy; however, it is easy to generalize the concerns he and others raise to VWs and games. In his story of Jane sitting down to read *Great Expectations* on a hypertext system, he compellingly raises the issue of how computer-based medium might impact a user's search for depth and ability to stick to long works, and so forth:

> *"Students in Jane's situation, in much larger numbers than any of us care to admit, have long turned to literary guides, simulacra of the texts—the most visible being the boldly striped Cliff's Notes—to provide them an easier path through (and, at least as often, around) complex and long literary texts. What Bolter does not consider in his discussion of the experience of reading Joyce's 'Afternoon' is what happens when the story is not a self-contained fictional universe, read by someone interested in having a rich aesthetic experience, but only a tiny part of a vast hypertext network. One where harried and information-driven readers, instead of spending a few hours exploring Joyce's*

> *constantly shifting story, can find out the least they need to know more readily by clicking on screens containing background information about Joyce, interactive fiction, and the story itself. The point at issue here is not whether the hypertext environment can support the level of aesthetic reading associated with print literacy, especially for those fully acculturated into the world of print—but how different the experience of reading the most aesthetically complex hypertext may be for 'readers' in the future who will be fully acculturated into an electronic world, possibly ordinary students of the next century who have no sustained experience of print. Just how likely is it that people for whom reading has become an act defined largely in terms of using the computer either to access needed information on demand or to be entertained by the slam-bang integration of 3D graphics and CD sound will be willing—or able—to sit before a terminal patiently selecting the paths in a single author-designed hypertext in order to have something akin to a traditional literary experience?"*

Clearly there will always be students who rise above any impediments to intellectual development, nonetheless it is clear that the individual differences must be understood and addressed. Hence in MOOSE experiments, some of the community learned a great deal in programming skills and some not (Bruckman, 2003). The same sentiment has been learned in Pueblo and indeed every VW application as well as every classroom. So the reader might ask why we are holding the VW to a higher standard than the average classroom. The answer is simply that we know what the problems (if not the solutions) are in real classrooms and we do not know the implications in VWs.

Virtual worlds take advantage of human minds, and hence there are individual differences and differences within an individual depending on their mood, motivation, and so forth. Hence, Bruckman (2002) discusses AquaMOOSE 3D, a graphical environment designed to support the exploration of 3D mathematical concepts. In a classroom comparison study, they were disappointed with the results between the use of AquaMOOSE and traditional curriculum:

> *"Despite this initial motivation to use AquaMOOSE, many students in the experimental class were disappointed with the software. After the study, students in the experimental class commented on the lack of action in AquaMOOSE and the imperfect models and environments that we used. One student, in response to a question about polar coordinates, said, 'I don't remember anything but the ugly little fish.' By telling the students beforehand that they were*

> *going to be using software that was game-like in nature, we set the AquaMOOSE software up to compete against commercial video games. As can be seen by the intense competition present in the commercial video game market, the students' high expectations are difficult to meet. For example, to create the massively multiplayer online role-playing game Asheron's Call, Turbine Entertainment had a staff of over 30 people working for four years (Ragaini, 2000). A research prototype made by a few graduate and undergraduate students and one faculty member clearly cannot compete."*

As can be seen from both of these articles, how to evaluate value is key problem for educational applications and indeed for all application areas. The difference is that in entertainment one can ask the user, "Did you enjoy this?" or watch them vote with their feet, but in education, one needs to have the goals for using the VW match their desired real-life behavior. In medicine and other applications, it just gets increasingly serious.

All of these are general challenges not just for the educational use of VWs but for all 'real' applications.

Challenges of "Getting Real" in Virtual Worlds

The reason for this chapter is not to describe the challenges of producing believable, attractive, and desirable social or gaming virtual worlds, although some of the comments will impact such current and much emphasized uses of VWs. There is a lot of active and good work describing how to do both state of the practice and state of the art in VWs (see Vince & Earnshaw, 1998; www.there.com) for the purposes of social worlds and role-playing games. For example, Vince and Earnshaw (1998) offer a number of useful chapters describing how the current technology is being used (on the use of VRML or ATM networks for example), with other chapters describing research in such areas as virtual reality interfaces. Similarly, Bartle (2003), one of the early developers of MUDs, describes the design concepts behind a wide range of games from the earliest MUDs to the current online multimedia role-playing games. But again this comprehensive book is geared towards those interested in designing games.

Many have noted in the research and development community that creating real-world applications in a virtual world is a matter of having sufficient detail. If there was a VW of sufficient richness, then one could truly use it as an exploratory scientific tool for many things:

1. Disease interventions and crisis interventions
2. Discovery and understanding
3. Analysis and engineering (building bridges, building new drugs)
4. Inventions (social and physical)

The hard question is what detail is needed and how to get the detail into the VW. The answer to this hard question is answered differently depending on whether one emphasizes the VW as a set of models, as a set of interfaces between the human being and various data sources, or as a set of human and machine processing capabilities and effectors. At first details may sound as if the problem is data, but it is not. Rather, it is data and process and an analyzable setting that allows critical events to be validated. It is also detail at all levels of a VW object, actions that characters can do, and the settings or contexts within which these objects and actions occur.

VWs as Models

VWs are in many ways models; they have representations for the human user and other agents. They have both static to dynamical models of the environment or the setting, they have both static and dynamical representations of any resources or objects in the world, and to add to complexity they can contain objects that are meant to be classical models or simulations of something in the world (such as players playing a hockey match or a tank firing and so forth). Darema (2002) calls them dynamical data-driven models because they have humans and potentially other ways of incorporating live data feeds that constantly change the "models" in the VW. Some of the greatest challenges to becoming real-world-capable VWs are to, on one hand, gracefully integrate the 'live' and modeled parts of a virtual world and, on the other hand, carefully distinguish—monitor, trace, and analyze the impacts—from what is 'real' input or behavior and what is from models (and therefore dependent upon modeling assumptions). For example, a surgeon who is about to cut into a patient in the manner suggested by a projected three-dimensional rendering of the tumor is very cognizant of the differences between the modeled tumor and the live patient beneath her knife. In the best case the augmented reality system will support careful analysis of the continual correctness of the projected image and tumor model based on the feedback occurring during the surgery.

VWs are like models: if they're good enough they can serve many purposes. When many people discuss scaling up VWs to real-world applications, they are often considering the issue in terms of how realistic the virtual world models are. For example, in a training exercise, it may be a matter of the realism of the terrain model (Darken & Goerger, 1999). In many gaming environments, it is the realism of the textures in the world or in the fabric available to avatars' costumes or in the bouncing of one object against another.

However, it soon becomes apparent that there are almost an infinite number of details that can be included in a virtual world. This then requires not only modeling skill, but modeling wisdom; it also requires new ways of integrating models since there will be many, many models governing different objects, and their interactions and their contexts or settings. It also has motivated many to move away from just modeling to a combination of modeling and live data cannily projected into the world. For example, many virtual museums no longer try to build models of their layouts and objects; instead they use static pictures taken from many angles, pasted into the virtual world and shown to the user based on where their avatar is located in the VW.

VWs as a Set of Human and Computational Processes

There is an intimate relationship between the computational capabilities available inside the VW and VW models, but it is worth bringing out some of issues separately. The most formidable types of models needed in a VW are not the ones used to give a realistic appearance to an object, but rather have the correct dynamics and behavior from that object. This can lead to a combinatorial nightmare of potential modeling interactions and side effects. Therefore, unless we are trapped and limited into analyzing ahead of time every detail's relationship with all other details, we need much more profound "meta-models" that can handle relationships among other models. Lastly, to handle the building of VWs, this modeling cannot be done all at once. Hence the emphasis here is on traceability and analysis to support an incremental process.

As in all models, some factors, features, and attributes will be as realistic as possible, and others need only be dummy variables or placeholders. If VW models never got beyond entertainment and some teaching and training apps, that would be a valuable contribution enough. However, there are some very good reasons for desiring more from VWs; the complexity of our modern era requires integration places. We need places that are infused with analysis capabilities. Due to the complexity they also need to be incrementally done and hence deeply traceable. They must have explicit criteria for desired realism or behavior and not just appearance of correctness.

That is, not just art, but rendering in the sense of the scientific side of art and craft. Pictures (museum sites) only make realism in appearance and not in behavior. To get realism in behavior, we need three things: more realistic input (such as data-driven systems), more realistic models or better models for a given function, and more realistic ways of driving output (not just text or pictures to a user, but for example instruments on a screen). A VW must eventually be alive with processes that go on regardless (Simcity like) and have dynamics, not just responsive actions.

But processes are not just within models (such as simulations); rather they are part of the challenge and the means of integrating the VW with the real world, and as such they must be able to cross from real world into VW as processable data streams and information flows, and from VWs to real world as electrical signals that cause automated devices to move, focus, release chemicals, and many other such actions. For example, how is the real live video on a fireman's hat to be fed into the virtual world that is being used to monitor and control the force's reaction to a fire? Or taking this same analogy, how does the VW's internal simulation of the fire and wind conditions result in changes in how the sensors are focused or water is being dropped from UAVs?

VWs as Interfaces

However VWs are not just the models underlying the world, they are also the means by which humans and other active processes interface with any of the content or capabilities of the virtual world. Especially now VWs are no longer just inside a box—they can include interfaces that run robots and factory floors, respond to events within a smart room, or provide live data projected into the behavior of virtual objects. Hence in the rest of this section, we will be discussing several examples of research that emphasizes new interfaces—both enriching the VW by the input into the world and the outputs from the world. To get real requires not only content inside, but also the ability to interface with the real world.

Real-World-Capable Challenges

Much of what we are going to discuss here are ways of filling the virtual world—but it is not enough to say it needs real content and sources of content. Rather it must be content placed in the virtual world in such a way that one can analyze it, track its usage, and experiment with it. This latter property will change virtual worlds from massive games with billions of homemade objects (even bought and sold) by users to a sufficiently realistic environment for medicine and other high-fidelity uses.

Based on the discussion so far, we can now summarize the three major challenges to real-world-capable VWs as realism, scalability, and analyzability at *all* levels of the VW—from the management of the network byte streams and sensors and effectors, to the processing capabilities and models inside the VW, to the impact on the psychology and culture of the human users. Virtual World users similarly demand increasing realism.

Realism

Howard Rheingold (1992) credits Sara Kiesler (1986) with noting "…that the word 'phony' is an artifact of the early years of the telephone, when media-naive people were conned by slick talkers in ways that wouldn't deceive an eight-year old with a cellular phone today."

As Mike Macedonia points out, one of the drivers for realism in virtual worlds has always been military training and planning needs (2002). He also points out that VWs are just a continuation in a long tradition of innovations starting "in 1887, when McCarthy Little, a military strategist at the Naval War College in Newport, R.I. devised a war game using miniature battleships on maps. Around the same time, the German Army developed the board game Kriegspiel. Such games soon spread to all the world's major armies and became critical in military education and planning. Greater realism came later. Virtual flight was the brainchild of Edwin A. Link, who in 1929 invented the Blue Box, an instrumented cockpit simulator that used pneumatic pumps to recreate an aircraft's motion" (p. 36). He goes on to discuss the sophistication of course added by computers: "By the 1980's Link's idea had been wholly transformed into digital flight simulators complete with 3D graphics to convincingly reproduce scenery, high-resolution displays, and moving platforms with 6 degrees of freedom." These military environments were early innovators in how to bring in realistic details. For example, Paul Debevec in the 1990s at UCB helped develop image-based rendering, which generates images directly from photos rather than building them graphically. Others worked diligently on how to bring in sound, vibration, motion, and so forth (Macedonia, 2002). There has been a growing amount of work now on how to have avatars display more realistic movements and expressions, including emotion (Trappl, Petta, & Payr, 2003; Perlin, 1995; Perlin & Goldberg, 1999; Johnson, 1999). However, in the author's experience in observing both flight simulators and tank simulators, it is interesting to note that as much as the sophistication of the interfaces has increased, there are still formidable challenges in providing the type of role that the human trainer plays in these simulators. For example, in the author's observation of the old flight simulators used for Naval fliers in Beeville, Texas, it was fascinating to interview the human trainer as he presented the student inside the cockpit trainer with poor weather conditions or malfunctioning components. He was as busy as the student. When the author asked one such trainer questions on why he presented turbulence at one point, he replied that the student was doing so well, it was time to stress him a little. Intuitively and carefully, the skilled trainers made the complex technology an educationally meaningful experience for the student. Unfortunately, this required a single trainer working in a very expensive simulator on the experience of one student. The question before us is how to somehow make computational and scale up this type of quality training and attention for numerous students—for even a battlefield of such students (Macedonia, 2002; Wallace & Sollenberger, 2001).

Mixed Reality: Telerobotics and Augmented Reality

From the earliest examples of controlling a coffeepot through a networked computer, it has been clear that the issue of VWs is not just a matter of how to get information into the VW, but also how to get real effects out of the VW into the real world. Paul Milgram's group in Canada has been doing leading work for many years in the area of augmented reality both from the perceptual issues to the control of effectors in telerobotics (Milgram et al., 1995). "The fields of artificial reality and conventional telerobotics share many common technological challenges." In their paper, they discuss "the concept of applying techniques of virtual environment simulation to address some of the challenges of remote manipulation of teleoperated systems in unstructured environments, with a focus on remote excavation."

Not Just Inside a Box

To do real work is going to require the ability of the virtual world not to be thought of as just in a box, but rather as something that accompanies one into the surgical ward (Parvati Dev, private communication), the classroom, and everywhere else.

It is important to not only think of virtual worlds as inside boxes, but rather as encompassing any highly computationally underwritten environment, for example, caves, to highly connected people in partly RL/VL spaces.

There are many recent experiments on how to incorporate mobile devices into e-learning (Berger, Rainer, Holger, & Klaus, 2003). In the Berger work, the tool "is embedded in an e-learning and m-learning environment at the University of Regensburg, which allows its functions to be accessed not only from a Web browser, but also from a personal digital assistant (PDA) or any phone that supports the wireless application protocol (WAP). In their paper, they discuss the advantages to learning groups and the "benefits gained by supporting them in mobile scenarios."

Obviously there needs to be many different kinds of collaborative support. However, for the purposes discussed here, what is missing from this work is how to gain a global perspective on what is being done and accomplished by the mobile participants. Therefore it is important to take some of the collaborative research on mobile devices and integrate it with virtual worlds, potentially in the helpful strategy of Nessie.

Analyzability

Because models of what humans and cultures are will become the silent underpinnings of our brave new world, we must develop the means by which those assumptions and biases can become known and analyzable to us.

We have already discussed that one of the most important qualities of MUVEs is that people are allowed the freedom and richness of word pictures. However, the advantage of imagination creates the equally strong disadvantage of increasing the challenges for analysis and experimentation in virtual worlds. That is, much work seeks to increase the type of applications that require high fidelity, analyzability, and traceability (e.g., what was responsible for what effect in the environment). How then are we to analyze, understand, and control for certain critical effects when the virtual world is to large extent "in the mind's eye of the beholder?" We need cognitive studies and experiments in VWs, and analysis at least equivalent to the early days of educational technology.

The ability to create an analyzable base within the VWs is critical to its ability to be developed and scalable for into real-world-capable VWs. Such an analyzable base will allow developers to continually assess the value of new capabilities or features of the world, and with that essential feedback refine the world and continually engineer it. Such analyses are also critical to supporting psychological scalability by helping the human user gain perspective over the enormous complexity of such environments, and have the means to address questions on how capabilities are being used, the impacts of these capabilities, the causal and correlative effects of different world, agent, and object characteristics. Lastly, the ability of this last point—to have the means by which humans can understand the VW and understand its impacts means that we can provide the crucial means for humans to take responsibility for what occurs in the virtual worlds. Responsibility is the flip side of the ability to evaluate. We will determine based on the evaluation.

Need to Take Responsibility for the Content and Capabilities in Virtual Worlds

Real-world-capable virtual worlds require many things, as we have discussed. But one issue is rarely brought up both as a motivation for the challenges described below, but also as a responsibility. Although it can be annoying or harmful even if social MUDs are not well-run, as we approach these critical real-world applications, it becomes increasingly critical that we build environments that not only can support the functions we care about, but for which we as developers can take responsibility. This will increasingly become a legal requirement; it already is an ethical one. This means that we must not only be able to support the right things in a VW, we must be able to prove that we are able to support the right things.

One of the things that has long been problematic in computer-based applications is the hidden power of the programmer to determine what is the real use of content within a system. As Tuman, who is not a cyberspace enthusiast, noted quite a while ago (1992): “Truth is still above the masses, but it is now conceived, not as something rarefied or spiritual, but as a trade secret at the top of a corporate pyramid—what separates holders of ‘truth’ from the people below is not knowledge but institutional greed and power.” Meanwhile, instead of Faustian man, committed to an endless, solitary quest for knowledge, the new age, Bolter speculates, is marked by the programmer, someone whose work at every step makes him or her aware of the physical limits of electronic time and space. The programmer, Bolter contends, does not make bold new discoveries but instead subtly manipulates finite parts within a finite world: “He remains in the confined logical universe of his machine, rearranging the elements of that universe to suit the current problem” (Bolter, 1984, p. 223).

Again the point of bringing this type of discussion in is not to just give balance to communities who do not agree or argue against the e-world, Internet communication, and of course virtual worlds; it is to raise several critical issues:

> *First, who will be responsible for the material in worlds (in social MUDs it is both the owner of the database and often a user group)—who and how to evaluate it in a multi-user multi-created world?*

Second, for some time now a number of us have been concerned about the hidden power given programmers in too many domains for which they do not have the knowledge. This is a problem very familiar to the modeling and simulation community, where the programmer often makes decisions for the sake of programming ease or computational efficiencies that may have profound impacts on the validity of the models involved. How in virtual worlds do we elevate this authoring responsibility. Tuman's remarks point to something that has been long discussed in the modeling community, the problem of the programmer's power—in most programmed environments, whatever knowledge and actions exist end up reflecting both for good and bad the perceptions, understandings, and knowledge of the programmer about that domain. This is why we must in fact create a very different paradigm in MUVEs.

Third, how do we bring depth and real content into an environment that encourages rapid information retrieval, rapid movement to the right place, and rapid discovery of interaction possibilities? In social MUDs, this is as simple as the power to not have to walk through spaces but to go immediately there. We want the advantage of the virtual worlds except where process—and the time it takes, the space it travels—is a meaningful part of the process. Hence in a surgical virtual world, we certainly do not want young surgeons practicing procedures by skipping to the end!

The point here is that the challenges are not all computer software and hardware technical. Rather the challenges include mathematics and social sciences, and how to use these environments to create something fundamentally new—not just a shareable space, but one that is profoundly analyzable; one that helps gather the data that will be meaningful within it.

Clearly, developing the means by which we can evaluate virtual worlds is a critical and urgent need. Luckily, aside from early pioneers such as Peg Syverson and Amy Bruckman, there are increasingly more and more researchers doing so (Bouthillier & Shearer, 2003; Lau, Adams, Dew, & Leigh, 2003).

Conclusions

Virtual worlds have enormous potential, not only in specific application domains, but in changing the way that researchers and developers are able to develop, integrate, monitor, analyze, and impact the complex system of humans and artifacts. However, in order to develop such systems, we must scale up in three very different ways: numerousness, variety, and what we are terming "psychological scalability." To do so we believe that we must 'open up the box' and integrate VWs into the real world via a variety of multi-sensory interfaces, live data feeds, telerobotics, and other effectors for performing real work from the virtual world. However, in order to scale up in these ways, we must develop VWs that have much better ways of analyzing and evaluating what has occurred within the virtual world, and much better ways of mapping the causative and co-occurring relationships among the many complex attributes of agent, world, and objects. To help develop testbeds for such VWs, we have been experimenting on both the types of experiments one can conduct within a virtual world and the necessary adaptive and flexible infrastructure for doing so. Eventually these testbeds will lead the way to Virtual Worlds that help us do real things.

References

Azuma, R. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 355-385.

Bajura, M., Henry F., & Ryutarou O. (1992). Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. Proceedings of SIGGRAPH '92, Chicago, Illinois, July 26-31. *Computer Graphics*, *26*(2), 203-210.

Bartle, R. (1990). Interactive multi-user computer games. *MUSE, Ltd.* Retrieved March 9, 2004 from: *www.cpsr.org/cpsr/sociology/mud_moo/mudreport.txt*

Bartle, R. (2003). *Designing virtual worlds.* Prentice-Hall.

Bellman, K. (1994). Playing in the MUD: Turning virtual reality into real places. In R.J. Seidel & P.R. Chatelier (Eds.), *Proceedings of the NATO Conference on Virtual Reality: Training for Tomorrow,* Portsmouth, England, February. NATO Defense Research Group #16 on Training and Training Technology.

Bellman, K. (1997). Sharing work, experience, interpretation, and maybe even meanings between natural and artificial agents. *Proceedings of SMC'97: The 1997 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 4127-4132), Orlando, Florida, October 12-15.

Bellman, K. (1999). Emotions: Meaningful mappings between the individual and its world. Paper presented at the *Workshop on Emotions in Humans and Artifacts,* Austrian Research Institute for Artificial Intelligence (ÖFAI), August 13-14.

Bellman, K. (2001). Building the right stuff: Some reflections on the CAETI program and the challenges of educational technology. Chapter 12 in K.D. Forbus & P.J. Feltovich (Eds.), *Smart machines in education: The coming revolution in educational technology* (pp. 377-420). AAAI Press.

Berger, S., Rainer M., Holger N., & Klaus J. (2003). Mobile collaboration tool for university education. *Proceedings of the 12th IEEE International*

Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria.

Bouthillier, F., & Shearer, K. (2003). Assessing collaborative tools from an information-processing perspective: Identification of value-added processes. *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'03), Linz, Austria.

Cabri, G., Leonardi, L., & Zambonelli, F. (2000). Mobile-agent coordination models for Internet applications (pp. 82-89). Washington, DC: IEEE Computer Society.

Clodius, J. (1995). Computer-mediated interactions: Human factors. *Proceedings of MUDshop II*, San Diego, California, September. Retrieved March 20, 2001 from: *www.dragonmud.org/people/jen/keynote.html*

Curtis, P. (1992). Mudding: Social phenomena in text-based virtual realities. *Proceedings of DIAC'92: 1992 Conference on Directions and Implications of Advanced Computing,* Berkeley, California.

Curtis, P., & Nichols, D. (1994). MUDs grow up: Social virtual reality in the real world. COMPCON 1994 (pp. 193-200). Retrieved from *citeseer.ist.psu.edu/curtis93muds.html*.

Darema. F. (2002). NSF, dynamic data applications systems technology for crisis management. *Proceedings of VWsim '02.*

Darken, R., & Goerger, S. (1999). The transfer of strategies from virtual to real environments: An explanation for performance differences? *Proceedings of VWsim '99: The 1999 Virtual Worlds and Simulation Conference.*

Dautenhahn, K. (1998). The art of designing socially intelligent agents: Science, fiction, and the human in the loop. *Applied Artificial Intelligence, 1*(7).

Dautenhahn, K. (1999a). Socially situated life-like agents. *Proceedings VWsim'99: The 1999 Virtual Worlds and Simulation Conference,* part of *WMC'99: The 1999 SCS Western Multi-Conference* (pp. 191-196). San Francisco, California, January 18-20.

Dautenhahn, K. (Ed.). (1999b). *Human cognition and social agent technology*. Benjamins.

Dautenhahn, K., & Nehaniv, C. (1999). Living with socially intelligent agents: A cognitive technology view. In K. Dautenhahn (Ed.), *Human cognition and social agent technology* (Chapter 16). Benjamins.

Drascic, D., & Milgram, P. (1996). Perceptual issues in augmented reality. In M.T. Bolas, S.S. Fisher, & J.O. Merritt III (Eds.), *SPIE Volume 2653: Stereoscopic displays and virtual reality systems* (pp. 123-134). San Jose, CA.

Feiner, S.K. (2002). Augmented reality: A new way of seeing. *Scientific American, 286*(4), 48-55.

Foner, L.N. (1997) Entertaining agents: A sociological case study. In W.L. Johnson (Ed.), *Proceedings of AA'97: The First International Conference on Autonomous Agents* (pp. 122-129), Marina del Rey, California, February 5-8. See also: *foner.www.media.mit.edu/people/foner/agents.html*

Forbus, K.D., & Feltovich, P.J. (eds.). (2001). *Smart machines in education: The coming revolution in educational technology.* AAAI Press.

Gallagher, W. (1993). *The power of place: How our surroundings shape our thoughts, emotions, and actions.* Harper Perennial.

Gerteis, W., & Schaper, J. (2003). Instructional design for collaborative e-learning. *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'03).

Gordon, A., & Hall, L.(1998). *Collaboration with agents in a virtual world. Proceedings of the Workshop on Current Trends and Artificial Intelligence in Education, Fourth World Congress on Expert Systems* (pp. 25-32), Mexico.

Greenhalgh, C., & Benford, S. (1997). Boundaries, awareness and interaction in collaborative virtual environments. *Proceedings of the 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises* (p.193), MIT, Cambridge, Massachusetts, June 18-20.

Herz, J. (1997). *Joystick nation: How computer games ate our quarters, won our hearts & rewired our minds.* Little, Brown & Company.

Hughes, B. (1995). Educational MUDs: Issues and challenges. Invited keynote presentation, *MUDshop II,* San Diego, California, September. Retrieved March 20, 2001 from: *www.pc.maricopa.edu/community/pueblo/writings/MudShopBillie.html*

Hughes, B., Walters, J., & Kort, B. (1994). Virtual space learning: Creating text-based learning environments. *Proceedings of the 1994 ACM Symposium on Applied Computing* (pp. 578-582), Phoenix, Arizona.

Hummel, J., & Lechner, U. (2002). Social profiles of virtual communities. *HICSS 2002,* 172.

Johnson, L. (1998).Pedagogical agents in virtual world training. In C. Landauer & K. Bellman (Eds.), *Proceedings of the Virtual Worlds and Simulation Conference* (VWSIM'98).

Kiesler, S. (1986). The hidden messages in computer networks. *Harvard Business Review,* (January-February).

Landauer, C., & Bellman, K. (1996). Integration systems and interaction spaces. *Proceedings of the First International Workshop on Frontiers of Combining Systems* (pp. 161-178), Munich, German, March 26-29.

Landauer, C., & Bellman, K. (1998a). MUDs, integration spaces, and learning environments. *Proceedings of the 31st Hawaii Conference on System Sciences,* Kona, Hawaii, January 6-9.

Landauer, C., & Bellman, K. (1998b). Integration and modeling in MUVEs. *Proceedings of VWsim'98: Virtual Worlds and Simulation Conference, 1998 SCS Western MultiConference* (pp. 187-192), San Diego, California, January 12-14.

Landauer, C., & Bellman, K. (Eds.). (1998c). *Proceedings of VWsim'98: Virtual Worlds and Simulation Conference, 1998 SCS Western MultiConference*, San Diego, California, January 12-14.

Landauer, C., & Polichar, V. (1998a). More than shared artifacts: Collaboration via shared presence in MUDs. *Proceedings of WETICE'98: Workshop on Web-Based Infrastructures for Collaborative Enterprises* (pp. 182-189), Stanford University, Palo Alto, California, June 17-19.

Lau, L.M.S., Adams, C.A., Dew, P.M., & Leigh, C.M. (2003). Use of scenario evaluation in preparation for deployment of a collaborative system for knowledge transfer—the case of KiMERA. *Proceedings of WETICE'03, The 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises,* Linz, Austria, June 9-11.

Lawley, E. (1994). *The sociology of culture in computer-mediated communication: An initial exploration.* Submitted in partial fulfillment of the requirements for LS695 Seminar.

Lipner, M. (1999). *The Web was pretty neat until they started messing it up with pictures.* Invited address, Symposium on Virtual Communities, American Association for the Advancement of Science Annual Meeting, Anaheim, California, January 21-26.

Macedonia, M. (2002). Games soldiers play. *IEEE Spectrum,* (March), 32-37.

Maes, P. (1987). Concepts and experiments in computational reflection. *Proceedings of OOPSLA'87* (pp. 147-155).

Mamei, M., Zambonelli, F., & Leonardi, L. (2003). Developing adaptive and context-aware applications in dynamic networks. *Proceedings of the 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (p. 401), Linz, Austria, June 9-11.

Milgram, P., Drascic, D., Grodski, J., Restogi, A., Zhai, S., & Zhou, C. (1995). Merging real and virtual worlds. *Proceedings of IMAGINA'95*, Monte Carlo, February 1-3.

O'Day, V., Bobrow, D., Bobrow, K., Shirley, M., Hughes, B., & Walters, J. (1998). Moving practice: From classrooms to MOO rooms. *Computer Supported Cooperative Work, Special Issue on Interaction and Collaboration in MUDs, 7*(1-2), 9-45.

Papert, S. (1980). *Mindstorms.* New York: Basic Books.

Perlin, K. (1995). Real-time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics, 1*(1).

Perlin, K., & Goldberg, A. (1999). Improvisational animation. In C. Landauer & K. Bellman (Eds.), *Proceedings of the Virtual Worlds and Simulation Conference* (VWSIM'99).

Polichar, V. (1996). An office MUD for fun and profit? Or maybe just better communication. *Login Magazine.*

Polichar, V. (1997). *On the value of MUDs as instructional and research tools.* Open letter provided to Northern Arizona University.

Psotka, J., Massey, L.D., & Mutter, S.A. (Eds.). (1998). *Intelligent tutoring systems—lessons learned.* Erlbaum, Hillsdale.

Raybourn, E., Kings, N., & Davies, J. (2003). Adding cultural signposts in adaptive community-based virtual environments. *Interacting with computers, 15*(1), 91-107.

Reid, E. (1994). *Cultural formations in text-based virtual realities.* Masters Thesis, Department of History, University of Melbourne, Australia.

Rheingold, H. (1992). Retrieved from: *gopher://gopher.well.sf.ca.us/00/Community/virtual_communities92*

Riner, R., & Clodius, J. (1995). Simulating future histories. *Anthropology and Education Quarterly, 26*(1), 95-104. Retrieved March 20, 2001: *www.dragonmud.org/people/jen/solsys.html*

Rowley, M. (1997). Distributing MOO-based shared worlds. *Proceedings of the Sixth Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises* (WETICE'97) (p. 155), MIT, Cambridge, Massachusetts, June 18-20.

Shilling, R., & Zyda, M. (2002). introducing emotion into military simulation and video game design. *America's Army: Operations and VIRTE.* GAME-ON 2002 23EE.

Silverman, B., Holmes, J., Kimmel, S., Branas, C., Ivins, D., & Weaver. R. (2002). The use of virtual worlds and animated personas to improve healthcare knowledge and self-care behavior: The case of the heart-sense game. *Intelligent agents and their applications* (pp. 249-294). Heidelberg, Germany: Physica-Verlag.

Soloway, E. (1993). Technology in education (introduction). *Communications of the ACM, 36*(5), 28-30.

Soto, M., & Allongue, S. (1997). A semantic approach of virtual worlds interoperability. *Proceedings of the 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises* (WETICE'97), (p. 173), MIT, Cambridge, Massachusetts, June 18-20.

Suthers, D. (1998). *CAETI architecture report.* Retrieved from: *lilt.ics.hawaii.edu/lilt/papers/1998/FINALREP.DOC*

Testani, S., Wagner, E., & Wehden, K. (1999). CIMBLE: The CADETT interactive multi-user business learning environment. *Proceedings of VWsim 1999, The 1999 Virtual Worlds and Simulation Conference.*

Towell, J. (2004). *MUDs in scientific conferencing.* Retrieved March 9, 2004 from: *www.hayseed.net/MOO/*

Trappl, R., Petta, P., & Payr, S. (eds.). (2003). Emotions in humans and artifacts. Boston, MA: MIT Press.

Turkle, S. (1995). *Life on the screen.* Simon and Schuster. *underground.musenet.org:8080/*

Viau, E.A. (1996). Melding technology and pedagogy. *Proceedings of AACE 1996*.

Viau, E.A. (1998a). World building: A course of the future. *Ed at a Distance Magazine*, (January).

Viau, E.A. (1998b). Color me a writer: Teaching students to think critically. *Learning and Leading with Technology, 25*(5), 2-5.

Viau, E.A. (1998c). Shades of meaning. *The Journal of Adolescent and Adult Literacy*.

Viau, E.A. (1998d). Building models, building worlds. *Proceedings of the National Educational Computing Conference,* San Diego, California, June.

Vince, J., & Earnshaw, R. (1998). *Virtual worlds on the Internet.* Wiley-IEEE Computer Society Press.

Wallace, J., & Sollenberger, J. (2001). Improving the state of military modeling and simulation: The joint synthetic battlespace. *Modeling and Simulation Magazine, 1*(1).

Welch, D., & Purtilo, J. (1997). Domain-driven reconfiguration in collaborative virtual environments. *Proceedings of the 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises* (WETICE'97) (p. 167), MIT, Cambridge, Massachusetts, June 18-20.

www.ibiblio.org/dbarberi/papers/mud/ (retrieved February 8, 2004).

Zyda, M. (2002). *Handling heterogeneity in networked virtual environments*. VR 2002: 7-14 21EE.

Zyda, M., Hiles, J., Mayberry, A., Wardynski, C., Capps, M., Osborn, B., Shilling, R., Robaszewski, M., & Davis, M. (2003) Entertainment R&D for defense. *IEEE Computer Graphics and Applications, 23*(1), 28-36.

Zyda, M., Mayberry, A., Wardynski, C., Shilling, R., & Davis, M. (2002a). *The MOVES Institute's America's army operations game*. SI3D 2003: 219-220 25EE.

Zyda, M., McGregor, D., Brutzma, D., & Kapolka, A. (2002c). *Tutorial 5: A hands-on introduction to networked virtual environments*. VR 2002: 304 22EE.

Chapter II

The Future Virtual Reality Melting Pot

Chadwick A. Wingrave
Virginia Tech, USA

Abstract

Virtual reality replaces or modifies human sensory input as do other technologies but with different methods or goals. Currently and even more so in the future, these technologies will work off the successes of each, creating a virtual reality melting pot. In this chapter, we look at some of these technologies and their current effect on virtual reality. From there, we identify human technological drives and use this to highlight future technologies that will meld into the melting pot. Lastly, we look at how some of these changes will impact human society and human everyday life.

Introduction

Many fields today are working towards a virtual, information-accessible world. Distance loses meaning, communication and information flows freely, and the sensory inputs are the manipulated mediums that make it so. Virtual reality (VR) is at the forefront of this, having the goal of replacing or modifying a range of sensory input of the user. The result of this goal is the study of how users behave and react when their environment changes in non-realistic ways. Other fields seek to modify the user's world through different means such as augmenting the user's senses or blending technology into the background of a human's life. However, the technology of these younger fields has not advanced as far as VR to make them practical for many studies involving users due to equipment and toolkit fragility. VR however has explored beyond merely equipment advances to create methodology and experiment with real users and their reaction to real VR experiences. Other fields in the future will then be able to work off VR research and literature, blending their advances with the inroads made by VR into a larger body of knowledge, a VR melting pot of technology.

An apology[i] is in order before we continue with the rest of this chapter. It becomes difficult to say where the bounds are of certain research fields. For instance, many fields seek the same goal of bettering the lives of humans through technology; they just take on different methods or emphasis. In the exploration of enabling VR technologies, it is likely that the statements of this chapter will annoy researchers from other fields as they claim their technology will be the dominant research that led to the change in our lives. The reality is that the future is built of many parts, interacting in a complex, seemingly chaotic manner. Judging importance and contribution will be left to the historians. Creating the importance and contribution is the charge for those with the creativity to see the connections and blur research lines, mixing the best of the various fields into something of utility for the average human.

But first, a vision of the future of VR through a scenario of a human's life:

As Brad exits his last required cycle of REM sleep, his blinds silently open to let in the rays of sunlight from the morning sun. He had no need to set an alarm as his room monitored his sleeping patterns and his virtual agent managed any interruptions (if an emergency arose, the agent could wake him). Still lying in bed, Brad asks his room what he missed through the night. The ceiling switches from a star-filled night sky to show the night's activity. Gesturing with his hands he moves through his messages…nothing outstanding. His news avatar clipped a few news stories, which he will look at once he gets to the office; Dave, Brad's friend, is having a dinner party in a few weeks and it was added to Brad's schedule; Brad's brother left a joke he had just heard about a man and an elephant; and, a few bits and pieces of work information from the global offices on their accomplishments through the night. Brad heads toward the shower to finish going over his messages as music turns on in the background. He always likes Mozart in *the morning, though every once in a while his room plays Beethoven, which is a surprising but welcome change. After his shower, Brad starts to head out the door when his virtual agent at the French office breaks in, appearing in his vision, and asks for help with a problem they have been stuck on. Normally, this activity would wait until work, but his French coworkers wanted to go to a long lunch so Brad's agent gets his attention before he enters the morning commute. Brad tells his agent to take him there and, though he is still standing in his living room, to his senses he appears in the European office surrounded by a few familiar French coworkers. They review diagrams of the building they have been developing and Brad points out a few subtleties of the construction project they had overlooked. He pulls a pencil from his pocket onto which the room overlays the blueprints of the hallway they are discussing. He then uses the pencil to point-out the location of doorways and how they connect to the main hall. Afterwards, Brad even tells them his brother's joke about the elephant. They all laugh and Brad says good-bye, thankful that the computer translated his joke into French properly, as humor has difficulties in auto-translation. With his meeting over, Brad walks out his front door and starts to review traffic patterns for his drive to work.*

One might not recognize this as a scenario of future virtual reality, but more of a scene from some science fiction novel. Today's clunky VR systems with expensive bulky trackers and large, low-resolution and small field-of-view displays are hard to imagine as portable, ubiquitous, high-resolution systems integrated into one's individualistic lifestyle. But then again, so were laptop-sized computers when a single mainframe filled a room. To draw parallels, just as computing was once only the domain for calculating artillery trajectories and business payrolls, so will VR grow out of its currently limited domain of expensive or hazardous-environment applications. The question is then, how will it grow?

In this chapter, we shall use two methods to support the VR melting pot and discuss how the changes will have social implications. The first method of understanding how VR will change is to look at existing and related technologies and extrapolate into the future. The conclusions that can be drawn are easily supported with existing facts. We will look at these technologies in the next section. The second method we will use to understand VR change is to understand what drives people and identify technologies that support those drives. This will be covered further on. Finally, we will look at some of the impacts on society that the melding of VR with other fields will have, and how people will adapt and blend technology into their everyday lives.

Currently Melding Technologies

VR will meld with other fields in the future including the currently mixing fields of Augmented Reality (AR), Ubiquitous Computing (UbiComp), Machine Vision (MV), Wearable Computing (WC), and Human Computer Interaction (HCI). AR in many cases uses wearable computers to do its computation and Machine Vision to track its users. UbiComp too uses Machine Vision to track users, along with many of the same types of projection technology as used with VR and AR. HCI methodology incorporates all of these fields into the everyday life of a user. Technological advances in each field allow for new possibilities, blurring the lines between them.

Augmented Reality

Augmented Reality focuses on the "enriching, rather than replacing" of reality. AR "annotates reality to provide valuable information, such as descriptions of important features or instructions for performing physical tasks" (Feiner, MacIntyre, & Sellgmann, 1993). Its hallmark technologies are see-through displays sometimes attached to wearable computers and tracking systems ranging in scale from worldwide global positioning systems (GPS) to typical VR room-scaled tracking systems.

In Augmented Reality, researchers study how to add information to the user's world in non-distracting, informative, and, not to be understated, safe ways. This extra information can be for tasks such as displaying shopping lists, viewing control panels, displaying safety information, and other work tasks.

Effect on VR

AR is perhaps the closest akin technology to VR, as demonstrated by the taxonomy of AR and VR experiences on the Virtuality Continuum (Milgram, Takamura, Utsumi, & Kishino, 1994), with advances in one field applicable to the other. Accurate outdoor AR tracking systems can be used for VR experiences, and interaction technologies are applicable to both as highlighted in Wingrave et al. (2003). Even some display technologies can be shared between the fields.[ii]

VR and AR will coexist as the importance of augmenting our everyday lives with graphical and textual information, mixed with the ability to switch to a completely virtual scene, becomes understood. An example of such use can be seen in the scenario with Brad appearing in France and working with his colleges while actually standing in his own house. In this scenario, displays and tracking systems are used that are familiar to both VR and AR.

Ubiquitous Computing

The guiding philosophy of Ubiquitous Computing was set forth years ago by Weiser (1991) as technologies that "weave themselves into the fabric of everyday life until they are indistinguishable from it." To this end, Weiser focused on displays scaled from palm-sized to wall-sized and the networking of them together. Others since have looked at a plethora of embedded technologies such as projected displays, networks such as wireless and Bluetooth, Machine Vision for tracking and face recognition, RFID tags, and more. UbiComp has a very large human emphasis in its research, with the technologies being created to solve or in response to very real human issues. Additionally, UbiComp has also been responsible for a very deep introspective look at how its technologies fit, ethically and legally, into the everyday lifestyle of its intended users. This is something that VR has not focused on, in large part due to VR being situated in the laboratory and rarely placed in the world.

Weiser worked hard to separate UbiComp from virtual reality, calling VR the "most diametrically opposed to our vision [of UbiComp]" (Weiser, 1991). He stated that "virtual reality focuses an enormous apparatus on simulating the world rather than on invisibly enhancing the world that already exists." His emphasis on the difference between the two is startling, considering the similarities between them. His position might have only been to separate UbiComp as a field unto itself as opposed to a subclass or type of VR. For instance, many of the same interaction techniques and display principles of VR can be applied to UbiComp. For example, the different methods of selection in VR (Wingrave, Bowman, & Ramakrishnan, 2002) and UbiComp (Myers et al., 2002) are quite similar and validated via similar research methodology. In fact, it is not hard to place UbiComp on Milgram's Virtuality Continuum and even use VR to prototype UbiComp applications without having to deal with each individual UbiComp technology. The only real difference is the placement of the hardware 'in the world' for UbiComp as opposed to 'on the user' for VR and some AR. The promise of both fields, the ability to work with information and people more efficiently, is the same.

Effect on VR

UbiComp's emphasis of embedding technology everywhere, invisibly into the background, will create many opportunities for VR. VR can run across the networks UbiComp envisions, the displays and speakers it seeks to embed, and more importantly, the philosophy of the effect of technology in daily life. UbiComp should also be congratulated for its early focus on privacy (Langheinrich, 2001) and social issues (Jiang, Hong, & Landay, 2002). This focus will pave the way for easier adoption of technologies that seem quite invasive and dangerous to the normal operation of people's private lives. Additionally, it gives guidance to law and policy makers hopefully in advance of their needs to respond so that the response will be well thought-out by those in the know and not result from panicked decisions by those ill-informed.

In this scenario, Brad interacted seamlessly with computers that opened his blinds, played his music, displayed images on his walls, monitored his sleep, and remembered his preferences—all UbiComp technologies.

Machine Vision

Machine Vision has been used to track people and items. It has been used in conjunction with AR and UbiComp to provide tracking information at coarse-grained (user is in the kitchen) (Kidd et al., 1999) and fine-grained (sign-language) (Starner, 1995) levels of detail. The recreation and improvement on the human visual system can be considered a distant goal of such technology, but it has had many successes in smaller tasks such as face recognition, eye tracking and position, and orientation trackers via fiducials (Hoff, Nguyen & Lyon, 1996; Neumann & Cho, 1996). Machine Vision has also been used in VR for tracking on a surface such as the Perceptive Workbench (Leibe et al., 2000). Vision tracking holds great promise because of its fast update rates, minimal environmental distortion, and lack of need for attaching receivers and wires to the user.

Effect on VR

The ease of using cameras as opposed to VR's traditionally low-range and encumbering trackers makes machine vision an enabling technology as it becomes more accurate and easier to apply.

In this scenario, when Brad rested in bed after waking and existed virtually in France, the room he physically existed in was able to watch his gestures so he could interact without being harnessed by wires and trackers.

Wearable Computing

Wearable Computing researches the impact of computation through light, compact, and low power computers into our daily lives. The field is split between finding uses for and increasing the computation. In addition to raw computational power, other factors such as power consumption and heat are important to the field. Much work has also gone into harvesting power and dissipating heat through the human host. Possible sources of power are heat, breathing, blood pressure, and limb motions (through multiple methods such as keyboards, flywheels, and piezoelectric materials) (Starner, 1996).

Effect on VR

Wearable computers can be seen as a huge enabling technology of VR. Its focus on smaller and more portable computing create the ability to take computers everywhere to augment our lives and create the virtual worlds on the fly. Additionally, the instrumenting of our selves will further increase our ability to replace haptic, visual, and auditory senses when the VR melting pot requires it.

In this scenario, though not specifically mentioned, Brad was wearing some type of computing device to drive his personal displays and review traffic patterns. In the long run the wearable computer is an enabling technology for VR and, inversely, VR is an application domain for wearable computers. Each field gives emphasis to the other.

Human Computer Interaction

Human Computer Interaction looks at smoothing the boundaries where computers and humans meet and interact. Having its birth in psychology, it has grown quite diverse through its successes and failures, eventually coming into its own as a separate and complete field of study with the focus on the user and user tasks. Early HCI focused on the user interacting with a single machine. This branched with time to people using machines to do work and workflow, with more recent studies focusing on interacting, working groups.

VR has to some extent adopted the methods of general HCI such as taxonomies of interaction tasks (Bowman & Hodges, 1999; Gabbard & Hix, 1997) and constraints (Bowman & Hodges, 1995), but this falls short of predictive models and detailed instructions on how to build interfaces. Good design still comes from experience, evaluation, and iteration on usually poor initial designs. There is even debate in the general field of HCI over whether or not generalized models can be created to inform design (Landauer, 1991).

There are major differences between VR melting pot technologies and the typical tasks approached by HCI. One underlying difference between typically desktop HCI and the VR melting pot HCI is the large volume of I/O in a VR melting pot between the user and the environment. For instance, whereas desktop interface input typically comes from 2D pointing devices and keyboards, VR can track multiple 6-D positions and hand joint angles (each hand alone can even be considered 19-D inputs). Another difference is the amount of familiarity a user of VR brings with them from the real world into a virtual world versus a typical desktop environment such as the WIMP (Windows, Icons, Menus, Pointers) metaphor. Users quickly realize that though the virtual world looks similar to reality, it hardly interacts as reality, leading in many cases to disappointing experiences. This is in contrast to desktop experiences that have very limited realism but result in successful interaction experiences because relatively low bandwidth devices make interpreting user intentions easier. Because of these differences, many of the successful models of HCI for improving the user experience in its typical domains, such as Fitts' Law (Fitts, 1954) and the Law of Steering (Accot & Zhai, 1997) for predicting user time to complete a task, are too simplistic to be usefully applied. Thus, VR remains a difficult domain to interact with, despite the research underway.

Effect on VR

VR has gained much from the field of HCI. User studies, evaluation, and a focus on the tasks of the user have started to make working, as opposed to just watching, VR a reality (Hix et al., 1999). A recently defined field of Information Rich Virtual Environments (IRVEs) (Bowman, North, Chen, Polys, & Pyla, 2003) is looking at ways to increase the utility of VR by supplementing the environment with several types of abstract information. Despite the efforts over the years to make VR usable, there is much work that needs to be done.

In the scenario, Brad worked effortlessly with the computers around him. He worked on the tasks he needed to perform and did not spend time or cognitive effort negotiating with the computing surrounding him.

Is This the Whole Story?

Futuristic extrapolations based on the technologies mentioned is easy but limited in that it cannot predict sideswiping events. Who could have predicted the impact that placing pictures into HTML documents for the Internet would have on today's economy. Or, who could have predicted that a simple spreadsheet application, VisiCalc, could bring about a personal computer revolution. These types of events sideswipe current projections and lines of thinking because the type of thinking in this section fails to focus on what meets people's wants, their drives. The more correct question then becomes: What are human drives and how does VR fit into them?

A Discussion of Drives

Recently, the author listened to a group of students discussing what made the Apple iPod[iii] a success. They named its sleek design, large song storage capacity, and simple and elegant interface. They viewed it as the sum of the technologies it comprised. This is unsettling because this view only allows for incremental improvements as the technologies comprising the iPod improve, for example, larger hard drives or a smaller case. A more thorough understanding of the iPod comes about in the realization of the task the iPod fills in people's lives; people want music in their lives. The sleek design minimizes the device's impact on the routines of people. The large storage space enables people to have their full collection of music at all times and not have to make decisions between songs they wish to carry and leave on their computer. The interface was made simple so as to reduce the amount of time people had to spend dealing with the device to access its functionality. People want more while giving less; less money, fewer tradeoffs, less time, and less cognitive effort. In essence, people's drives are for cheaper, more robust, faster, and simpler technologies. These drives are the spending capital of humans on technology and the method we wish to use to identify the important VR-enabling technologies in the future.[iv] To explore this concept, Table 1 lists the technologies discussed according to the drives they address.

A technology is only as useful as the benefits received by the user minus the hassle to afford, use, and maintain the technology. Augmented Reality addresses all the drives of the VR melting pot applications through better toolkits, better

Table 1. The drives technologies seek to address

	Cheaper	More Robust	Faster	Simpler
Augmented Reality	**X**	**X**	**X**	**X**
UbiComp		**X**		**X**
Machine Vision		**X**		
Wearable Computing	**X**	**X**		
Human Computer Interaction	**X**	**X**	**X**	**X**

design methodology, tracking accuracy, and so forth. UbiComp addresses the drives by a more robust integration of devices in the environment and a better understanding of the user. Machine Vision too assists in understanding the user. Wearable Computers reduce the costs of the computers in the environment. Applications are made cheaper by HCI design methods, more robust by a better understanding of the user, faster by reducing the amount of work to achieve user goals, and simpler by reducing the complexity and confusion of interfaces.

Validation of the VR melting pot can be seen in the successful existing applications of VR. Military trainers, psychological therapy, prototyping applications, and architectural and visualization walkthroughs (Brooks, 1999) all use VR because it is either cheaper, more robust (as per safety, transfer of knowledge, etc.), faster (to construct, design, or visualize), or simpler (to learn, cognitively use, etc.). As VR becomes more mature, as per the drives mentioned, the problems that VR will be applied to will no longer be just a few, large applications, but many smaller applications too. The results will be the blurring of the lines of reality as the environment slowly becomes augmented, hardware fading into the scene becoming ubiquitous, and VR no longer seen only as Fishtank VR, Head-Mounted Displays, or projected displays, but existing in the world as a melting pot of technologies. The following examples highlight everyday problems that can be addressed:

Problem: Did I leave my pen on my desk?

Solution: Virtually travel to the desk using embedded cameras to view the desk and the items on it.

Problem: Do I have access to the door in the building or is the door currently unlocked?

Solution: Reference the door in the environment and view situated information about the door. Just as Web pages have an Internet address, so too can abstract information be linked to real-world locations and settings.

Problem: Do I have a ripe tomato in the refrigerator?

Solution: While standing in the store, virtually take a tour of the home refrigerator. The user is not interested in just an inventory, but the properties of objects in the inventory. What does the tomato look like?

Problem: Bob needs to give the OK to proceed, where is he?

Solution: If Bob wants to be found at the given time and by the given person (social context of human-human interaction is important), then Bob will

Table 2. Advancements that will meld with VR to fulfill the identified human drives

	Cheaper	More Robust	Faster	Simpler
Agents, Interruption, and the Periphery		**X**		**X**
Computer Supported Collaborative Work		**X**	**X**	
Magic			**X**	**X**
Publication and Propagation of Information	**X**		**X**	**X**
Tangible Interfaces	**X**	**X**		**X**

> let them know where he is or where to meet him if he does not feel they need to know his current location.

So if these types of problems are exemplary of the drives people have and the problems to be addressed, then the question is, what type of advancements can address the drives we have identified? A partial listing of these advancements is in Table 2. It is easy to imagine the impacts of several other advances such as material science, nano technology, and wireless technologies too, but those are beyond the scope of this chapter.

Agents, Interruption, and the Periphery

The information age ushered in a tidal wave of easily obtainable information without any cognitive change in the ability of the human to manage. To deal with the problem, research on agents acting on behalf of or with the human have increased the volume of data with which the human can easily work. Additionally, work on when to bring information to the attention of the human and how to display information in a non-distracting manner has also increased the volume of data that can be worked with (McCrickard, Czerwinski & Bartram, 2003).

Once information is in a useful form, it needs to be presented in an intuitive manner. There has been research on how people respond to new media (Reeves & Nass, 1996) and how to use animated agents to converse with users for certain tasks (Cassell, Sullican, Prevost, & Churchill, 2000). Cassell has worked on conversational agents to simulate human-human interaction by emulating verbal and non-verbal communication. Affective computing (Picard, 1997), which deals with human emotions and computing, is also relevant. The hope is that by simulating human-human interaction with computer agents, information can be communicated in a more natural, easily understood manner.

Peripheral and notification displays and interfaces also seek to convey information in a natural, non-distracting manner. Ideally as information is updated and presented, the user should only be aware of it as needed without being constantly reminded of its presence. Change blindness (Intille, 2002) and inattentional blindness (Simons, 2000) give the ability to update displays without the user perceiving a change.

Drives of the Technology

Agents, interruption, and periphery technologies address the human drives of technology needing to be more robust and simple. These technologies have the ability to work with the human as the human goes about his or her everyday tasks. The technology will blend into the background and the agents will be more of a filter through which information is passed from the outside world to the user. The recreation of human-human protocols could potentially make the interfaces simpler to learn and use.

Effect on the VR Melting Pot

The melting pot technologies have the ability to present information to the user, but agent, interruption, and periphery technologies will guide how that information is presented. In the example scenario, Brad interacted with a virtual agent in the morning about activities that went on during the night. Additionally, Brad's agent was able to negotiate without any input from Brad a time to interact with his French coworkers that was convenient for them, so they could go to lunch, and for him, so as not to interrupt his time in the morning with his wife but before he entered his commute.

Computer Supported Collaborative Work

Computer Supported Collaborative Work focuses on computer systems that support collaborative human work, careful to support the subtle human-human interactions. The applications of CSCW can generally be described along two axes, the first being communicating synchronously (at the same time) or asynchronously (at different times), and the second being interacting collocated (existing in the same location) or remotely. Videoconferencing is a simple example of a synchronous and remote CSCW application, since groups are in different locations and the communication takes place at the same time for both remote groups. There are major technical and non-technical challenges building more complex systems (Grudin, 1990), with one major reason being the difficulties in supporting the lack of formalism or explicitness of human interaction (Shipman & Marshall, 1999). Despite these challenges, CSCW has had success in such applications as collaborative calendars (Palen, 1999), collaborative meeting rooms (Johanson, Fox, & Winograd, 2002; Stefik et al., 1987), MUDs and online communities (Kollock, 1996; Carroll et al., 2000), and even simple applications such as e-mail.

Drives of the Technology

CSCW focuses on making interactions between humans more robust and the ability to collaborate on work faster.

Effect on the VR Melting Pot

CSCW stands to give the VR melting pot a methodology of how to electronically support people doing work and a body of knowledge to guide collaboration in VR. A subfield of VR already exists for dealing with collaborating VR users called Collaborative Virtual Environments (CVEs), which has its own biennial conference started in 1996.

In the scenario, Brad worked collaboratively with a remote office in France. Language and distance were removed as a barrier to the work being done. Additionally, his friend Dave was able to access Brad’s personal calendar to add appointments.

Magic, the Breakdown of Reality

The ability for VR to break reality’s limitations has long been touted as an advantage, but it leads to the struggle between intuitiveness and efficiency. Interacting magically is not as intuitive as naturalistic interaction because of the lack of familiar affordances. These affordances tell the user how to behave and react, just like a doorknob affords turning and a button affords pushing. Despite its lack of affordance, magic in VR can be quite efficient, allowing users to interact without regard to some fundamental assumptions of reality (Pierce & Pausch, submitted) such as reaching beyond arm lengths, existing virtually in multiple locations, lifting and manipulating objects without regard to size and mass, just to name a few. Additionally, magic allows retraining the brain to respond to new stimuli by piping information down existing sensory channels such as visually overlaying building support structures in walls (Feiner, Webster, Krueger, MacIntyre, & Keller, 1995), billiard shot angles for the game of pool (Jebara, Eyster, Weaver, Starner, & Pentland, 1997), and repair instructions for a laser printer (Feiner et al., 1993); other potential applications include visualizing radiation levels, heat, ultrasonic sounds, and other information. Visual overlays are not the only methods, as sound overlays have been used to convey medical information by moving a CAVE wand through a volume denoting density (Brady et al., 1995). Danger or heat can be conveyed using psychophysically similar sounds such as sizzling. One could even imagine other dimensions of sensory information, for example: "Hmm, the data seems to extend quite far into the pine tree scented dimension at this point here."

Drives of the Technology

Though not a specific technology, magic interaction has the ability to make interaction faster and, with experience, simpler as users learn new magical metaphors of interaction.

Effect on the VR Melting Pot

By retraining the mind to interpret sensory information differently yet in consistent ways, people's reality becomes accepting of new information types as they can perform actions not previously possible and deal with information not previously available.

In the opening scenario, Brad's nighttime environment was a star-filled sky, and when he needed to travel to France to work with his officemates, his room was transformed to their location and he existed virtual there.

Easy Publication and Propagation of Information

The ability to easily create, destroy, situate, and manage information for self and others, locally and globally, will be a driving motivation behind the value in the melting pot technologies. Again not a specific technology in itself, this is a requirement of the protocols that need to allow the generation of information quickly and effortlessly. In effect, information can be placed where people with appropriate access will see the published information and hopefully respond with the same amount of ease. This allows communities to develop simple messages between people to build up knowledge in the world in which they interact. This places content creation into the hands of common people, scientists, engineers, and artists to assist their lives. These same people will not have this opportunity if the manner of creating this content requires understanding of complicated markup languages or experience in information tricks.[v] Ultimately, if the effort required to post a note or object digitally is more than the effort to write a note on paper and place it in the environment, the value of the content will suffer. Examples of this concept in action are:

Context:	Housemate to other housemates
Note:	"I left the pizza in the fridge for you." or "My pizza. I spit on it. Don't eat it."
Effect:	The note will always be seen when the user opens the refrigerator or enters the kitchen area. Additionally, if dinner comes up in conversation or the housemate enters a restaurant forgetting they have leftovers, this note becomes relevant and can notify the user. The note then has a presence outside just the home.

Context:	Restaurants to pedestrians
Note:	Lunch menu and specials on the outside of a restaurant.
Effect:	Easily done with signs now, but electronic publication enables people to browse restaurants virtually through such scenarios as, "I have to run an errand near midtown, let's see what restaurants are there and their specials." This same scenario allows people to also post comments about the restaurant's quality and situate the comments next to the restaurant. The legalities and consequences of such postings would have to be determined.

Drives of the Technology

Better methods of publication and propagation of information will allow for cheaper information creation, faster access to information, and simpler access to the correct information at the correct time.

Effect on the VR Melting Pot

The VR melting pot technologies create ways to propagate information, but it needs important content to propagate. In the scenario, Brad's colleagues were able to post information to Brad about notifying them before he left for work, Brad's brother was able to publish a joke to Brad, and Dave updated information on Brad's calendar. The incorporation of those information streams into the life of Brad meant that people could rely on the information getting through and would assume that the information was presented at an appropriate time.

Tangible Interfaces

Tangible Interfaces deal with interaction built into real-world objects that people can intuitively manipulate in the foreground (graspable media) or observe an unseen phenomenon in the background (ambient media). The ability for people to work with real objects that they are familiar with, yet are now augmented, enables faster understanding of applications and a more natural interaction because of the haptic feedback and affordances provided.

A good review of tangible interfaces can be found in Ishi and Ullmer (1997). Such applications include the ambientROOM, where grasping a model of a car causes Web hits to a car Web site to be displayed audibly as raindrops or visually on the ceiling with ripples in water caused by light projected through a water tank. Another application, mediaBlocks, uses blocks of electronically tagged wood to represent media that play when the wood blocks are inserted into media players, creating an intuitive interface for media control. Live Wire was a system for the ambient display of network traffic by having a cord hanging from the ceiling jostle about as packets traveled through a network (Weiser & Brown, 1995). Toolkits such as Phidgets support the creation of Tangible Interfaces through a collection of physical widgets such as sensors, motors, and RFID tags and readers (Greenberg & Fitchett, 2001).

Drives of the Technology

Tangible interfaces make hardware cheaper, as common items or simply instrumented items can be used for new or grander purposes. Additionally, more robust interfaces can be created due to the affordances of common, familiar objects.

Effect on the VR Melting Pot

By instrumenting common devices, the VR melting pot can report and interact with information and functionality in the tangible world. The tangible devices gain better ways to distribute their information to the user through VR, and in turn the VR melting pot gains more information about the environment the user is in.

In the scenario, Brad holds a simple pencil in his hand that is representative of a hallway they are discussing. Using the pencil as an augmented prop, he is able to show extra information such as doorways.

Changes to Social Practice

The technological changes we have mentioned so far will lead to changes in the social practice of people, just as all innovations do. Telephones caused people to stay in better touch with distant friends. Radio, television, and more recently the Internet have brought about the propagation of information and the unification of disparate cultures. Portable music has brought about an interest in sharing music collections, and so forth. It is most likely that advances in all the aforementioned technologies will have an impact also.

Below are four instances of impacts to social practice and should be looked at as examples of what will occur due to the VR melting pot. The first impact discussed is the creation of socially awkward situations through new technologies. The second impact is on entertainment as new tools increase the ways for entertainers to entertain and situations in which to entertain us. The impact of personal preference and individuality and how it will shape our spaces with different imagery follows. Lastly, we should remember that some parts of life cannot be impacted as people are human, and despite technology's ability to manipulate the human experience, not all things are able to change.

Social Awkwardness and Protocols

As researchers, we should constantly be aware of how our technologies fit into the human lifestyle—if for no other reason than just to avoid duplicating one of technology's greatest social flops: the inappropriate cell phone ring. Cell phones have caused oddities in social behavior from badly timed cell rings to inappropriate calling and answering times. We can only expect these oddities to increase with advances to the technology. For instance, shrinking cell phones and earpiece sizes have cause people to appear as if they are talking to themselves in public, with some people facing walls to avoid such confusion. Even desktop social conventions and problems exists. The lack of ability to express emotion in pure text transmissions such as e-mail and chat was partially alleviated through the use of emoticons (Rivera, Cooke, & Bauhs, 1996), defined as sideways punctuation that looks like various emotion-containing faces. In the early Internet days, many new users to the Internet broke the online text communication social convention by using all capital letters in newsgroups. Existing practice was for capital letters to be saved to express emphasis which caused many experienced users to feel like they were being yelled at.

Social protocols for the correct method of dealing with technological issues have to be established by society. The author once overheard a conversation between two women in a coffee shop discussing how they did not get along with a third because the third did not apologize properly for accidentally sending an e-mail virus. One could imagine a Seinfeld episode dealing with just such a situation with hilarity ensuing. As reality becomes augmented as discussed earlier in this chapter, we can expect awkward social settings to occur. People gesturing about wildly, manipulating virtual objects or responding to periphery information existing solely in their view, will be seen as scary or even insane until people become familiar with such behavior. The implications are that as we create new technology, it impacts more than just allowing us to do more things; it changes how we go about our lives and work and live with others. Over time, social conventions adapt to fit the technology, but we must try to help or predict this adaptation. For example, Brad's wife might not enjoy her husband working with the French office during their private time in the morning when he and she have a few moments to spend together before they both leave for work. This might change as Brad and his wife adapt over time to their being able to work anywhere at any time. This gives them the flexibility to spend more time at home with each other (so much time that interruptions from the French office might be welcome by Brad's wife!).

Entertainment

The new medium of VR will likely spur new types of entertainment, as artists, directors, and technologists utilize the new possibilities presented by VR and its related technologies. Already, the movie industry has incorporated motion capture technology to create realistically moving computer-generated animations and movies such as *Final Fantasy.*

New possibilities for entertainment, not just better methods of creating the old, are possible however. DisneyQuest has been at the forefront of this, creating games using VR technology, and making them robust and playable enough to keep even Disney audiences entertained. Games played in people's daily lives with standard devices like a cell phone are gaining in popularity such as those created by the It's Alive (www.itsalive.com) company. It creates games such as BotFighters, where players shoot it out with others close by using their cell phones to attack with lasers and rockets, with thousands of users in multiple countries. Still in the research domain, games such as "AR Quake" (Piekarski

& Thomas, 2002) superimpose the classic first-person shooter game “Quake” into the real world, tracking the user as he or she moves and shoots at monsters.

It would be shortsighted to suggest that entertainment stops at work. Currently, game playing at work is considered a major corporate inefficiency, as game playing takes away from work time. Many games even come with a bosskey[vi] to facilitate playing at work but not getting caught. One group has created a game that supports work as a game by putting part of a system administrator’s job, that of process control, into a popular first-person shooter “Doom” (Chao, 2001). One could imagine future scenarios of work where games, by virtually replacing and adding information to our world, make tedious or boring work into a game. Motivation from the business side would come from better methods of quantifying worker production or efficiency by keeping track of employee scores. By taking an employee motivation to be entertained and VR melting pot technologies, a new situation can emerge of entertained employees happily and efficiently working while managers are better able to predict costs and production through better information about their employees. “The employee of the month goes to Angela who successfully completed level 8 for an office new high score.”

Personal Preferences and Individuality

Not all environments need to be experienced the same, as not all people are the same. The phrase “Beauty is in the eye of the beholder” is functionally true and can be supported through changing how the user sees walls, hears sounds, or walks about during their day. The ability of the iPod to augment a person’s everyday life with music has been discussed, but not the ability of VR and AR to change the perception people have of buildings, light, and color. One could imagine a theme that replaces all buildings with 16th century counterparts and redraws people as of the same era complete with clothes, carts, and so forth. More functional than themed reality, however, is the creation of interfaces that react based upon the cognitive or motor factors of the immersed individual. For instance, users with higher spatial abilities would prefer certain interfaces, and people in better physical shape would not mind fatiguing interactions to the same extent as, say, the stereotypical user (Wingrave, Tintner, Walker, Bowman, & Hodges, 2004).

Recommender systems (Resnick & Varian, 1997) are currently being used to predict music, book, and other items based upon shopping habits of the community. In much the same way, it could be used to suggest certain preferences based upon personal characteristics. For instance, when entering a new town, suggestions for a Mexican restaurant that is highly rated could be brought to the user’s attention. This is not to be confused with target marketing, which is directing advertising towards the user as opposed to the former, which gathers benevolent information in the environment for the user. In the scenario, Brad's room suggested music for him to listen to in the shower, even music he normally did not listen to but liked.

Distance Matters

Despite all the technological advances, there will be some things that will remain the same. Olson and Olson (2000) discuss four different issues of distance that will not disappear with technology: (1) common ground, context, and trust; (2) time zones; (3) culture; and (4) interaction among factors and technology.

In common ground, they refer to the failure of the technology to place people in a similar situation so as to gain trust and promote teamwork. They site a situation in which a snow storm in Chicago made the Americans late for a videoconference with their British counterparts and which left the British wondering where the Americans were.

Time zones will never disappear and will lead to problems with people being unable to communicate due to one or the other being asleep. In our example, Brad was sleeping while his French counterparts were working and needed his help. Technology will, however, increase the ability to be aware of people being asleep and notified of possible times to remotely converse. Even new methods of managing sleep, in some cases through drugs, are being developed to affect the amount of sleep required by humans which could give people more control over their rest periods (Fleming-Michael, 2003).

Culture will always be a problem for people misinterpreting actions, and this will not change in the future, possibly becoming more of a problem as technology allows decidedly different cultures to come into closer contact. People will have to grow tolerant to deal with the closeness of those different than themselves in the future. Additionally, the technology itself might alleviate parts of the differences, such as style of dress or different speech turn-taking behavior, by virtually modifying people's attire or inserting and removing pauses in the audio communication of the participants.

These issues will become larger in the future as the melting pot technologies break down distance, physically and culturally. Technology can help solve some of the resulting problems, but as always, the technology is only a tool that social conventions will form around, and some social conventions cannot change. For the workday, there are many scenarios of how people might adapt to globalization. Effective but unlikely scenarios are the breakdown of the standard workday for people working around the world so as to increase time where all offices can be awake. Another would be the adoption of a world time to reduce the difficulties in scheduling.

VR and its related technologies will be a long time coming before they can fully simulate the human-human interaction of having a beer with a coworker or playing a round of golf, typical relaxed settings of the western work experience where people build trust and talk candidly. Someday however, it might be able to simulate this or perform some other task that performs the same functionality inside VR.

Conclusions

In this chapter, we have looked at two methods of understanding the technologies that are going to be relevant to the future of VR and the social implications of that future. The first method, that of looking at existing technologies, helps give an understanding of what is currently happening and what we might expect in the near future. The second method, that of understanding the drives of humans and then deciding which advances will impact those drives, should help to explain the long-term directions of the virtual reality melting pot; several technologies were named and discussed. Lastly, the possible impacts on humans due to the proposed changes were discussed.

References

Accot, J., & Zhai, S. (1997). Beyond Fitts' Law: Models for trajectory-based HCI tasks. *Proceedings of ACM CHI* (pp. 295-302).

Bowman, D., & Hodges, L. (1995). *User interface constraints for immersive virtual environment applications.* Graphics, Visualization and Usability Center Technical Report GIT-GVU-95-26.

Bowman, D., & Hodges, L. (1999). Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *The Journal of Visual Languages and Computing*, *10*(1), 37-53.

Bowman, D.A., North, C., Chen, J., Polys, N.F., & Pyla, P.S. (2003). Information-rich virtual environments: Theory, tools, and research agenda. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*.

Brady, R., Pixton, J., Baxter, G., Moran, P., Potter, C.S., Carragher, B., & Belmont, A. (1995). Crumbs: A virtual environment tracking tool for biological imaging. *Biomedical Visualization*, 18.

Brooks, F.P. (1999). What's real about virtual reality? *IEEE Computer Graphics and Applications, 19*(6), 16-27.

Carroll, J.M., Rosson, M.B., Neale, D.C., Isenhour, P.L., Dunlap, D.R., Ganoe, C.H., Van Metre, C.A., Seals, C., Fogarty, J., Schafer, W.A., Bussom, T., Bunn, K., Davie, P., Freeman, M., Goforth, A., Mauney, S.M., Rencsok, F.C., Anderson, C., Hertel, M., & Svrcek, B. (2000). The LiNC Project: Learning in networked communities. *Learning Technology, 2*(1).

Cassell, J., Sullivan, J., Prevost, S., & Churchill, E. (2000). *Embodied conversational agents*. Boston, MA: MIT Press.

Chao, D. (2001). Doom as an interface for process management. *Proceedings of CHI* (pp. 152-157).

Feiner, S., MacIntyre, B., & Sellgmann, D. (1993). Knowledge-based augmented reality. *Communications of the ACM*, *36*(7), 53-62.

Feiner, S., Webster, A., Krueger, T., MacIntyre, B., & Keller, E. (1995). Architectural anatomy. *Presence*, *4*(3), 318-325.

Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, *47*, 381-391.

Fleming-Michael, K. (2003). The sleep factor. *Soldiers*, (October), 38-41.

Gabbard, J., & Hix, D. (1997). *A taxonomy of usability characteristics in virtual environments.* Deliverable to Office of Naval Research from Department of Computer Science, Virginia Tech.

Greenberg, S., & Fitchett, C. (2001). Phidgets: Easy development of physical interfaces through physical widgets. *Proceedings of the UIST 2001 14th Annual ACM Symposium on User Interface Software and Technology* (pp. 209-218), Orlando, Florida, November 11-14. New York: ACM.

Grudin, J. (1990). Groupware and cooperative work: Problems and prospects. In B. Laurel (Ed.), *The art of human-computer interface design* (pp. 171-185).

Hix, D., Swan II, J.E., Gabbard, J.L., McGee, M., Durbin, J., & King, T. (1999). User-centered design and evaluation of a real-time battlefield visualization virtual environment. *IEEE Virtual Reality*, 96-103.

Hoff, W., Nguyen, K., & Lyon, T. (1996). Computer vision-based registration techniques for augmented reality. *Proceedings of IRCV* (SPIE 2904, pp. 538-548).

Intille, S.S. (2002). Change blind information display for ubiquitous environments. In G. Borriello & L.E. Holmquist (Eds.), *Proceedings of the Fourth International Conference Ubiquitous Computing* (LNCS 2498, pp. 91-106), September. Berlin: Springer-Verlag.

Ishi, H., & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. *Proceedings of the Conference on Human Factors in Computing Systems (CHI)* (pp. 234-241).

Jebara, T., Eyster, C., Weaver, J., Starner, T., & Pentland, A.. (1997). Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers. *Proceedings of the International Symposium on Wearable Computers* (pp. 138-145).

Jiang, X., Hong, J.I., & Landay, J.A. (2002). Approximate information flows: Socially based modeling of privacy in ubiquitous computing. *Proceedings of UbiComp* (LNCS 2498, pp. 176-193).

Johanson, B., Fox, A., & Winograd, T. (2002). The Interactive Workspaces Project experiences with ubiquitous computing rooms. *IEEE Pervasive Computing Magazine, 1*(2).

Kidd, C.D., Robert J.O., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E., Starner, T.E., & Newstetter, W. (1999). *Proceedings of the Second International Workshop on Cooperative Buildings-CoBuild'99* (Position paper), October.

Kollock, P. (1996). Design principles for online communities. *Harvard Conference on the Internet and Society*.

Landauer, T.K. (1991). Let's get real: A position paper on the role of cognitive psychology in the design of humanly useful and usable systems. In J.M. Carroll (Ed.), *Designing interaction psychology at the human-computer interface*. New York: Cambridge University Press.

Langheinrich, M. (2001). Privacy by design—principles of privacy-aware ubiquitous systems. *Proceedings of UbiComp*.

Leibe, B., Starner, T., Ribarsky, W., Wartell, Z., Krum, D., Weeks, J., Singletary, B., & Hodges, L. (2000). Towards spontaneous and natural interacting in semi-immersive virtual environments. *IEEE Virtual Reality* (pp. 13-20), New Brunswick, New Jersey, March.

McCrickard, D.S., Czerwinski, M., & Bartram, L. (2003). Introduction: Design and evaluation of notification user interfaces. *International Journal of Human-Computer Studies, 58,* 509-514.

Milgram, P., Takemura, H., Utsumi, A., & Kishino, F. (1994). Augmented reality: A class of displays on the reality virtuality continuum. *Telemanipulator and Telepresence Technologies*. SPIE.

Myers, B.A., Bhatnagar, R., Nickols, J., Peck, C.H., Kong, D., Miller, R., & Long, C. (2002). Interaction at a distance: Measuring the performance of laser pointers and other devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves* (pp. 33-40).

Neumann, U., & Cho, Y. (1996). A self-tracking augmented reality system. *Proceedings of ACM VRST*.

Olson, G.M., & Olson, J.S. (2000). Distance matters. *Human Computer Interaction, 15,* 2-3, 139-178.

Palen, L. (1999). Social, individual and technological issues for groupware calendar systems. *Proceedings of CHI 1999* (pp. 17-24).

Picard, R.W., & Picard, R. (1997). *Affective computing*. Boston, MA: MIT Press.

Piekarski, W., & Thomas, B. (2002). ARQuake: The outdoor augmented reality gaming system. *Communications of the ACM, 45*(1), 36-38.

Pierce, J., & Pausch, R. (submitted). Creating 3D interaction techniques by identifying and breaking assumptions. Submitted to *PRESENCE*.

Reeves, B., & Nass, C. (1996). *The media equation: How people treat computers, television and new media like real people and places*. City: Cambridge University Press.

Resnick, P., & Varian, H.R. (1997). CACM special issue on recommender systems. *Communications of the ACM, 40*(3), 56-58.

Rivera, K., Cooke, N., & Bauhs, J. (1996). The effect of emotional icons on remote communication. *Conference on Human Factors in Computing Systems* (pp. 99-100).

Shipman III, F.M., & Marshall, C.C. (1999). Formality considered harmful: Experiences, emerging themes and directions on the use of formal representations in interactive systems. *Computer Supported Cooperative Work, 8*(4), 333-352.

Simons, D.J. (2000). Attentional capture and inattentional blindness. *Trends in Cognitive Sciences, 4,* 147-155.

Starner, T. (1995). *Visual recognition of American Sign Language using hidden Markov models.* Thesis, Massachusetts Institute of Technology, Boston, Massachusetts, USA.

Starner, T. (1996). Human powered wearable computing. *IBM Systems Journal, 35*(4).

Stefik, M.J., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S., & Suchman, L. (1987). Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM, 30*(1), 32-47.

Weiser, M. (1991). The computer for the 21st century. *Scientific American, 265*(3), 95-104.

Weiser, M., & Brown, J.S. (1995). Designing calm technology. Retrieved from *www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm*

Wingrave, C., Bowman, D.A., Feiner, S., Schmalstieg, D., Mine, M., & Swan, E. (2003). Mixed reality interaction: The continuum from virtual to augmented reality. Conference Panel, *IEEE Virtual Reality*.

Wingrave, C., Bowman, D., & Ramakrishnan, N. (2002). Towards preferences in virtual environment interfaces. *Proceedings of the Eurographics Workshop on Virtual Environments* (pp. 63-72).

Wingrave, C., Tintner, R., Walker, B., Bowman, D., & Hodges, L. (2004). *Exploring individual differences in Raybased selection: Strategies and traits*. Human Computer Interaction Consortium.

Endnotes

[i] As per the meaning of defense or justification, not an expression of regret or asking for pardon.

[ii] Some AR displays operate by placing AR imagery into a video feed that is then presented in an HMD, which is very much like a VR HMD.

[iii] An iPod is a digital music player created by Apple Computer Inc.

[iv] Even this approach has its limitations, however, as new technologies can be discovered that create drives that were not previously envisioned.

[v] An example comes from the early days of the Web, when search engines were easily fooled by repeating keywords multiple times to increase the likelihood of that page being associated with a concept.

[vi] A bosskey quickly pauses a game and reverts to a work screen to make it look as if work is being done when the boss enters an office.

Section I

Whole Virtual Environments Development Methods

Chapter III

A Methodology of Design for Virtual Environments

Clive Fencott
University of Teesside, UK

Abstract

This chapter undertakes a methodological study of virtual environments (VEs), a specific subset of interactive systems. It takes as a central theme the tension between the engineering and aesthetic notions of VE design. First of all method is defined in terms of underlying model, language, process model, and heuristics. The underlying model is characterized as an integration of Interaction Machines and Semiotics with the intention to make the design tension work to the designer's benefit rather than trying to eliminate it. The language is then developed as a juxtaposition of UML and the integration of a range of semiotics-based theories. This leads to a discussion of a process model and the activities that comprise it. The intention throughout is not to build a particular VE design method, but to investigate the methodological concerns and constraints such a method should address.

Introduction and Problem Statement

Interactive systems (ISs) are becoming ubiquitous to the extent that there is the very real possibility of their disappearing altogether, at least in the sense of users' perceptions of them as entities worthy of conscious identification. This very ubiquity will largely be the result of effective design, which results in ISs becoming so embedded in our everyday lives that we use them without conscious thought. We can draw an analogy here with the electric motor, which pervades almost all everyday technologies and yet is hardly ever noticed. In the early twentieth century, it was possible to buy electric motors for the home along with a variety of attachments for food preparation, hair drying, vacuum cleaning, and so on. Today we buy specialized gadgets, many of which contain electric motors that go largely unnoticed by us. Even the mobile phone contains an electric motor that is weighted to spin off-centre in order to create the vibrations that can silently signify an incoming call.

Will this ever be the case with ISs? Will they ever be so effectively designed that they cease to attract conscious attention in their final ubiquity? Certainly, the theory of design for ISs is still in its infancy; hence the need for the present volume.

Before considering their design, we first need to make clear what we mean by ISs. Many systems are interactive but outside the remit of this book. Motor cars, power drills, electric kettles, and so on are all interactive systems that will not be the subject of this chapter. By ISs we surely mean interactive digital systems (IDSs) that make use of digital representations and operations on these in order to effectively perform their allotted tasks. IDSs will therefore identify everything from ATMs and remote controlled TV teletext systems to PC and game console applications to onboard computers in cars and fly-by-wire aircraft.

An interesting subset of IDSs are interactive digital environments (IDEs) by which we mean an IDS that creates a large-scale digital environment that takes time and effort to explore and otherwise interact with. Examples of IDEs are videogames and virtual environments (VEs) in general, computer-based learning applications, and large-scale sites on the World Wide Web. These are interesting because the scale and complexity of their content demands that their effective design transcend established user interface techniques. Indeed, for VEs the very term *design* is a problem because it has to be interpreted in two quite distinct ways. First of all there is the notion of designing something to create the desired perceptual and aesthetic responses: essential for computer games. Secondly, there is the engineering notion of design as the creation of plans and models from which to test and build the desired artefact and ensure its correct functioning. Both forms of design are of equal importance to the design of effective VEs. It is the tension between these two notions of design and the resolution of this ‘design tension’ that is the central problem addressed in this chapter.

The need to resolve or at least alleviate this tension leads to a consideration of methods for VE design. It is assumed by some that the design of effective VEs will necessitate a development methodology akin to those used (or not) by software engineers. This is not necessarily the case. A craft-based approach based on the application of good practice—perhaps acquired through some form of apprenticeship—might do equally well. The computer games industry seems to prosper on just such an approach. The approach taken in this chapter is that an appropriate form of development methodology for VEs is viable, but that that methodology needs to accommodate—and certainly not stifle—the creative flair that is at the heart of aesthetic design of such large and complex systems.

This chapter therefore concerns itself with the investigation of what form an appropriate design methodology for VEs would take and the obstacles to establishing such a methodology. It is thus primarily concerned with a methodology of design—in other words, the meta-study of VE design methods rather than the outline of a particular method, although this is an obvious objective.

This chapter first undertakes an overview of the meaning of the various terms involved in the discussion: method, methodology, model, and language, among others. It then goes on to discuss the particular form an ‘underlying model’ for a VE method would have to take. Following this the issue of the form a language for expressing VE design decisions might take with regard to the underlying model put forward in the third section is addressed. The chapter then goes on to establish a process model for VE design and the ‘practice of methodology’ it to a large extent determines. It finally attempts to address future trends in the field and is followed by a short conclusion to the issues raised.

Terminology

A methodology of design for VEs concerns itself with the study of methods for the design of VEs; in other words, the nature, definition, and application of such methods. This notion of methodology, while being quite correct, is at variance with a related but somewhat different notion that commonly views a methodology as a configurable method. In this chapter we use the approach of the former in order reach some conclusions with respect to achieving the latter.

If we are considering the study of methods for VE design, what do we mean by method in the first place? In software engineering the concepts of method and model are commonly understood, although the formality with which they are defined and applied varies considerably.

With respect to the question posed above, we will adopt the definition of Kronlof (1993) who defined a method as consisting of the following:

- An underlying model
- A language
- A process model
- Heuristics

Fencott et al. (1994) discuss these terms in the context of investigating the integration of structured and formal methods for software engineering. Methods integration will also be at the heart of the investigations of this chapter. Before using this characterization of method to address VE design, we will discuss the concept of model in some detail as it appears twice above in seemingly different contexts.

Models have been at the heart of much of human understanding and enquiry from very ancient times. Cultures very often attempt to explain the world and human beings' place in it by means of complex mythologies. Such mythologies are essentially abstractions—etiological fables (Carruthers, 1998)—that allow complex and inexplicable phenomena to be understood in terms of a more accessible set of characters and stories set around them. Very often the underlying explanation of phenomena will map onto supernatural beings and phenomena which thus replace unfathomable cause with commonly held narrative.

With time, more rigorous forms of modelling were invented. The ancient Mesopotamians developed sophisticated mathematics as a technique for modelling trade involving large numbers of items and customers (Davis & Hersh, 1983). This early theory of mathematics was thus being used to build abstract models of trade and stock control. The ancient Greeks and following them the Arabic world continued to develop models—mathematical and otherwise—for a variety of phenomena ranging from cosmology to music and poetry. Meter and rhyming schemes for poetry, for example, are models that facilitate the construction of new poems within established forms. This leads us naturally to ask what we mean by the term model, and how and why models are so generally useful?

The Concise Oxford dictionary variously describes a model as "a representation of structure"; "a summary, epitome, or abstract"; and "something that accurately resembles something else." Formal logic uses the term *model* to mean the system of rules by which meaning is mapped onto the syntactic constructions expressed within a particular logic. It is thus possible for a model to be highly formal—that is, expressed in mathematics—or highly informal, but not presumably both. Scientific models may be more pragmatic in that they are related to some aspect of reality by means of observational data, which in turn causes the hypothesis upon which the model is constructed to be reformulated and so on. In other words they are empirical rather than strictly formal and thus sit somewhere between the extremes of the formal-informal axis.

As already mooted with respect to etiological fables, models may be quite instrumental in the sense that the application of the model as an analysis technique—and the results obtained therein—may be more important than the degree to which the model accurately reflects reality; psychoanalysis is an obvious example. Semiotics (Chandler, 2002) is perhaps another case in point because it has never been ascertained whether or not signs as defined by semioticians actually represent structures or functions within the human brain. There is some evidence to support this (e.g., Damasio, 1994). Nonetheless, semiotic analysis of communications artefacts—texts to semioticians—is a very valuable and general technique for gaining insights into the way in which humans communicate and make meaning using a whole range of media. Semiotics is very important to this chapter.

With respect to Kronlof's characterization of method, we can see that the term model is used in two rather different ways:

1. An 'underlying model' is a semantic structure to which terms of the language of the method are mapped in order to assign meaning to them.
2. A 'process model' is an abstract representation of the activities undertaken as part of the model along an expression of their ordering.

The first use of the term model given above is a formal notion, while the second is the more intuitive notion of an abstraction of some more complex system, both discussed in our aside above. If we were to take the language and its underlying model together, we would arrive at the second form of model which is essentially a notation for simplifying and elucidating a more complex system. But what language and underlying model are we to use for VE design? The role of the former is to facilitate the creation and expression of design decisions. The role of the latter is less obvious, but its nature has a direct bearing on the applicability of the method in general. The two parts of this question are addressed in the succeeding sections of this chapter.

The process model of a method is most often expressed as a simple diagram, a graph where the nodes name particular activities and the arcs indicate the relative ordering over time of these activities. The graph is thus a focused simplification of a complex set of activities and the relationships between them and their products. What process model might be suitable for VEs? Kaur (1998) put forward a tentative process model for VEs as an ordered list of activities. These activities and their ordering were deduced from questionnaire data drawn from a limited number of VE developers. Fencott (1999b) put forward a process model that was more representative of the design tensions inherent to VEs. We will return to the process model after the sections devoted to language and underlying model.

Heuristics are essentially advice and guidelines on the successful application of the model to real problems. In terms of VE design, we can observe that there are a lot of such heuristics around in terms of standalone advice that is almost invariably devoid of a methodological context with respect to VEs. There are exceptions to this, the 'SENDA' method of Sanchez-Segura et al. (2003; also, see Chapter 4) for example.

The 'design tension' identified above as the driving force in the methodology of VE design has its antecedents. In the early 1990s there was a debate as to whether formal methods or structured methods for software design were most appropriate. The former use logic and set theory to build mathematical models of software systems, while the latter use diagrams, pseudo code, and other 'non-formal' notations to the similar ends. Integrated methods research attempted to combine these approaches to maximize the strengths and minimize the weaknesses of both (Fencott et al., 1992, 1994). In this chapter we draw on the experiences gained in the earlier research in order to address the design tension directly.

In this section we have posed a number of questions with respect to a possible VE design method:

1. What language and underlying model are we to use for VE design?
2. What process model is appropriate for VE design?
3. What sort of heuristics do we need and are any of those extant adaptable to the model we hypothesize in 1 and 2 above?

In this chapter we specifically deal with Questions 1 and 2. Question 3 will be for future consideration, as it depends on the answers to Questions 1 and 2.

The Underlying Model

The question of what an underlying model might be for a VE design methodology might seem of purely theoretical interest, but attempting to answer it necessitates a consideration of the design tension highlighted in the previous two sections. We have to find an underlying model that expresses the meaning of a VE design in terms of both:

- *Engineering:* as a computer system composed of program and hardware, understood largely by those trained in computer science and related disciplines;
- *Aesthetics:* as an interactive communications medium, understood by those trained in the creative arts.

We appear to have confounded the issue, as we now seem to need an underlying model that not only addresses two different design issues, but that is understood differently by two quite different groups of professionals. Is one underlying model possible, and who on earth is going to understand it? In fact there have been various attempts to reconcile the two with varying degrees of success, but it's useful for our purposes to consider them separately for the time being.

We can begin to suggest possible underlying models, bearing in the mind the tension already identified. VEs and IDSs in general have interaction machines (IMs) as their underlying model (Goldin et al., 2001) in terms of computational functionality, but we also need a model that operates at the perceptual, meaning-making level. Semiotics (Chandler, 2002) is highly appropriate for the latter. Interaction Machines encompass a set of possible computational systems—more expressive than Turing Machines—that allow for the persistence of state and unlimited user inputs that characterize interactive media, IDSs in general and VEs in particular. Semiotics is the study of sign systems and the way humans find meaning in them. The two might not be so incompatible as a cursory glance might seem to suggest. We will briefly consider each separately and then consider their integration.

For much of the latter half of the twentieth century, it was the received wisdom that Turing Machines captured the notion and limits of what is computable. In the 1990s a number of researchers began to develop models which showed that Turing Machines were not expressive enough to model interactive computer systems. In fact it was shown that the simplest interactive program:

```
P := input(x:Boolean); output(x); P
```

which recursively inputs a Boolean value for x and simply outputs that same value, cannot be programmed using any Turing Machine. That this is so even for a very simple datatype such as Boolean might be somewhat surprising. The reason is that although each input and output is finite—a requirement for conventional Turing Machine input—there might be an infinite number of them, and it is impossible to represent such an infinite set of choices on a sequential, yet infinite tape.

Goldin et al. (2001) have shown that Turing Machines can be extended to model interaction by defining Persistent Turing Machines (PTMs), which employ dynamic streams to model inputs and outputs, and a tape to remember the current state ready for the commencement of a new computation. PTMs are an example of the general class of IMs.

PTMs are certainly not the only possible characterization of IMs. We could, for instance, have used an approach based on concurrent systems in the manner of Milner (1989). In many respects this would be better as it not only captures the notion of VEs as IMs, but also allows us to consider them as being the composition of a number of embedded systems—autonomous agents and non-playable characters, for instance. PTMs are, however, better suited as a brief illustration of the concept for our present purposes.

Human beings ceaselessly work to find meaning in any situation they might find themselves, in any communications media they might find themselves using, and in even mundane situations such as walking down the street or sitting on a train or bus. Semiotics is the study of this meaning-making process, and signs are the basic unit of the theory (e.g., Eco, 1977; Barthes, 1987). The most common characterization of signs consists of two components, a:

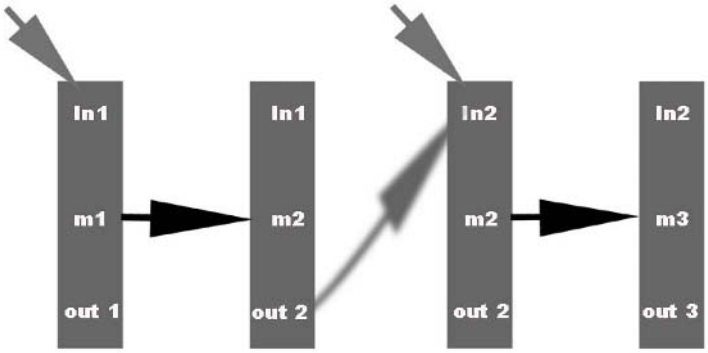
- *Signifier:* that which we can perceive in the world around us using any of our senses;
- *Signified:* the meaning(s) we form in our minds as a response to perceiving the signified.

Communications artefacts, texts to semioticians, are made up of signs and can be anything we humans find meaningful, for instance: novels, films, body language and facial expressions, and VEs.

Semiotics provides us with a means of understanding the output of a VE, the digital displays, and the signs of intervention, as we shall call them, that the user generates by means of the input technology. VEs are a particular form of IM that attempt to restrict its users' environments to the digital displays it generates in response to user input. We thus have a partially closed system. Semiotics can provide a means of analysing how a user might make meaning out of such a system and thus make meaningful choices about how to interact with it. We can thus refer to our underlying model as a Semiotically Closed Interaction Machine (SCIM).

Figure 1 shows the relationship between semiotics and IMs. The two downward pointing arrows represent inputs by the user, in1 and in2. The horizontal, black

Figure 1. Semiotically closed interaction machines



arrows represent computation steps that result in the generation of new outputs, out2 and out3. The fuzzy, curved arrow represents the semiotic closure between out2 and in2; in other words the cognitive process of finding meanings in out2 and formulating a response to them as in2. On the one hand we have the human, meaning-making process and on the other the non-semiotic act of using the signs of intervention to create a new input to the IM and thus instigate a further macro-computation step. Note that the diagram is a simplification, as in VEs in general outputs may also be produced without direct input from the user.

In SCIMs such as VEs, the semiotic link is very strong, whereas in IDSs in general, the link may be far weaker and intermittent. There is no semiotic link between individual customer transactions at an ATM, for instance. There is also no recognizable semiotic link between a customer inserting his or her debit card, the PIN input, and the amount of money requested; ATMs are not SCIMs.

Both IMs and semiotics are appropriate as a choice of an integrated, underlying model because they do not constrain us to particular programming languages or computational platforms on the one hand, nor particular modes of communication on the other. That will be the business of the next section when we consider the nature of a language suitable for expressing VE design decisions.

An integrated underlying model is not the only approach. There is a field of enquiry called computational semiotics that has as one of its concerns the integration of semiotics and computer science; this can operate at the level of the underlying model or at the level of language within a methodological context while sometimes at both. For instance, Goguen (1999) defines ‘algebraic semiotics’ as semiotics formalized using the algebraic specification language OBJ. He has outlined the application of this formalism to user interface design and VE design. As another example, Doben-Henisch (1999) has attempted to integrate semiotics with Turing Machines. The problem with the latter is that Turing Machines are not expressive enough to model VEs. The problem with the former as an underlying model for VEs is that the formalism makes use of difficult mathematical concepts, such as category theory, which obscure the insights into the nature of VE that our integrated approach highlights—difficult, that is, for those VE designers without a strong mathematical background.

The integrated underlying model we have adopted is a very practical one, as it preserves the 'design tension' rather than allowing the engineering or the aesthetic dimension to dominate.

The Language

We move now to the nature of languages for expressing VE design decisions. A review of existing work on VE design (Fencott, 2003b) reveals that while there is a quantity of research and commentary on the human factors affecting design, for instance, there is very little that is directly relevant to VE content modelling, which is at the heart of this chapter. There are examples of the construction and application of methods or guidelines for realizing certain aspects of VE design; some of these are:

1. Various work on usability for VEs (e.g., Workshop on Usability Evaluation of Virtual Environments, 1998)
2. Structured methods for VEs (e.g., Workshop on Structured Design of Virtual Environments, 2001)
3. Various commentaries from the computer games world (e.g., Gammasutra, Rollings, & Adams, 2003)
4. Semiotics of games and new media (e.g., Lindley et al., 2001)

In light of the discussion in the previous section, we can make the following observations: 1 and 2 are insufficient to express VE design decisions because they do not address aesthetics adequately; 3 provides some very useful insights; 4 gives us a way to alleviate the inadequacies of 1 and 2.

If we continue with the integrated approach adopted for the underlying model in the previous section, we need a language to express the programming (the engineering) side of a VE and one to express its aesthetic dimension. The standard for the former should, most likely, be some form of object-oriented programming language and the standard methodology for such languages is the Unified Modelling Language (UML). In fact, Goldin, Keil, and Wegner (2001) document the suitability of UML as a language for expressing designs that have IMs as their underlying model. UML would seem a good choice of language for this aspect of VE design.

Aesthetics of VEs has been a constant theme of this chapter, and we now discuss them in some detail. Church calls for a set of “formal, abstract, design tools” (FADTs) that will not only guide the design of successful games, but which will also enable designers to compare and contrast computer games from diverse genres (Church, 1999). Church’s FADTs are perhaps better understood as an aesthetic characterization of computer games and are:

- *Intention:* being able to establish goals and plan their achievement;
- *Perceivable consequence:* a clear reaction from the game world to the action of the player;
- *Story:* the narrative thread, both designer-driven and user-driven, that binds events together.

Other computer games designers talk in a similar vein: of players needing to feel in control, of maintaining the emotional feel of a game and/or level, of providing suitable and timely rewards for effort, and of a perceivable gross structure that allows players to identify what is required of them at the beginning of a level, plan to achieve this, and understand the significance of their achievement (Saltzman, 1999). Intentions and perceivable consequences are the building blocks for this.

Brenda Laurel introduced the term ‘narrative potential’ to capture the idea that VEs can offer users the possibility of building their own stories out of virtual experiences (Laurel, 1992). We will adopt narrative potential rather than ‘story’ as part of the aesthetics of VEs.

From the field of media studies, Murray (1996) identifies the following aesthetic characterization of interactive media as:

- *Immersion:* the feeling of being completely absorbed (almost literally immersed) in the content (we will use the term presence for reasons detailed below);
- *Agency:* being able to affect change in the VE;
- *Transformation:* being able to become someone or something else.

Lombard and Ditton (1997) define presence as *the perceptual illusion of non-mediation*. This characterizes presence as the state of mind of a visitor to a VE as not noticing or choosing not to notice that that which they are experiencing and interacting with is artificially generated. They document the evaluation of the embodying interface of a VE in terms of presence seen largely as the degree of fidelity of sensory immersion. Much of the research to date into presence is particularly concerned with the embodying interface as well as researches into the mental state of people who are present in VEs. Immersion is thus the degree to which the technology of the embodying interface mediates the stimuli to the senses. Slater has shown that high degrees of sensory immersion heighten the emotional involvement with a VE (Slater et al., 1999).

However, as presence is a mental state, it is therefore a direct result of perception rather than sensation. In other words, the mental constructions that people build from stimuli are more important than the stimuli themselves. It is the patterns that we, as VE constructors, build into the various cues that make up the available sensory bandwidth for a given VE that help or hinder perception and thus presence. These patterns are the result of what is built into the VE and the way the user behaves in response to them. The fidelity of the sensory input is obviously a contributing factor, but by no means the most important. In the context of the working VE builder, being able to identify and make effective use of the causes of presence is more important than the nature of presence itself. This means that it is the effective consideration of the perceptual consequences of what we build into VEs that will give rise to the sense of presence that we are looking for. In this sense it is the content of VEs that has the greatest effect on the generation of presence. Thus, for our purposes, content is the object of perception.

Agency is the fundamental aesthetic pleasure of VEs and IDSs in general and the one from which all the others derive. Agency actually equates quite nicely to Church's intention and perceivable consequence; agency is in part the interplay between intention and perceivable consequence.

Transformation is important to many communications media. One of the great pleasures of novels is seeing the world through someone else's eyes, to view the world through the eyes of another creature, machine, or alien being. VEs in particular are ideally suited to this, and much of the success of 3D computer games is due to the player being able to be the hero or villain in some great and dangerous adventure. In such games the player cannot only play an alien, but through the real-time graphics actually see the world as the alien would see it. It seems certain, for instance, that one of the reasons for the success of the classic Hubble Space Telescope Virtual Training Environment (Loftin et al., 1994) was that members of the ground-based flight team could actually become astronauts for a while, and experience some of the drama and spectacle of a space walk. To the author's knowledge and despite the insightful research into the effect and effectiveness of the Hubble, the question "Did you enjoy being an astronaut for a change?" was never asked. Yet it seems highly likely that this was a major experience for the subjects.

Finally, in this brief review of aesthetics for IDSs, we must include Turkle's (1995) observation that being present with others—sentient beings, robots, creatures, and autonomous agents in general—is something that has drawn users to IDSs since the earliest days of Eliza and MUDs.

Bringing these various aesthetic viewpoints together, we can characterize the aesthetics of VEs as:

- *Agency:* which itself consists of:
    - *Intention:* being able to set goals and work towards their attainment.
    - *Perceivable consequence:* being rewarded for one's mental and virtual activity by sensing the VE change appropriately as a result of the actions taken.
- *Narrative potential:* the sense that the VE is rich enough and consistent enough to facilitate purposive experience that will allow the user to construct her own narrative accounts of it.
- *Co-presence:* being present with others.
- *Transformation:* temporarily becoming someone or something else as a result of interacting with the VE.
- *Presence:* the perceptual illusion of non-mediation (Lombard & Ditton, 1997).

In terms of underlying theory, aesthetics are signifieds of a particular type; they are connotations that arise from interacting with VEs. Connotations, in semiotic theory, are deeper levels of meaning that humans build up from the level of denotation: the commonplace or everyday meanings of things.

On a more concrete level, Murray (1996) equates the structure of interactive media with the notion of the labyrinth and asserts that this structure works best when its complexity is somewhere between the 'single path maze' and the 'rhizome' or entangled Web. Aarseth (1999) has proposed the notion of cybertext to capture the class of texts, not just digital, which require the visitor to work to establish their own path(s) through the possibilities offered. He calls this class of text ergodic from the Greek words meaning work and path. So we have a notion of a labyrinth that requires effort to explore. Equating the structure of VEs in general with the notion of a labyrinth of effort would seem useful, but poses several questions. First of all, what are the actual components with which VE designers build such experiential labyrinthine structure? Second, how do VE designers structure a VE so that the visitor follows an appropriate path and, moreover, accumulates an appropriate set of experiences so as to discover and remember the intended purpose of the VE?

Fencott (1999a, 2003a, 2003b) draws on these various aesthetic views to define a model of VE content, Perceptual Opportunities (POs), which focuses on the aesthetic design of the perceptual experiences over time which users are intended to accumulate.

Figure 2. Perceptual opportunities

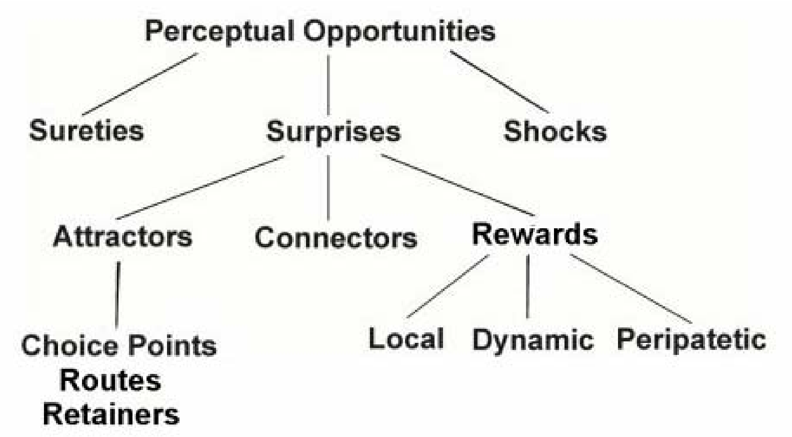


Figure 2 characterizes the breakdown of POs in terms of:

- *Sureties*: designed to deliver belief in a VE, equated with unconscious experience (e.g., Spinney, 1998; Blackmore, 1999)
- *Surprises:* designed to deliver the essential purpose of the VE
- *Shocks:* perceptual bugs that undermine the first two

Surprises are further broken down into:

- *Attractors:* literally content that attracts attention
- *Connectors:* content that supports the achievement of goals
- *Rewards:* content that literally rewards users for effort

Attractors can be characterized in two ways: By the way they attract attention—they might be mysterious, awesome, active, alien, complex—collections of attractors—and so on. They can also be characterized by the basic emotions they stimulate, typically fear and desire. Rewards can be information, access to new areas of the VE, new activities enabled, and so on. Connectors can be as simple as railings, footpaths, and street signs, but can also be dynamic maps, indicators of health, wealth, and so on. Attractors are the means by which users are led to form intentions. The perceivable consequences of a player attempting to realize an intention leads to the identification of rewards which leads to the identification of new attractors and so on. Thus agency and POs are very strongly associated.

POs can be organized into higher level structures, perceptual maps, which characterize patterns of behaviour that users exhibit when interacting with a VE.

A perceptual map can be made up of:

- *Choice points:* basically the choice between intentions stimulated by one or more attractors.
- *Challenge points:* intentions that have to be satisfied.
- *Routes:* linear sequences of attractors.
- *Retainers:* mini-missions or mini-games, tightly grouped attractor-reward pairs, puzzles, and so forth.

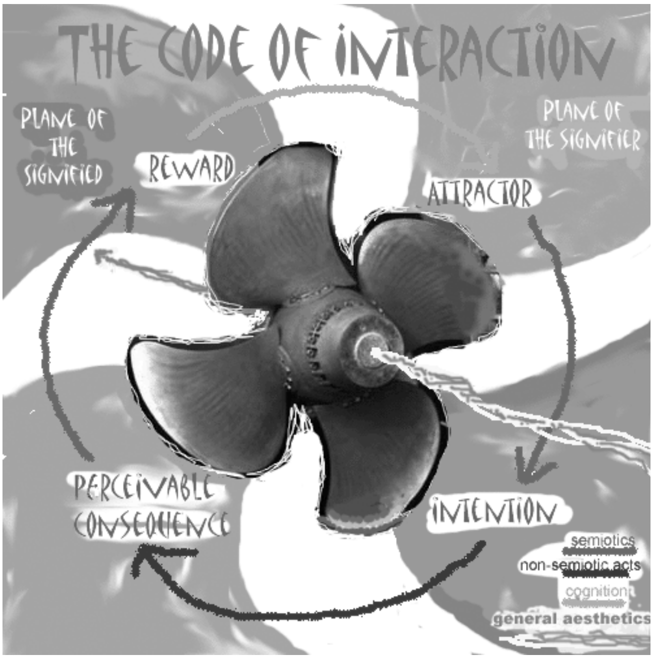
The arrangement of such structures are thus a realization of Murray's rhizome and lead to the other aesthetic pleasures of narrative potential, co-presence, transformation, and presence.

In a later publication, the same author asserts that POs, as well as the aesthetics identified above, have semiotics as their underlying model (Fencott, 2003b). Essentially, POs and in particular surprises are connotations that humans derive through interacting with VEs. POs interface very closely with the aesthetic pleasure of agency, but at a more abstract level of VE content. Figure 3 illustrates the relationships between POs, aesthetics, and semiotics at the level of the language of a VE design method:

- The two arrows linking attractor and intention and perceivable consequence and reward are semiotic acts, meaning making, on the part of people interacting with a VE. Attractors and perceivable consequences are signifiers, while intentions and rewards are signifieds.
- The arrow linking reward and attractor indicates cognition, though of course cognition is a continuous process and not a segment of a cycle as this diagram would seem to suggest.
- The arrow linking intention and perceivable consequence represents what Tronstad (2001) calls non-semiotic acts that are essentially the site of the IM, the computer-based system in the wider IDE. The term non-semiotic is used because, while the user might draw some significations from pressing interface buttons and so on, the computer responds algorithmically.
- The arrow that runs through the cyclic plane of the above relationships from right to left represents the development over time of the other aesthetic properties of narrative potential, co-presence, transformation, and presence.

On the level of aesthetics and POs, we see the following. Having formed an intention, a user will provide input to the VE, which will trigger the execution of one or more calculations, non-semiotic acts on the part of the computer. This will

Figure 3. The code of interaction



result in a change (perceivable consequence) in the various digital elements of the VE's display which provide signifiers (rewards) to start off the whole semiotic and cognitive process once again through the identification of attractors.

Figure 3 shows quite clearly the dependant relationship between semiotic and non-semiotic acts, which Tronstad (2001) sees as being fundamental to interactive digital experience. If we compare Figure 3 with Figure 1, we see that what has changed is that the arrow that represented the semiotic closure of the output and input step in the latter has been dramatically expanded in the former; it is almost as if it has been turned 'inside out'. Figure 3 characterizes the 'code of interaction'. In semiotics, codes are the often innate rules that allow us to make meaning of signifiers. Interaction is a complex process and the diagram reflects this. So much so, in fact, that Fencott (2004) devotes a whole chapter to the 'code of interaction. In the context of our present discussions, the various components and the relationships between them that make up the code constitute the general aesthetic side of the language of our method.

Semiotics not only provide an underlying model for POs and aesthetics, they also operate at the level of the language of a method as well. In interacting with VEs we not only recognize the code of interaction—connotations specific to VEs and IDSs in general—but we also find meanings that correspond to world of the 'real' outside the world of the VE. We recognize shops and cars and people and furniture and so on and so on. It is semiotics itself that is used as 'language' in this type of meaning-making.

Therefore, in addressing the second question concerning the nature of the language of a VE design method, we now need to consider how POs, aesthetics, and semiotics on the one hand and UML on the other might work together. In this respect, the central issue that needs to be addressed concerns what we might call the 'object problem'. Objects or rather object-oriented design (OOD) might seem a very promising candidate for our language for representing VEs at the design phase. OOD applies at all stages in the VE production lifecycle, addresses both coding and user-cantered issues, and has been applied directly to VE design and implementation (McIntosh).

However, in the act of perception, people do not break the world down into nicely programmable units. They group things together into perceivable units, complex attractors, which focus their attention. A crowd of autonomous agents—non-playable characters (NPCs) in computer games parlance—are perceivable as a single entity, but are unlikely to be a single object in an OO model. Certainly a crowd of NPCs in a busy shopping centre with all its shop fronts, street furniture, paving, and so on is not going to be an object in an OO specification for a shopping centre. However, each of the entities that makes up the perceivable unit that is the crowded shopping centre will have to be identified in terms of its capacity (or not) for interaction as a basis for its incorporation into a functioning scene graph.

On the one hand, we have the Unified Modelling Language (UML), which models structural, engineering aspects of a scene graph, and on the other hand, we have POs and so on which model content at the level of perception, of aesthetics (Fencott, 1999b, 2003b). There is in fact a bridge, a semiotic bridge, which links the two, and this is Andersen's Computer Based Signs (CBSs) (Andersen, 1997), which model interactive aspects of individual signs (objects) in IDSs in general and thus VEs. Fencott (2003a) discusses this relationship and its relevance to VE design.

Andersen arrives at the following classification of signs in IDSs:

- *Interactive:* signs that can be controlled by the user and can affect other signs; such signs are subject to the signs of intervention.
- *Actor:* signs that to a limited extent are autonomous and can affect other signs.
- *Controller:* signs that constrain other signs but do not themselves change nor can they be affected by other signs.
- *Object:* signs that can be affected but cannot affect others.

- *Ghost:* a sign that affects others, but only becomes apparent by its effects on others; a sign particular to IDSs. Essentially a controller that signifies its presence solely through effect.
- *Layout:* non-interactive signs.

CBSs essentially constitute six distinct classes that will be used to instance all objects in a VE implementation. The integration of the three elements of our language of VE design can now be summarized thus:

- Each content item in the perceptual model is assigned to a CBS class.
- Other aesthetic attributes of content items carry over directly to UML, that is, colour, form, and so on.
- General information in the perceptual model carries over to UML to become the game engine, the visualiser.
- Other such information carries over directly to UML in terms of the semiotic realization of the VE: mood, myths, and hyperrealities.

So the language of our VE design method is an amalgam of OO and POs and so on—with some bridging by CBSs. It is, in fact, an integrated method, a process, rather than a statically characterisable relationship. In this way we have carried the design tension identified early in this chapter, and clarified through to the language stage. It seems that we might be able to make this tension work for us rather than it being a hindrance to try to do away with.

The Process Model

The process model captures the relationship over time between the constituent activities of a method. In a sense it captures the essence of the 'practice of methodology', the choosing of how to apply a method. As part of a study of VE design practice, Kaur (1998) constructs the following outline VE design methodology:

1. requirements specification;
2. gathering of reference material from real-world objects;
3. structuring the graphical model and, sometimes, dividing it between designers;
4. building objects and positioning them in the VE;

5. enhancing the environment with texture, lighting, sound, and interaction, and optimising the environment.

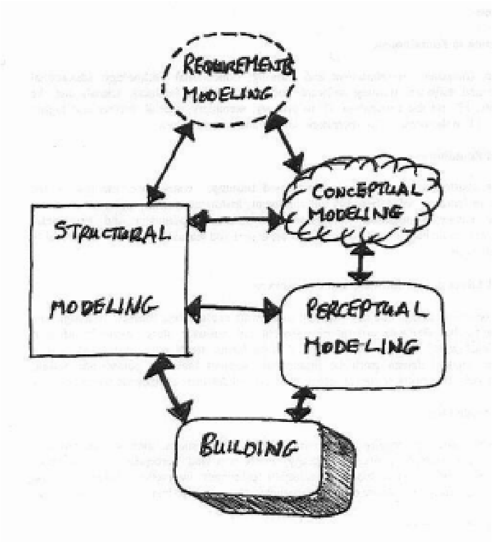
She also notes that there might be a narrative design component missing here, but this is probably because of the small scale of the VEs in the study. Certainly the narrative aspects of 3D game design are considered as soon as the principle subject and genre are established. Computer games are almost certainly the major examples of VEs large enough to benefit from software engineering practice.

With these arguments in mind, Fencott (1999b) offered a prototype design methodology for VEs which attempts to resolve the two-sided design problem for IDEs by juxtaposing structural and perceptual modelling, and attempting to empathize with current practice. The methodology is also based on practical experience gained in building a variety of desktop VEs, and in particular a virtual tourism project, as well as teaching VE design to several hundred undergraduate and master's students over a number of years. Figure 4 characterizes this suggested process model, and we now go on to revisit the original, tentative discussions that were offered in the 1999 paper, in the light of the discussions concerning the possible underlying model and language required for a design method for VEs laid out above.

In terms of the design tension, the route down the left-hand side of the diagram represents engineering design and the route down the right-hand side represents aesthetic design. The horizontal arrows represent interactions that seek to resolve the tension.

- **Requirements modelling** equates to Point 1 in Kaur's methodology above and parallels very closely the software engineering concept. One of the chief requirements is that purpose should be clearly established here. In terms of our integrated underlying model, we might here conduct as 'use case analysis' in UML and commence the analysis of our intended VE in terms of Barthe's notion of *myth*—connotations so seemingly natural as to be unquestioned (Barthe, 1987)—and perhaps Baudrillard's notion of hyperreality (Baudrillard, 1995). Both are concerned with the cultural basis upon which a VE's belief system will be grounded. At this stage we are thus making direct use of the underlying model and the techniques associated with it. The semiotic and software engineering viewpoints are left unresolved until the latter stages of structural and perceptual modelling.
- **Conceptual modelling** equates to Point 2 in Kaur's methodology and is effectively the background research activity common to many design projects, but in particular those with an aesthetic component. It is the gathering of materials, taking of photographs, sketches, sound and video

Figure 4. A process model for VE design



recordings, and so forth. It might also include the construction of mood boards as well as potential storyboards. This is where the VE builder or builders get to know the world they have to build. Note that the world to be built might have no real-world counterpart, which will of course impact on the kinds of activities that might be undertaken here. The artists' accounts and the techniques employed by animators are sources of applicable techniques (e.g., Moser, 1996). An important outcome of this stage will be a choice of genre, to best achieve the purpose established at the requirements stage, with which to inform the nature of the meta-narrative structure to be developed in the perceptual modelling phase.

The end of this phase is effectively concerned with the semiotic activity of translating the decisions concerning myth and/or hyperreality—from the requirements phase—into connotation, metaphor, and metonymy.

- **Perceptual modelling** is the act of building up a model of the nature of the perceptual opportunities and their inter-relationships. It equates very roughly to Point 5 in Kaur's methodology. It is of course modelling the intended users' experience of the VE. In Fencott (2003a) perceptual maps, for instance attractor graphs, are used to build up a meta-narrative structure of POs, analogous to the comprehensible labyrinth of Murray (1997), which are categorized according to the role they play in the planned scheme of possible user activity. Perceptual opportunities deal not only with conscious experience—derived from *the specifically designed infidelities* of Whitlock et al. (1996)—but also with unconscious experience, sureties, which deliver belief in the VE—perceptual realism in Lombard and Ditton (1997)—irrespective of any real-world counterpart. The existence and importance of unconscious experience is identified and modelled by considering sureties.

- **Structural modelling**, Point 3 in Kaur's methodology, covers a variety of activities that relate to the underlying realization of the VE that the delivery platform uses to construct the run-time sensory stimuli. Structural modelling would seem to commence alongside conceptual modelling and to run on alongside perceptual modelling. It starts with decisions on scale, the construction of plans, and diagrams. It draws on Andersen's CBSs to further decompose the perceptual map constructed in the perceptual modelling phase in terms of the way in which particular objects implement gross structure of attractors and rewards identified in perceptual modelling.

  The conclusion of the structural modelling phase will result in a scene graph diagram that lays out the code structure of the VE and its programmed behavioural components. In terms of software engineering practice, UML has already been identified as a candidate language here. In later stages, object models would lay out the actual structure of nodes in the scene graph as well as class diagrams for programmed components.
- **Building** here relates more closely to the software engineering coding phase that should occur after all requirements, specification, and design activities have been completed. Building refers to authoring using a WIMP-based tool, direct coding of scene graph and program code itself, in VRML and Java/Javascript for example, and using an API such as World Tool Kit.

We will now consider some of the flows (arrows) in this process model, first of all the structural-conceptual flow. The conceptual modelling stage can deliver important high-level plans for the layout of the VE as well as the principle entities that will need to be present to reinforce the results of *use-case-analysis*, for instance. The structural-perceptual flow delivers object denotations to do with such attributes as appearance and sound. It will deliver object connotations concerned with the way objects contribute to the overall purpose of the VE. Importantly it will also—via CBSs—deliver attributes concerned with interactive capabilities of objects.

Finally, we note that we do not address the question of heuristics in this chapter. There are two reasons: first of all our method is too methodological at the moment to be able to be supported by practical advice; secondly there is a wealth of help and advice on VE design and games design in particular, and it will be necessary to investigate how it might integrate with the method under consideration.

Future Design Methodology

It is the author's suspicion that one of the foreseeable trends will be ever more sophisticated VE authoring tools, which will mean that the explicit use of OO techniques such as UML will be more and more hidden from the author. Much of the time the scene graph, whether at the level of an OO specification or at the level of OO coded implementation, will only be available to authors via specific views rather than as a coherent whole. More and more it will be the perceptual modelling and the interface between this and the structural views that will be made explicit and malleable. The process model discussed above shows the nature of this interface and provides clues as to how this might be supported by authoring technology.

However, in terms of authoring tools, there is a serious problem that we have not identified nor discussed so far. This is the problem of authoring agency, which lags far behind the authoring possibilities on offer for 3D modelling, texture mapping, shading, and rendering, to name but a few. Nothing approaching the sophisticated tools on offer for these exists for authoring agency. Typical examples of this are easy to find in a wide range of VE authoring tools for both games and VR. In the excellent Unreal Editor for example, the only agencies we can easily implement are such concerned with opening doors, travelling in lifts (elevators), and shooting guns. Unreal is a *first-person-shooter* and it has in-built agencies typical of its genre. If an author wants to implement additional agency, then she has to program it in Unrealscript, a Java variant.

Yet a theoretical analysis of games genres has shown that agency is exactly what characterizes games (Fencott, 2004). Any game design method should not only incorporate the analysis and design of appropriate agency in its process model, but should encourage authors to reconsider it throughout the lifecycle from early requirements analysis through to later modelling stages. By focusing on agency in terms of the aesthetic pleasures of intention and perceivable consequence, and in terms of the POs of attractors and rewards, the process model does indeed ask the designer to consider agency in a fundamental way that authoring tools do not at present support.

In terms of underlying theory, two significant trends can be identified. The growing interest in the investigation, formalization, and application of interaction machine theory (e.g., Goldin et al., 2001) and the emergence of semiotics and computational semiotics as a tool to analyse and design VEs and IDSs in general—for example the COSIGN series of conferences (COSIGN). Of particular interest will be the further investigation of the possible integration of interaction machines and semiotics, which in effect amounts to the nature of the interplay between empirical computer science and interactive media aesthetics. As has already been pointed out, the tempting approach is to formalize semiotics as computation (e.g., Dogen-Henisch, 1999; Goguen, 1999), but this does not capture or investigate the playfully surprising relationship people have with IDSs and IDEs in particular.

Conclusions

A little tension can be a good thing; too much can be very destructive. We need to keep the VE design tension apparent throughout the analysis, conceptual, and perceptual design stages to be more or less resolved in the structural modelling stage. It must not be allowed to tear the process apart. On the other hand an imbalance biasing one pole of the tension or the other will result in an equally unbalanced VE—either well engineered and boring, or fascinating but badly made. The author believes that the design tension will manifest itself in a benign way in a well-designed VE and that users will recognize and appreciate that manifestation.

In effect VE design methodology is encouraging us to confront and meld a great rift in contemporary Western culture, namely that between the arts and the sciences. Of course, at present it is inviting us to do this in terms of two particular forms of abstraction which represent the two sides of the divide. That we should confront reality through virtual reality might come as a surprise, but the concept has been around since the early days of virtual environments and was clearly articulated by Lauria (1997) when she envisioned virtual reality as a ‘metaphysical testbed’.

On a less grandiose scale, it may well be that no design method for VEs ever becomes a real practicality or if it does is ever widely adopted by the developer community. Surely, however, the investigation of the methodology of VE design will inform us far better than we are now as to the fundamental nature of VEs and thus be of benefit to us when we come to design future interactive systems.

References

Aarseth, E.J. (1997). *Cybertext: Perspectives on Ergodic literature*. Baltimore, MD: John Hopkins University Press.

Andersen, P.B. (1997). *A theory of computer semiotics*. Cambridge University Press.

Barthes, R. (1987). *Mythologies*. New York: Hill and Wang.

Baudrillard, J. (1995). The Gulf War did not take place (trans Patton P.). Indiana University Press.

Blackmore, S. (1999). *The meme machine.* Oxford University Press.

Carruthers, M. (1998). *The craft of thought: Meditation, rhetoric, and the making of images, 400-1200.* Cambridge University Press.

Chandler, D. (2002). *Semiotics*: *The basics*. Routledge.

Church, D. (1999). Formal abstract design tools. *Games Developer Magazine*, (August).

COSIGN. Retrieved from: *www.cosignconference.org*

Damasio, A.R. (1994). *Descarte's error: Emotion, reason and the human brain.* Papermac.

Davis, P.J., & Hersh, R. (1983). *The mathematical experience*. Pelican Books.

Doben-Henisch, G. (1999). Alan Mathew Turing, the Turing Machine, and the concept of sign. Retrieved from: *www.inm.de/kip/SEMIOTIC/DRESDEN_FEBR99/CS_Turing_and_Sign_febr99.html*

Eco, U. (1977). *A theory of semiotics.* Macmillan Press.

Fencott, C. (1999a). Content and creativity in virtual environment design. *Proceedings of Virtual Systems and Multimedia '99,* University of Abertay Dundee, Scotland.

Fencott, C. (1999b). Towards a design methodology for virtual environments. *Proceedings of the International Workshop on User Friendly Design of Virtual Environments,* York, UK.

Fencott, C. (2003a). Virtual saltburn by the sea: Creative content design for virtual environments. *Creating and using virtual reality: A guide for the arts and humanities*. Oxbow Books, Arts and Humanities Data Service.

Fencott, C. (2003b). *Perceptual opportunities: A content model for the analysis and design of virtual environments.* PhD thesis, University of Teesside, UK.

Fencott, C. (2004). *Game invaders: Computer game theories.* In preparation.

Fencott, P.C., Fleming, C., & Gerrard, C. (1992). Practical formal methods for process control engineers. *Proceedings of SAFECOMP '92*, Zurich, Switzerland, October. City: Pergamon Press.

Fencott, P.C., Galloway, A.J., Lockyer, M.A., O'Brien, S.J., & Pearson, S. (1994). Formalizing the semantics of Ward/Mellor SA/RT essential model using a process algebra. Proceedings of Formal Methods Europe '94. *Lecture Notes in Computer Science, 873.* Berlin: Springer-Verlag.

Gammasutra. Retrieved from: *www.gamasutra.com*

Goguen, J. (1999). An introduction to algebraic semiotics, with application to user interface design. Computation for metaphor, analogy and agents. *Springer Lecture Notes in Artificial Intelligence, 1562,* 242-291.

Goldin, D., Keil, D., & Wegner, P. (2001). An interactive viewpoint on the role of UML. *Unified Modelling Language: Systems analysis, design, and development issues.* Hershey, PA: Idea Group Publishing.

Goldin, D.Q., Smolka, S.A., Attie, P.C., & Wegner, P. (2001). Turing Machines, transition systems, and interaction. *Nordic Journal of Computing.*

Kaur, K. (1998). *Designing virtual environments for usability*. PhD Thesis, City University, London.

Kronlof, C. (1993). *Methods integration: Concepts and case studies.* New York: John Wiley & Sons.

Laurel, B. (1992). Placeholder. Retrieved from: *www.tauzero.com/Brenda_Laurel/Placeholder/Placeholder.html*

Lauria, R. (1997). Virtual reality as a metaphysical testbed. *Journal of Computer Mediated Communication, 3*(2). Retrieved from: *jcmc.huji.ac.il/vol3/issue2/*

Lindley, C., Knack, F., Clark, A., Mitchel, G., & Fencott, C. (2001). New media semiotics—computation and aesthetic function. *Proceedings of COSIGN 2001,* Amsterdam. Retrieved from: *www.kinonet.com/conferences/cosign2001/*

Loftin, R.B., & Kenney, P.J. (1994). *The use of virtual environments for training the Hubble Space Telescope flight team*. Retrieved from: *www.vetl.uk/edu/Hubble/virtel.html*

Lombard, M., & Ditton, initial. (1997). At the heart of it all: The concept of telepresence. *Journal of Computer Mediated Communication*, *3*(2). Retrieved September 1997 from: *jcmc.huji.ac.il/vol3/issue2/*

McIntosh, P. Course notes on UML/VRML. Retrieved from: *www.public.asu.edu/~galatin/*

Milner, R. (1989). *Communication and concurrency*. Englewood Cliffs, NJ: Prentice-Hall.

Moser, M.A. (1996). *Immersed in technology*. Boston, MA: MIT Press.

Murray, M. (1997). *Hamlet on the Holodeck: The future of narrative in cyberspace.* New York: The Free Press.

Rollings, A., & Adams E. (2003). *Andrew Rollings and Ernest Adams on games design.* New Riders.

Ryan, T. (1999). Beginning level design. Retrieved from *www.gamasutra.com*

Saltzman, M. (ed.). (1999). *Games design: Secrets of the sages.* Macmillan.

Sánchez-Segura, M.I., Cuadrado, J.J., de Antonio, A., de Amescua, A., & García L. (2003). Adapting traditional software processes to virtual environments development. *Software Practice and Experience, 33*(11). Retrieved from: *www3.interscience.wiley.com/cgi-bin/jhome/1752*

Slater, M. (1999). Co-presence as an amplifier of emotion. *Proceedings of the Second International Workshop on Presence*, University of Essex, UK. Retrieved from: *www.essex.ac.uk/psychology/tapestries/*

Spinney, L. (1998). I had a hunch…. *New Scientist,* (September 5).

Trondstad, R. (2001). Semiotic and nonsemiotic MUD performance. *Proceedings of COSIGN 2001,* CWI, Amsterdam.

Turkle, S. (1995). *Life on the screen: Identity in the age of the Internet.* Phoenix.

UML Version 1.1 Summary. Retrieved from: *www.rational.com/uml/resources/documentation/summary/*

Whitelock, D., Brna, P., & Holland, S. (1996). *What is the value of virtual reality for conceptual learning? Towards a theoretical framework.* Retrieved from: *www.cbl.leeds.ac.uk/~paul/papers/vrpaper96/VRpaper.html*

Workshop on Structured Design of Virtual Environments. (2001). *Proceedings of Web3D Conference,* Paderborn, Germany. Retrieved from: *www.c-lab.de/web3d/VE-Workshop/index.html*

Workshop on Usability Evaluation for Virtual Environments. (1998). De Montforte University. Retrieved from: *www.crg.cs.nott.ac.uk/research/technologies/evaluation/workshop/workshop.html*

Chapter IV

SENDA: A Whole Process to Develop Virtual Environments

Maria-Isabel Sánchez-Segura
Carlos III Technical University of Madrid, Spain

Angélica de Antonio
Universidad Politécnica de Madrid, Spain

Antonio de Amescua
Carlos III Technical University of Madrid, Spain

Abstract

The use of virtual environments (VEs) is increasing rapidly, and people are demanding easier and more credible ways to interact with these new sites. We define VEs as a special kind of 3D virtual environment, inhabited by avatars which represent humans in the VE, or even autonomous agents. This kind of software was selected because of its increasing importance as the new future trend in interactive software applications. From a software engineering point of view, VEs can be seen as a special kind of information system, so they must be analyzed, designed, and implemented in this respect.

Our aim is to improve software engineering's traditional software processes to achieve quality VEs. In this chapter, we present a framework called SENDA, which defines a formal process model to develop VEs.

Introduction

With the increase of computer networks, and especially Internet, people have felt attracted to applications like CHATs, MUDs (multi-user dungeons), and social VEs (virtual environments). These are different generations of applications where the main idea is not only interacting with the system, but also interacting with other users connected through these networks in different parts of the world.

Today, virtual environments are being used in many fields: social, finance, commerce, banking, information system sciences, communication, CSCWs (computer supported collaborative worlds), education, entertainment and leisure, medicine, architecture, and geography (CALT, 2000). This kind of application also seems to be the future of interactive programming (Berenguer, 1997) and can be used especially to demonstrate situations at risk.

We are going to focus on the most recent VEs based on 3D graphics and inhabited by Avatars and autonomous agents. These types of applications are called VEs, the acronym for *virtual environments*. They are also referred to as *multi-user virtual worlds* (Damer, 1997), but in essence, they are the same.

In the earlier VEs the following technological problems were solved:

- Multi-user communication
- Graphic representation
- Real-time communication

Much of the research done in the *inhabited virtual environments* field has focused on computer graphics rendering technologies and communication protocols.

Nowadays, a large number of VEs' technical problems have been solved. Therefore, our next goal is to provide these VEs with enough support to develop these environments. However, it is difficult to find reports on the process that must be followed to develop VEs. This may be due to insufficient experience in this field. We can say that VE development methods and processes are in their infancy. At the moment, the development of VEs is not following a mature process, so it is necessary to provide this development with Software Engineering Paradigms, Principles, and Procedures.

However, technology is not the most important issue in VEs. Even in the first VE called *Habitat*, which was bi-dimensional, some interesting conclusions were reached.

> *"The essential lesson that we have learnt from our experience with Habitat, is that cyberspace is defined more by the interaction among the players within it than by the technology with which it is implemented."* (Morningstar & Farmer, 1990)

Nowadays, the implementation process of VEs is well known but informal. In fact, good and useful results can sometimes be achieved with a modest outlay of hardware and resources. The problem comes from the very expensive constructions (Venus, 1999) derived from following the informal process.

Therefore, the need for a more formal process is evident. This chapter presents the formal approach to VE development under the SENDA framework, developed to improve the quality of VE developments. In "Background" we present the up-to-date approaches to VEs' formal development, the weaknesses of the traditional software engineering discipline to develop VEs, and how to improve the existing deficiencies. The section "SENDA: Development Framework Proposed," describes SENDA framework, and the remainder of the chapter describes conclusions and future lines of work respectively.

Background

Since the identification of the "Software Crisis" in the 1960s, many institutions have dedicated their efforts to defining standards, process models, and so forth formally for software development (Moore, 1998).

The Software Engineering Research Community is not the only one interested in this area. The need to define new techniques inspired by the Software Engineering discipline is widely known to scientific bodies related to HCI (human computer interaction). (Brown, 1999). Outside the software engineering discipline, some researchers like Fencott (1999) and Kaur (1998) from the HCI field have already dealt with the problem of developing VEs from a usability of software point of view.

Within the software engineering discipline, Larijani (1994) said that the object-oriented paradigm was the one which best fit VEs development.

From the study of the object-oriented methodologies used to develop VEs, we can conclude that:

1. A great number of deficiencies are not strictly related to the object-oriented (OO) methodologies. As methodologies are instances of process models, these are deficiencies of process models and not of development methodologies.
2. The processes with deficiencies are: estimation, analysis, design, implementation, scheduling, and verification.

In order to correct the process model deficiencies, ISO 12207 (ISO, 1995) and IEEE 1074 (IEEE, 1991) process models, pillars of Software Engineering, were tailored to VE development. It must be noted that the modifications of these processes are valid for both structured and OO paradigms. The undefined activities of VE development were proposed and integrated in the OO methodologies.

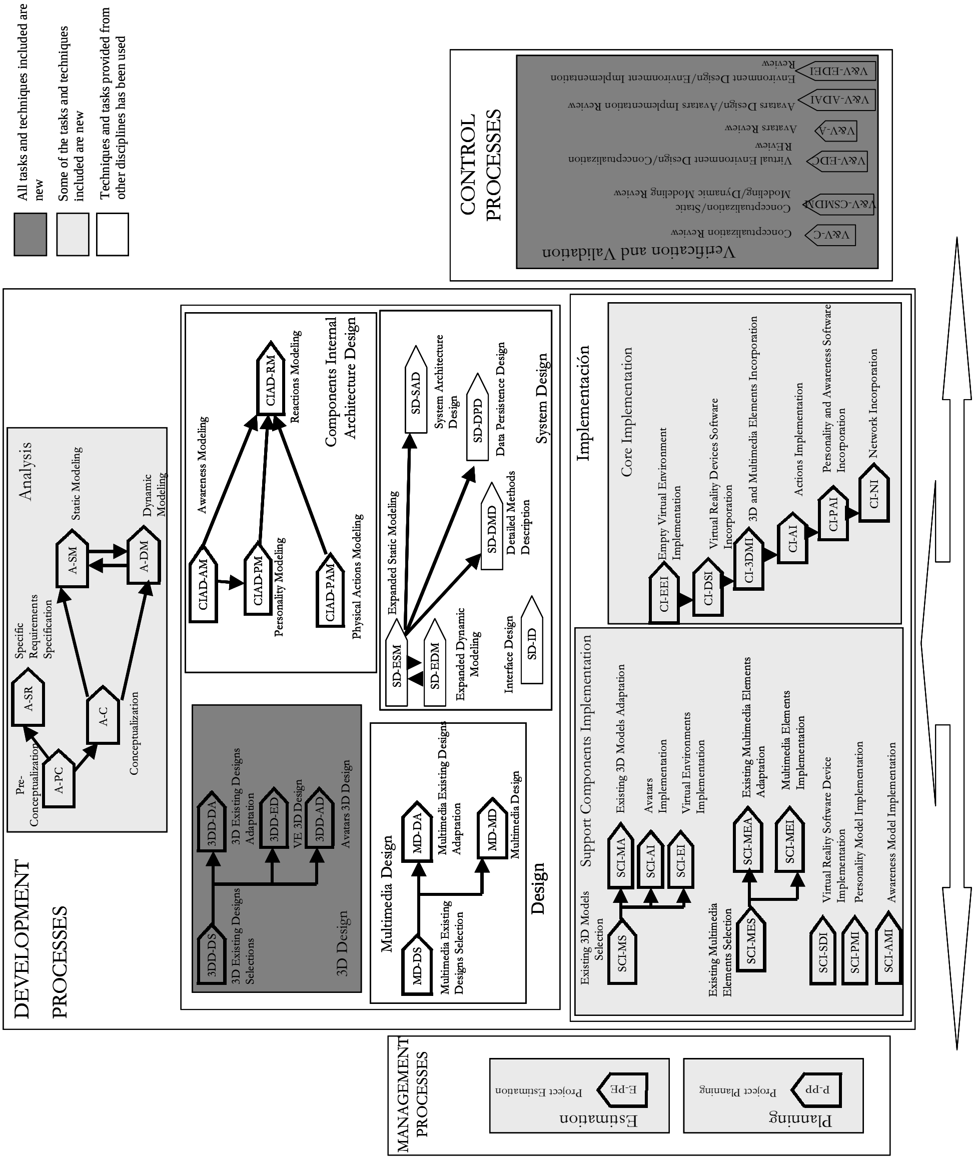
Although all the processes can be used as they are defined in the process model, we have chosen only those that required special treatment or new techniques to build a VE system.

SENDA improves traditional software engineering process models by providing new processes and techniques, improving existing processes and techniques, and using techniques provided by different disciplines. SENDA is described in detail in Sanchez-Segura (2001) and comprises 10 processes and 36 tasks as seen in Figure 1. Each task is described through its input products, corresponding outputs, techniques, and the participants.

SENDA: Development Framework Proposed

Although the SENDA framework specifies processes and tasks that cover the whole development lifecycle, in this chapter we focus on a short description of the analysis, design, and implementation processes and all their tasks and interrelationships. The traditional VE *design* process was divided into four processes, namely, *3D design, multimedia elements design, components internal architecture design,* and *system design,* and the implementation process into two: *components support implementation* and *core implementation* (see Figure 1). The reason for this splitting is the relationship among the tasks located in each process. Management processes proposed in SENDA are

Figure 1. SENDA processes and tasks


All tasks and techniques included are new
Some of the tasks and techniques included are new
Techniques and tasks provided from other disciplines has been used
CONTROL PROCESSES
Verification and Validation
V&V-C Conceptualization Review
V&V-CSMDM Conceptualization/Static Modeling/Dynamic Modeling Review
V&V-EDC Virtual Environment Design/Conceptualization REview
V&V-A Avatars Review
V&V-ADAI Avatars Design/Avatars Implementation Review
V&V-EDEI Environment Design/Environment Implementation Review
DEVELOPMENT PROCESSES
Analysis
A-PC Pre-Conceptualization
A-SR Specific Requirements Specification
A-C Conceptualization
A-SM Static Modeling
A-DM Dynamic Modeling
Design
Components Internal Architecture Design
CIAD-AM Awareness Modeling
CIAD-PM Personality Modeling
CIAD-PAM Physical Actions Modeling
CIAD-RM Reactions Modeling
System Design
SD-ESM Expanded Static Modeling
SD-EDM Expanded Dynamic Modeling
SD-SAD System Architecture Design
SD-DPD Data Persistence Design
SD-DMD Detailed Methods Description
SD-ID Interface Design
3D Design
3DD-DS 3D Existing Designs Selections
3DD-DA 3D Existing Designs Adaptation
3DD-ED VE 3D Design
3DD-AD Avatars 3D Design
Multimedia Design
MD-DS Multimedia Existing Designs Selection
MD-DA Multimedia Existing Designs Adaptation
MD-MD Multimedia Design
Implementación
Core Implementation
CI-EEI Empty Virtual Environment Implementation
CI-DSI Virtual Reality Devices Software Incorporation
CI-3DMI 3D and Multimedia Elements Incorporation
CI-AI Actions Implementation
CI-PAI Personality and Awareness Software Incorporation
CI-NI Network Incorporation
Support Components Implementation
SCI-MS Existing 3D Models Selection
SCI-MA Existing 3D Models Adaptation
SCI-AI Avatars Implementation
SCI-EI Virtual Environments Implementation
SCI-MES Existing Multimedia Elements Selection
SCI-MEA Existing Multimedia Elements Adaptation
SCI-MEI Multimedia Elements Implementation
SCI-SDI Virtual Reality Software Device Implementation
SCI-PMI Personality Model Implementation
SCI-AMI Awareness Model Implementation
MANAGEMENT PROCESSES
Estimation
E-PE Project Estimation
Planning
P-PP Project Planning

not the focus of this chapter, but these processes are described in greater detail in Sánchez-Segura (2004).

The symbol notation used to represent tasks and the relationship among them can be found in Kruchten (1999) and "*Process Acronym plus Task Acronym*" has been used to name tasks.

The acronyms of the processes we describe in this chapter are:

- **A:** Analysis Process
- **3DD:** 3D Design Process
- **MD:** Multimedia Design Process
- **SD:** System Design Process
- **CIAD:** Components Internal Architecture Design Process
- **SCI:** Support Components Implementation Process
- **CI:** Core Implementation Process

Analysis Process

The analysis process is one of the traditional processes that has been improved, providing some new tasks and techniques. A summary of the analysis process appears in Table 1; proposed elements appear in italics.

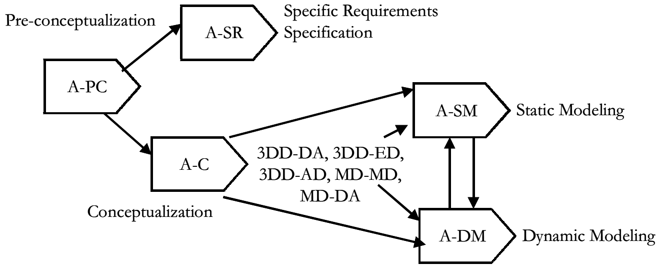
Many researchers suggest that the analysis phase must take into account a requirements specification task, which must gather only the system features and not how the system performs (Davis, 1993). Like Sommerville (1997), we think that although this idea is very attractive, it is not very useful in practice.

The first task to carry out in the analysis process is *pre-conceptualization*, which allows the identification of the set of tasks to be developed. To achieve this, Questionnaire 1 must be completed, as the answers in this questionnaire allow the project manager to know the SENDA tasks to be developed.

Once the pre-conceptualization task is finished, the *conceptualization* task must be developed in order to obtain the "conceptualization document" that contains "use cases" and "use concepts." Use cases are taken from the Unified Modeling Language standard. We propose a new term, **"use concept,"** not yet defined, as a tool to describe the system functionalities not triggered by an external author. Each use concept is defined by a brief description of the functionality, which will not be demanded directly by the user, and the following three fields:

- **Purpose:** Use concept's main goal.

Table 1. Analysis process



TASK NAME	TASK ACRONYM	TASK DESCRIPTION		
Pre-Conceptualization	A-PC	Artifacts	Input	Contract definition
			Output	Selected tasks to be developed according to the kind of VEs to be developed
		Techniques		Interviews with the clients
		Participants		System analyst
Specific Requirements Specification	A-SR	Artifacts	Input	Problem definition
			Output	*Specific requirements document*
		Techniques		Alternatives study, interviews
		Participants		System analyst, client, user
Conceptualization	A-C	Artifacts	Input	Selected tasks to be developed, output of A-PC
			Output	Problem definition, acronyms, abbreviations, initial list of requirements, use cases and use concepts classified
		Techniques		Use cases, *use concepts*
		Participants		System analyst, client, user
Static Modeling	A-SM	Artifacts	Input	Conceptualization document, outputs of 3D Design tasks, outputs of multimedia design process
			Output	Classes model; amplified classification of use cases and use concepts table
		Techniques		Structural diagram
		Participants		System analyst
Dynamic Modeling	A-DM	Artifacts	Input	Conceptualization document, classes model
			Output	Dynamic model
		Techniques		UML sequence diagrams, scenarios, operational contracts
		Participants		System analyst

- **Working Mode:** How the use concept is going to be used.
- **Dynamic:** The use frequency.

Table 2 illustrates use concept, which represents the functionality to prevent the avatar from colliding with an obstacle in its path.

Table 2. Use concept example

Use concept name: The avatar must not collide with the walls. **Use concept code:** Concept(7)	**Purpose:** To prevent the avatar from going through the walls so the environment is more credible. **Working mode:** When an avatar arrives at a wall, it is not allowed to go through the wall and the avatar must stop. **Dynamic:** Each time the avatar is near a wall.

Questionnaire 1. Pre-conceptualization questionnaire

Is VE only for guided tours without any type of interaction?	**Yes** **No** ☐ ☐ If Yes, ignore the Internal Characteristics of the VE components in the Design process; the ICS-IMCI and ICS-IMP tasks of the Implementation of Components and Support process; and the IMP-IMCL task of the Principal Module Implementation process.
Will VE be networked?	**Yes** **No** ☐ ☐ If No, delete the IMP-ISRE task of the Implementation of the Principal Module process.
Will VE use virtual reality mechanisms?	**Yes** **No** ☐ ☐ If No, delete the ICS-SDRV task in the Implementation of Support Components process and the IMP-ISRV task in the Implementation of the Principal Module process.
Will VE be used for teaching?	**Yes** **No** ☐ ☐ If Yes, a tutor module should be considered in the general architecture of VE.
Will VE be used to develop social relations?	**Yes** **No** ☐ ☐ If No, the DAI-SMCI task should be deleted from the Internal Characteristics of Components Design process of the VE. If Yes, the need to include a personality module or a social module in the VE should be considered.
Will the VE have 3D elements?	**Yes** **No** ☐ ☐ If No, the 3D Design and the ICS-S3D, ICS- AR3D, ICS-IA3D, ICS-IVE tasks of the Implementation of Support Components, and the IMP-IO3D tasks can be deleted. Remember that the part corresponding to loading the 3D elements of the VE should not be executed.
Will the VE have multimedia elements?	**Yes** **No** ☐ ☐ If No, the Multimedia Elements Design, and the ICS-SEM, ICS-AREM, ICS-IEM tasks of the Implementation of Support Components can be eliminated. Remember that the part corresponding to the insertion of multimedia elements of the IMP-IO3D task should not be executed.
Will the VE have avatars guided by agents?	**Yes** **No** ☐ ☐ If Yes, the avatars should be modeled to be controlled by agents, that is, they should be automatically controlled by an interface within the system. Therefore, the formalism of Use Concepts to define some of the requirements of the Conceptualization task should be used.
Will the VE control the personality model of the avatar partially or totally?	**Yes** **No** ☐ ☐ If No, the DAI-SMCI task of the architecture of the Internal Components Design process, and the ICS-IMCI task of the Implementation of Support Components process can be deleted.
Will the VE partially or totally control the reasoning model of the avatar?	**Yes** **No** ☐ ☐ If No, the DAI-DMR task of the architecture of the Internal Components Design process can be deleted.
Will the VE totally or partially control the model perception of the avatar?	**Yes** **No** ☐ ☐ If No, the DAI-IMP task of the architecture of the Internal Components Design process architecture of the Internal Components Design process and the ICS-IMP task of the Implementation of Support Components should be deleted.

As VEs are constantly evolving, the type of virtual reality mechanism, the development software, the hardware, and so forth had to be chosen as soon as possible in order to test the compatibility of these elements which influence the rest of the development processes. We proposed the *specific requirements* task to list the VEs' specific decisions that were taken regarding virtual reality devices, compatibility between software and devices, and so forth.

We propose a set of categories to classify use cases and use concepts according to their special characteristics such as perception, reasoning, animation, and visualization. So, each one of these categories could be dealt with in the components internal architecture design process. These categories are: connection to the VE, virtual reality devices interface, animation, perception, VE evolution, reasoning and decision, communication with other connected users, and scene visualization. Using this classification, it is easy to trace the requirements into use concepts and use cases and, for instance, if a category is empty, the analyst can ask the client to verify that the category is really empty, and if the requirements were not properly extracted, to take this opportunity to define them.

Static and dynamic modeling have been taken from object-oriented methodologies, relating them to the rest of SENDA tasks.

Design Processes

Due to the features of VEs developments, the traditionally known "Design Process" has been subdivided into four processes: 3D design process, multimedia design process, components internal architecture design process, and design process. In the following subsections these processes are presented in detail. In design processes, two main kinds of roles are involved: system and graphic designers.

- **System Designer:** Typically assigned to define "how" the application is. By this, we mean the person who defines the control of the system following the system analyst's definition of "what." In VEs, the system designer is also the person who guides the graphic designer because of his or her knowledge of the application to be developed. The system designer must also have a basic knowledge of graphic design.
- **Graphic Designer:** His or her job in the design process is feedback, view maps, environment modeling forms, and avatar modeling forms for the system designer. After the feedback stage, graphic designers can begin the implementation task.

3D Design Process

There is a set of requirements associated with VEs which are not usually described as they are not related to the functionality of the system. The *3D design process*, a new process SENDA proposes, takes these requirements into account and is critical:

1. To describe VE aesthetic details.
2. To facilitate a common language which allows the system and model designers to understand each other. In these cases, a natural language is lacking due to the different backgrounds of the system designer (computer expert) and the model designer (graphic arts expert). As a result, it is necessary to translate the system requirements into comprehensible specifications for the model designer. For example, performance in real time means nothing to the model designer, but provides a lot of information for the system designer.
3. To ease and minimize time consumed in the support components implementation process, one of the implementation processes.

This process is classified as design processes because the information extracted from the proposed tasks describes in detail how the environment is and not what the VE does. A summary of the 3D design process appears in Table 3.

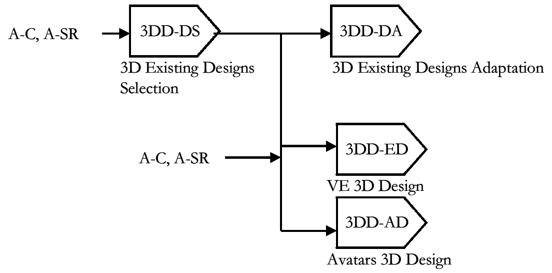
For the 3D design process, SENDA proposes the following techniques:

- **Two Forms:** An environment form that describes the VE, and a special components form that describes the components in more detail. These forms have to be completed by the system designer and validated by the model designer.
- **View Maps** to facilitate the spatial location of the VE components.
- The **Hierarchical Structure of Avatars and VE Components**. The hierarchical structure allows the design of avatars or components in any form, whereas standards (Roehl, 1998) only allow the avatars to be described in human form.
- A **Navigational Diagram** represents the links between different logical spaces within the VE.

For details and examples of these techniques (see Sánchez-Segura, 2003). We describe briefly the goals of the tasks included in this process:

- **3D Existing Designs Selection:** This task is defined for reuse in the design process. 3D designs developed in previous projects can be reused.

Table 3. 3D design process



TASK NAME	TASK ACRONYM	TASK DESCRIPTION		
3D Existing Designs Selection	3DD-DS	Artifacts	Input	Conceptualization document, specific requirements document from analysis; existing 3D designs
			Output	Selected existing 3D designs
		Techniques		Evaluation of previous 3D designs
		Participants		System designer
3D Existing Designs Adaptation	3DD-DA	Artifacts	Input	Conceptualization document, specific requirements document from analysis; existing 3D designs; selected existing 3D designs
			Output	Selected 3D designs
		Techniques		3D design adaptation
		Participants		System designer
VE 3D Design	3DD-ED	Artifacts	Input	All outputs from analysis that can give aesthetic VE details
			Output	VE modeling forms, view maps, behavior tables, navigational tables, elements structural hierarchies, table of elements structural hierarchies, table of description of articulations
		Techniques		View maps, environment forms, navigational diagram
		Participants		Graphic designers, system designer, client (to provide VE aesthetic details)
Avatars 3D Design	3DD-ED	Artifacts	Input	All outputs from analysis that can give aesthetic VE details
			Output	Hierarchical structure of avatars and VE components
		Techniques		Avatars and components hierarchy
		Participants		Graphic designers, system designer, client (to provide avatars aesthetic details)

For instance, if we design a child's room with the 3D design process proposed in this chapter, it is possible to reuse this design in other virtual environments where this room must be designed.

- **3D Existing Designs Adaptation:** 3D designs selected in the above task must be analyzed to check if these designs have to be adapted to satisfy the specifications identified for the current project. For instance, the above room may need another door, so the 3D design for that room must be adapted.
- **VE 3D Design:** This task includes the definition of a set of virtual spaces and the objects to be included in them. We will focus on this task in detail.

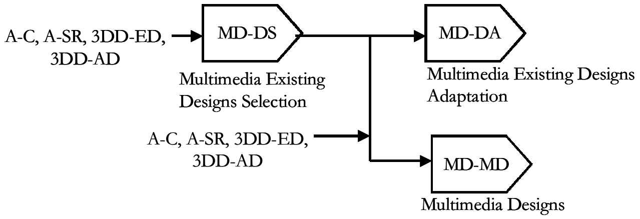
- **Avatars 3D Design:** This task includes the definition of a set of virtual inhabitants, their appearance, and physical structure.

Multimedia Design Process

This process is classified as design process because the information extracted through the proposed tasks describes in detail *how* the multimedia elements—sound, images, animations—are and not *what* they do. A summary of the multimedia design process appears in Table 4.

- **Multimedia Existing Designs Selection:** In this task, previous multimedia designs must be selected for the current project.
- **Multimedia Existing Designs Adaptation:** Selected multimedia designs must be analyzed to identify necessary adaptations or modifications.
- **Multimedia Design:** This task includes the definition of a set of tools, such as storyboards, to describe the multimedia elements. The techniques used

Table 4. Multimedia design process


A-C, A-SR, 3DD-ED, 3DD-AD
MD-DS
Multimedia Existing Designs Selection
MD-DA
Multimedia Existing Designs Adaptation
A-C, A-SR, 3DD-ED, 3DD-AD
MD-MD
Multimedia Designs

TASK NAME	TASK ACRONYM	TASK DESCRIPTION		
Multimedia Existing Designs Selection	MD-DS	Artifacts	Input	Conceptualization and specific requirements documents; both are analysis process outputs; existing multimedia designs; 3D environment elements specific forms; avatars forms
			Output	Existing multimedia selected designs
		Techniques		Evaluation of existing multimedia designs
		Participants		System designer
Multimedia Existing Designs Adaptation	MD-DA	Artifacts	Input	Selected multimedia existing designs; 3D environment elements specific forms; avatars forms
			Output	Updated existing multimedia designs
		Techniques		Adaptation of existing multimedia designs
		Participants		System designer
Multimedia Design	MD-MD	Artifacts	Input	Multimedia selected elements updated
			Output	Multimedia elements description
		Techniques		Storyboard or any other multimedia existing methods
		Participants		System designer

in this task are not indicated in this chapter because they are well defined in the multimedia field.

Components Internal Architecture Design Process

This is a new process SENDA proposes to deal with the categorization of use cases and concepts in the analysis process. To be exact, the especially critical categories are:

- **Perception**
- **Internal characteristics:** personality model, mood, social models, and so forth
- **Reasoning**
- **Reaction:** this may imply a simple modification of a variable or the representation of a very complex VE scene.

This process manages the above-mentioned categories that must be designed for each class. For instance, detection does not have to be designed for a particular class if there are no perception mechanisms.

A summary of the components internal architecture design process appears in Table 5.

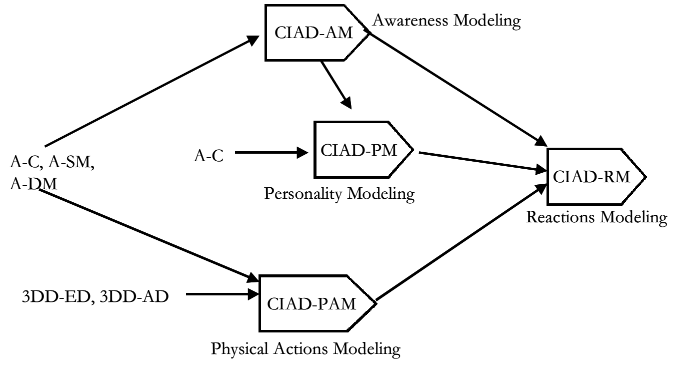
The aim of this process is to define the actions that can take place within the VE. Many people are involved in this process: psychologists, sociologists, and so forth, because a multi-disciplinary work is necessary to provide this kind of application with sufficiently interesting interactive features to give credibility to avatars and the rest of the VE elements.

It is very important to emphasize the relation between "components internal architecture design" and "3D design" processes. They must be coherent.

"Conceptualization" task A-C specifies every action to be done by avatars and the rest of the elements within the VE. These actions include the detection of the events that occur in the environments, how avatars feel these events, and how these feelings are shown through the physical representation of elements. These are all defined by the set of tasks included in this process:

- **Awareness Modeling:** In this task the way avatars and agents can perceive the rest of the inhabitants and objects in the VE is defined so that they can react to the stimulus coming from their surroundings. Specific techniques to achieve this task can be found in Chapter 7.
- **Physical Actions Modeling:** The activities that the avatars and agents must be able to perform in the environment must be designed with respect

Table 5. Components internal architecture design process



TASK NAME	TASK ACRONYM	TASK DESCRIPTION		
Awareness Modeling	CIAD-AM	Artifacts	Input	Conceptualization document from analysis process; expanded use cases and concepts classification from analysis process
			Output	Detection methods description
		Techniques		No specific technique is proposed
		Participants		System and graphic designer
Personality Modeling	CIAD-PM	Artifacts	Input	Conceptualization document from analysis process; outputs from CIAD-AM task
			Output	Description of internal features; internal model definition if needed
		Techniques		No specific technique is proposed
		Participants		System designer
Physical Actions Modeling	CIAD-PAM	Artifacts	Input	Conceptualization document from analysis process; expanded use cases and concepts classification from analysis process; outputs from the VE 3D design process
			Output	Elements position interpretation table; avatar position interpretation table; animations table
		Techniques		*VE physical animations description:* ➢ *VE elements position interpretation table* ➢ *Avatar position interpretation table* ➢ *Animations table*
		Participants		System and graphic designer
Reactions Modeling	CIAD-RM	Artifacts	Input	Outputs from CIAD.AM, CIAD-PM, and CIAD-PAM
			Output	Decision and reasoning rules definition
		Techniques		*Definition of the reasoning model*
		Participants		System analyst and system designer

to the avatar's structure defined in the *avatars modeling* task. For instance, Table 6 describes the action through which the visitor gets an object.

- **Personality Modeling:** It is important to endow avatars and agents with a personality and emotion model, and to relate the actions defined above with the emotions. Specific techniques to achieve this task will be discussed in Chapter 7.

Table 6. Example of action description

Element code: visitor			
Animation	**Variant**	**Involved Elements**	**Position**
Get an object	There are no variants	Head	Looking ahead
		Shoulders	Lift
		Body	Lift
		Arms	The left arm stays at the visitor's side while the right arm reaches out to be near the object to be cached.
		Legs	Immobile

Table 7. Example of decisions rules design

Rule number	**Description**
Rule 1	If the avatar has just entered the VE, then it must go to the radiation protection counter, bring its target, and get the dosimeter.
Rule 3	If the avatar is in the changing room with work clothes on, then the avatar must go to the card reader and enter though the turnstile.

- **Reactions Modeling:** This task defines the way in which VE elements are able to react to reason and make decisions. For instance, in Table 7 there are some rules related to the performance of a system in nuclear plants to train people to use some elements in their daily work.

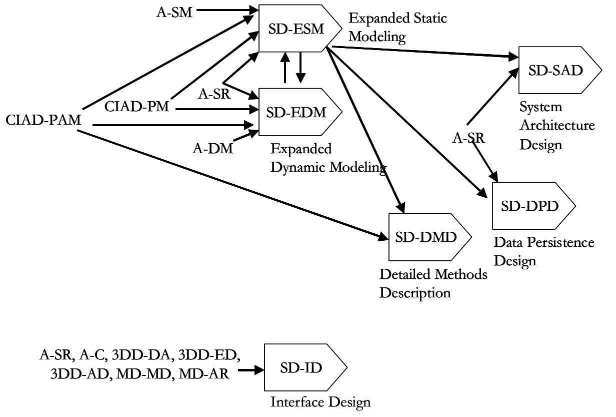
System Design Process

As we have mentioned before, the tasks included in this process are well defined in most object-oriented methodologies. A summary of the system design process appears in Table 8.

Expanded static modeling (SD-ESM) and *expanded dynamic modeling (SD-EDM)* tasks take:

- Class diagram from *"static modeling" (A-SM)* and transition diagrams and event traces from *"dynamic modeling" (A-DM)* task (both from the analysis process).
- *"Physical actions modeling"* task (CIAD-PAM) to create new classes and methods derived from the exact definition of movements, and so forth, which is the output of CIAD-PAM.

Table 8. System design process



TASK NAME	TASK ACRONYM	TASK DESCRIPTION		
Expanded Static Modeling	SD-ESM	Artifacts	Input	Static modeling task outputs; physical actions modeling task outputs; personality modeling task outputs; specific requirements specification task outputs
			Output	Design classes model
		Techniques		Entity-relationship model
		Participants		System designer
Expanded Dynamic Modeling	SD-EDM	Artifacts	Input	Dynamic modeling task outputs physical actions modeling task outputs; personality modeling task outputs; specific requirements specification task outputs
			Output	Design dynamic models
		Techniques		Interaction diagrams, states diagrams
		Participants		System designer
Detailed Methods Description	SD-DMD	Artifacts	Input	Physical actions modeling task outputs; expanded static modeling task outputs
			Output	Methods description in pseudo code
		Techniques		Pseudo-code
		Participants		System designer
System Architecture Design	SD-SAD	Artifacts	Input	Specific requirements specification task outputs; expanded static modeling task outputs
			Output	Components, deployment, and packages models
		Techniques		Components, deployment, and packages models
		Participants		System designer
Data Persistence Design	SD-DPD	Artifacts	Input	Specific requirements specification task outputs; expanded static modeling task outputs
			Output	Database design
		Techniques		Entity-relationship model
		Participants		System designer
Interface Design	SD-ID	Artifacts	Input	Conceptualization document
			Output	Interface design
		Techniques		No specific technique is proposed
		Participants		System designer, client

The static model developed in the analysis process must be expanded in the *expanded static model task*, and the dynamic model developed in the analysis process must be expanded in the *expanded dynamic model task*.

Figure 2. Pseudo-code corresponding to the catch action

```
Catch( )
 ArmR=Instance from class ARM
           ArmR.Move(Get)
 AVATAR.Get_element(PRVIR OBJECT)
 VE.Remove_element(PRVIR OBJECT)
END Cath
```

Figure 3. Pseudo-code corresponding to Rules 1 and 3

```
RULE 1
IF "Avatar has just arrived in the Virtual Environment" THEN
-Go to the radiation protection counter
-Get the dosimeter

RULE 3
IF "avatar is in the changing room with clothes to work on" THEN
go to the card reader and the enter though the turnstile
```

System architecture design (SD-SAD) task and *data persistence design (SD-DPD)* tasks take the "specific requirements document," returned from A-SR, where restrictions or details to be used in SD-SAD and SD-DPD are provided. In the system architecture design task, the classes, from static model, are packaged. Deployment and component diagrams, from UML, must also be developed. In Chapter 8 some guidelines to define specific architectures for VEs can be found. The data persistence design task defines the way VE information is managed.

In the *interface design task*, a prototype of the user interface must be developed.

The actions identified in the rest of design processes are detailed in pseudo-code in detailed methods descriptions tasks. For instance, pseudo-code in Figure 2 corresponds to the action designed in Table 6. And pseudo-code in Figure 3 corresponds to Rules 1 and 3 designed in Table 7.

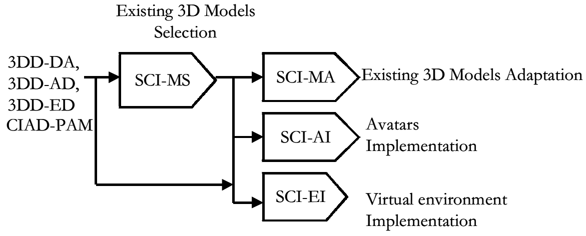
Implementation Processes

The traditional implementation process has been split into two processes: support components implementation process (SCI) and core implementation process (CI). The core implementation process proposes an incremental development of the VE by first creating an empty VE and adding some functionalities to each task. On the other hand, the tasks under the support components implementation process can be developed along with the rest of processes, at specific points. We explain both of them.

Support Components Implementation Process

VEs have many modules or pieces. The *support components implementation* process was proposed to build the VE modules independently. This is proposed in the *core implementation* process, which is incremental, and allows the progressive addition of small modules. Adjustments could thus be made to any module of the system without having to modify others. Examples of modules are the interface with the virtual reality device, 3D models, and multimedia elements.

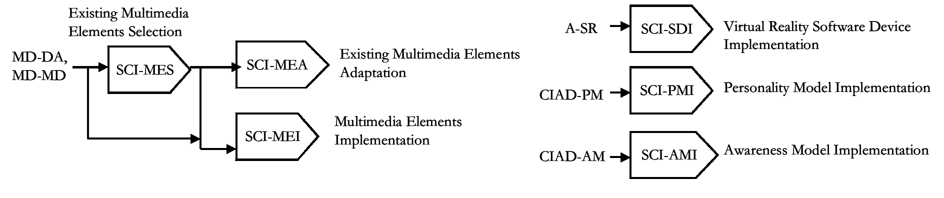
Table 9. Support components implementation process (Part I)


Existing 3D Models Selection
3DD-DA,
3DD-AD,
3DD-ED
CIAD-PAM
SCI-MS
SCI-MA
Existing 3D Models Adaptation
SCI-AI
Avatars Implementation
SCI-EI
Virtual environment Implementation

TASK NAME	**TASK ACRONYM**	**TASK DESCRIPTION**		
Existing 3D Models Selection	SCI-MS	Artifacts	Input	3D design tasks outputs; animations description table, output from physical actions modeling task
			Output	Existing selected 3D models
		Techniques		The ones preferred by the graphic designer
		Participants		Graphic designer
Existing 3D Models Adaptation	SCI-MA	Artifacts	Input	3DD-DA task outputs; selected models in task SCI-MS; avatars position interpretation table, output from physical actions modeling task
			Output	Graphics files corresponding to the adapted implementation of the 3D avatars, and VE; 3D exported models table; 3D exported avatars table
		Techniques		The ones preferred by the graphic designer
		Participants		Graphic designer
Avatars Implementation	SCI-AI	Artifacts	Input	3DD-AD task outputs; selected models in task SCI-MS; avatars position interpretation table, output from CIAD-PAM task
			Output	Graphics files corresponding to the implementation of the 3D avatars; 3D avatar exported elements table
		Techniques		The ones preferred by the graphic designer
		Participants		Graphic designer
Virtual Environments Implementation	SCI-EI	Artifacts	Input	Outputs from task 3DD-ED Outputs from task CIAD-PAM
			Output	Graphics files corresponding to the implementation of the 3D VE elements; 3D exported models table
		Techniques		The ones preferred by the graphic designer
		Participants		Graphic designer

Table 11. Core implementation process

Table 10. Support components implementation process (Part II)


Existing Multimedia Elements Selection
MD-DA, MD-MD
SCI-MES
SCI-MEA
Existing Multimedia Elements Adaptation
SCI-MEI
Multimedia Elements Implementation
A-SR
SCI-SDI
Virtual Reality Software Device Implementation
CIAD-PM
SCI-PMI
Personality Model Implementation
CIAD-AM
SCI-AMI
Awareness Model Implementation

TASK NAME	**TASK ACRONYM**	**TASK DESCRIPTION**		
Existing Multimedia Elements Selection	SCI-MES	Artifacts	Input	All outputs from MD-MD task
			Output	Existing multimedia elements
		Techniques		The ones selected by the multimedia expert
		Participants		Multimedia expert
Existing Multimedia Elements Adaptation	SCI-MEA	Artifacts	Input	All outputs from MD-MD task; selected models in SCI-MES
			Output	Sound, video, image, files corresponding with the updated multimedia files; multimedia files table
		Techniques		The ones selected by the multimedia expert
		Participants		Multimedia expert
Multimedia Elements Implementation	SCI-MEI	Artifacts	Input	All outputs from MD-MD task
			Output	Sound, video, image, files corresponding with the updated multimedia files; multimedia files table
		Techniques		The ones selected by the multimedia expert
		Participants		Multimedia expert
Virtual Reality Software Device Implementation	SCI-SDI	Artifacts	Input	Specific requirements document from A-SR task
			Output	Virtual reality device software with its corresponding interface
		Techniques		Implementation techniques
		Participants		Programmer/s; virtual reality device provider
Personality Model Implementation	SCI-PMI	Artifacts	Input	Output from CIAD-PM task
			Output	Personality traits software
		Techniques		Implementation techniques
		Participants		Programmer/s
Awareness Model Implementation	SCI-AMI	Artifacts	Input	Output from CIAD-AM task
			Output	Awareness model software
		Techniques		Implementation techniques
		Participants		Programmer/s

As soon as these modules were completed , they were added to the real system for the client to see the final VE and decide if the system met their expectations.

In this process, all the tasks related to implementation, but strongly dependent on VE features, are included. As a result, all the models designed in the *components* *internal architecture design process* must be developed here. A summary of the support components implementation process appears in Tables 6 and 7.

Core Implementation Process

This process evolves from an empty to a completed VE and includes a different kind of component in each step. A summary of the core implementation process appears in Table 11.

The *"3D and Multimedia Elements Incorporation" (CI-3DMI)* task is not always easy. For example, if the tool selected to develop the VE is based on graphic libraries, for instance, WorldToolKit™ (Sense8), DirectX™ (Microsoft), there are three main difficulties:

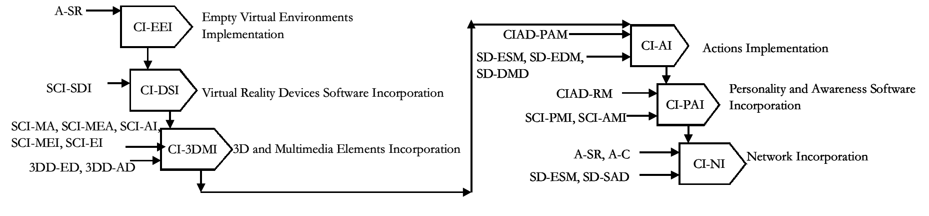
- All avatar mobile pieces must be loaded independently and then the full avatar structure must be reconstructed. This reconstruction must follow the avatar modeling form guidelines designed in the *"Avatars 3D Design"* (3DD-AD) task. The same goes for any other element that must have mobile pieces. This means that *"VE 3D Design" (3DD-ED) task* products are needed.
- Avatars and the rest of the objects with actions associated in the SD-ESM, SD-EDM2, and SD-DMD tasks must be endowed with their respective methods.
- In general, all environment objects must be loaded several times because the programmer cannot see the VE aspect until the VE is compiled and executed (rendered).

To make the implementation process easier and to link up with the rest of development processes, we have defined and developed a tool that can extract the design data, composed of 3DD-ED and 3DD-AD products, stored in the "VE design database." This database stores the information described in 3D design process forms and 3D design process view maps.

By following the design guidelines stored in the "VE design database," this tool will allow:

- The programmer to specify objects location. By this, we mean the (x,y,z) position which is not always included in the design process.
- The reconstruction of the avatar's hierarchy.
- The automatic generation of the program lines to load objects in the VE and to render this one.

Table 11. Core implementation process


A-SR
CI-EEI
Empty Virtual Environments Implementation
SCI-SDI
CI-DSI
Virtual Reality Devices Software Incorporation
SCI-MA, SCI-MEA, SCI-AI, SCI-MEI, SCI-EI
3DD-ED, 3DD-AD
CI-3DMI
3D and Multimedia Elements Incorporation
CIAD-PAM
SD-ESM, SD-EDM, SD-DMD
CI-AI
Actions Implementation
CIAD-RM
SCI-PMI, SCI-AMI
CI-PAI
Personality and Awareness Software Incorporation
A-SR, A-C
SD-ESM, SD-SAD
CI-NI
Network Incorporation

TASK NAME	TASK ACRONYM	TASK DESCRIPTION		
Empty Virtual Environment Implementation	CM-EEI	Artifacts	Input	Specific requirements document, output from A-SR task
			Output	Minimum code to represent an empty VE; this will be Version 1
		Techniques		Implementation techniques
		Participants		Programmer/s
Virtual Reality Devices Software Incorporation	CM-DSI	Artifacts	Input	Output from SCI-SDI task
			Output	VE Version 2—improves Version 1 by adding the communication with the virtual reality device code previously developed
		Techniques		Implementation techniques
		Participants		Programmer/s
3D and Multimedia Elements Incorporation	CM-3DMI	Artifacts	Input	VE Version 2—outputs of 3DD-ED and 3DD-AD tasks; outputs of SCI-AI and SCI-EI tasks
			Output	VE Version 3—improves Version 2 by adding the necessary code to load and render all the elements of the VE
		Techniques		Implementation techniques
		Participants		Programmer/s
Actions Implementation	CM-AI	Artifacts	Input	VE Version 3—outputs from SD-DMD, SD-ESM, and SD-EDM tasks; output from CIAD-PAM task
			Output	VE Version 4—improves VE Version 3 by adding the code corresponding to all the actions to be performed by the VE elements
		Techniques		Implementation techniques
		Participants		Programmer/s
Personality and Awareness Software Incorporation	CM-PIA	Artifacts	Input	VE Version 4—outputs from SCI-PMI and SCI AMI tasks
			Output	VE Version 5—improves VE Version 4 by adding the software of personality model awareness model and reasoning model
		Techniques		Implementation techniques
		Participants		Programmer/s
Network Incorporation	CM-NI	Artifacts	Input	VE Version 5—outputs from SD-SAD and SD-EDM tasks; outputs from A-C and A-SR tasks
			Output	VE Version 6—the final version except this version must be validated with the client
		Techniques		Implementation techniques on Web as the ones considered by the systems expert
		Participants		Programmer/s; system expert

- The generation of a "VE implementation database" with all the implementation details added to the "VE design database."

There are many tools that allow programmers to build VEs visually (Caligary True Space, etc.), but our aim is to link all the development processes through a set of connected tools; this is not possible with the above-mentioned tools.

Conclusions

The whole SENDA framework has been used in some developments of VEs. From the results obtained it must be noted that the processes and techniques proposed are powerful and flexible enough to allow for the creation of different VEs, respecting the constraints of the application (to run in real time, etc.). A detailed explanation of the results obtained using SENDA in different projects can be seen in Sanchez-Segura (2003).

Therefore, although these techniques guide the graphic designer, they do not interfere with their artistic approach to the task. The techniques are independent of the application implementation.

The proposed techniques have also proved useful to verify and validate the graphic designer's job after 3D models are implemented. Proposed tools and mechanisms allow:

- communication between graphic and system designers;
- comparison between the designed and implemented 3D models;
- reuse of sub-VEs design, and even implementation between different projects though the database designs.

References

Berenguer, X. (1997). Writing interactive programs. *Magazine Formats.*

Brown, J., Encarnaçao, J., & Schniderman, B. (1999). Human-centered computing, online communities, and virtual environments. *IEEE Computer Graphics and Applications, 19*(6), 70-74.

CALT: The Center for Advanced Learning Technologies. INSEAD–Bd de Constance–F–77305 Fontainebleau Cedex, France. (2000). Retrieved from: *www.insead.fr/CALT/Encyclopedia/ComputerSciences/VR/vr.htm*

Damer, B. (1997). Interacting and designing in virtual worlds on the Internet. *Tutorial for CHI97.*

Davis, A., & Hsia, P. (1993). Status report: Requirements engineering. *IEEE Software, 10*(6), 75-79.

Fencott, C. (1999). Towards a design methodology for virtual environments. *Proceedings of the Workshop on User Centered Design and Implementation of Virtual Environments,* University of York, UK.

IEEE Std. 1074-1991. (1991). *IEEE standard for developing software life cycle processes.* New York: IEEE Computer Society.

ISO/IEC Standard 12207:1995. (1995). Software life cycle processes. Ginebra (Suiza): International Organization for Standardization.

Kaur, K. (1998). *Designing virtual environments for usability.* PhD Thesis, City University, London.

Kruchten, P. (1999). *The rational unified process. An introduction.* Addison-Wesley (Object Technology Series).

Larijani, L.C. (1994). *Realidad virtual.* McGraw-Hill.

Moore, J. (1998). *Software engineering standards: A user's road map.* Los Alamitos, CA: IEEE Computer Society.

Morningstar & Farmer, F.R. (1990). The lessons of Lucasfilm's habitat. *Proceedings of the First Annual International Conference on Cyberspace.* Cambridge, MA: MIT Press.

Roehl, B. (1998). Specification for a standard VRML humanoid. Retrieved from: *ece.uwaterloo.ca:80/~h-anim/newespec.html*

Sánchez-Segura, M., Cuadrado, J., Moreno, A., Amescua, A., de Antonio, A., & Marbán, O. (2004). Virtual reality systems estimation vs. traditional systems estimation, *Journal of Systems and Software.*

Sánchez-Segura, M.I. (2001). *Aproximación metodológica al desarrollo de entornos virtuales.* PhD Thesis, Technical University of Madrid, Spain.

Sánchez-Segura, M.I., Cuadrado, J.J., de Antonio, A., de Amescua, A., & García, L. (2003). Adapting traditional software processes to virtual environments development. *Software Practice and Experience, 33*(11), 1050-1080. Retrieved from: www3.interscience.wiley.com/cgi-bin/jhome/1752

Sommerville, I., & Sawyer, P. (1997). *Requirements engineering: A good practice guide.* New York: John Wiley & Sons.

Venus: Virtual Environments for You. (1999). Retrieved from: *leonardo.ucs.ed.ac.uk/venus/foryou/foryou.html*

Section II

Designing Virtual Environments

Chapter V

Steps Toward a Design Theory for Virtual Worlds

Joseph A. Goguen
University of California at San Diego, USA

Abstract

Virtual worlds, construed in a broad enough sense to include text-based systems, as well as video games, new media, augmented reality, and user interfaces of all kinds, are increasingly important in scientific research, entertainment, communication, commerce, and art. However, we lack scientific theories that can adequately support the design of such virtual worlds, even in simple cases. Semiotics would seem a natural source for such theories, but this field lacks the precision needed for engineering applications, and also fails to addresses interaction and social issues, both of which are crucial for applications to communication and collaboration. This chapter suggests an approach called algebraic semiotics to help solve these and related problems, by providing precise application-oriented basic concepts such as sign, representation, and representation quality, *and a calculus of representation that includes blending. This chapter also includes some theory for narrative and metaphor, and case studies on information visualization, proof presentation, humor, and user interaction.*

Introduction: Motivation, Difficulties, and Approaches

The term "virtual world" is used in many ways, but perhaps virtual worlds can be broadly characterized as the class of media experiences that provide some sense of immersion and closure. By *immersion*, which is sometimes also called *virtuality*, we mean a sense of being engaged with non-physically present entities through material mediation in the immediate real world, but not with the other aspects of the immediate real world, and by *closure* we mean that the virtual world gives an appearance of relative completeness, although it may of course be changing. A lecture, a conversation, a movie, a magazine, a formal paper, a video game, a user interface, can all be virtual worlds in this sense. A major factor in creating immersion and closure is the coherence of the world; of course, there are many other factors, relating for example to the situation, background, and interests of participants, but this chapter is focused on ways to achieve coherent representations.

Given the enormous cultural and economic importance of current media for communication, entertainment, and art, as well as the promise of new media, there would be many uses for scientific theories that could provide guidance for difficult tasks, such as the following:

- designing new media (e.g., virtual reality environments with haptics);
- creating new metaphors (e.g., beyond the desktop for PCs);
- making new hardware (such as wireless appliances) more usable;
- designing new genres (such as interactive poems); and
- supporting non-standard users (e.g., with disabilities).

Because virtual worlds are user interfaces in some broad sense, and because user interface design is a well-developed area of computer science (which is also known as human-computer interaction, or HCI, or sometimes CHI), this would seem a good place to look for appropriate theory. But most HCI results are either very precise but also highly specialized and therefore not very useful (e.g., Fitt's law), or else they are very general but of uncertain reliability and generality (e.g., protocol analysis, questionnaires, case studies, usability studies).

Another plausible place to seek a theory of virtual world design would be semiotics, a subject founded by Charles Sanders Peirce (1965) and Ferdinand de Saussure (1976) in the late nineteenth century. Peirce was an American logician concerned with problems of meaning and reference, who concluded that these are relational rather than denotational, and who also made an influential distinction among modes of reference as symbolic, indexical, or iconic. Saussure, a Swiss linguist, wanted to understand how features of language relate to meanings, and he emphasized binary features and denotational meaning. More recent thinkers like the French literary theorist Roland Barthes (1968) combined and extended these theories, creating a powerful language for cultural and media studies, which in various versions has been called semiotics, semiology, structuralism, and finally post-structuralism. Unfortunately, this tradition:

1. Does not have the mathematical precision needed to integrate well with engineering processes;
2. Does not consider representing signs in one system by signs in another, as is needed for the study and design of interfaces;
3. Has not addressed dynamic signs, which are necessary for the study and design of interaction;
4. Has not much considered social issues, such as arise in shared worlds;
5. Tends to ignore the situated, embodied aspects of sign use;
6. Tends towards a Platonistic view of signs, as actual existing abstract entities; and
7. Often considers only single (complex) signs (e.g., a novel or a film), rather than systems of signs.

Therefore semiotics needs to address some significant problems before it can meet all our needs. This chapter sketches how algebraic semiotics attempts to bridge this gap. The theory originated in an attempt to understand data from an early experimental study of multimedia learning (Goguen & Linde, 1984), and was later elaborated for applications to user interface design; more complete expositions appear in Goguen (1999a, 2003), and Goguen and Harrell (2003), though the theory is still evolving. Here we focus more on motivation and applications.

There are at least two perspectives that one might take towards the study of signs and representations: pragmatic and theoretical. The first is the perspective of a designer, who has a job to get done, often within constraints that include cost, time, and stylistic guidelines; we may also call this an engineering perspective, and it will generally involve negotiating trade-offs among various values and constraints. The second is the perspective of a scientist who seeks to understand principles of design, and is thus engaged in a process of constructing and testing theories. From the second perspective, it makes sense to describe semiotic theories in a detailed formal way, and to test hypotheses by doing calculations and experiments with users. But from the pragmatic perspective, it makes sense to formalize only where this adds value to the design process, for example in especially tricky cases, and even then, only to formalize to the minimum extent that will get the job done. Our experience is that one can often get considerable benefit from applying principles of algebraic semiotics, such as identifying and preserving key features of the source theory, without doing a great deal of formalization.

From either the pragmatic or theoretical perspective, one should seek to model semiotic theories as simply as possible, since this will simplify later tasks, whether they are engineering design or scientific theorizing and experimentation (not forgetting that the conceptual simplicity of a theory does not necessarily correspond to the simplicity of its expression in any particular language). However, from a pragmatic perspective, good representations need not be the simplest possible, for reasons that include engineering tradeoffs, the difficulty (and inherent ambiguity) of measuring simplicity, and social and cultural factors, for example relating to esthetics. Similar considerations apply, though to a notably lesser extent, to the simplicity of semiotic theories, since creating such theories is itself a design task, subject to various trade-offs. It may be reassuring to be reminded that in general there is no unique best representation.

The next two sections develop some basic theory of algebraic semiotics. Two main concepts are semiotic theory and semiotic morphism, which generalize the conceptual spaces and conceptual mappings of Fauconnier and Turner (1998, 2002), by taking account of structure and dynamics. Some measures of quality and design principles are given, including a trade-off between form and content. Although similar principles can be found in many places, none seem to be either as precise or as general as those described here. This section also discusses metaphor and blending in natural language, and gives some basics of a calculus of representation. A number of case studies, including information visualization, proof presentations, humor, and user interaction, are then described, and a discussion of narrative is also given, followed by some conclusions, future research directions, and social implications for virtual worlds.

Before beginning, it may help to be clear about the philosophical orientation of this work, because it is very common in Western culture for mathematical formalisms to claim and be given a status beyond what is warranted. For example, Euclid wrote, “The laws of nature are but the mathematical thoughts of God.” Similarly, the “situations” in the situation semantics of Barwise and Perry, which resemble conceptual spaces (but are more sophisticated—perhaps too sophisticated) are considered to be actually existing, ideal Platonic entities

(Barwise & Perry, 1983). Somewhat less grandly, one might consider that conceptual spaces are somehow directly instantiated in the brain. However, the point of view of this chapter is that such formalisms are constructed in the course of some task, with the heuristic purpose of facilitating consideration of certain issues in that task, which might be scientific study or engineering design. Under this non-Platonist view, all theories are situated social entities, mathematical theories no less than others; of course, this by no means implies that they cannot be useful.

Algebraic Semiotics

The basic notions of algebraic semantics are sign (or semiotic) system, semiotic morphism, and representation quality; these are discussed in the following subsections.

Signs and Sign Systems

The definition below of sign system incorporates the insight of Saussure (1976) that individual signs should not be studied in isolation, but rather as elements of systems of related signs; of Peirce (1965) that signs may have parts, subparts, and so forth that play different roles; and of Goguen (1999a) that sign parts have different saliencies, depending on the roles that they play.

The structure of a sign system can be described by an algebraic theory, since they are in particular abstract data types, and it is well known that these can be defined algebraically (Goguen & Malcolm, 1996). In addition, signs become what they are by virtue of attributes that differ from those of other signs, as shown for example by vowel systems (how the space of possible vowel sounds is divided into specific vowels for a given dialect of a given language), as well as by traffic signs, alphabets, and numerals. However, these attributes need not be binary, as was supposed by Saussure and his followers in the French structuralist movement including Levi-Strauss and early Barthes. Also, the same sign in a different system can have a different meaning, as illustrated by the way similar characters in different alphabets can take different meanings, for example, in the Roman and Cyrillic alphabets, the token “P” denotes different sounds.

We formalize[1] sign system as many sorted loose algebraic theories with data, plus two additional items that are specifically semiotic:

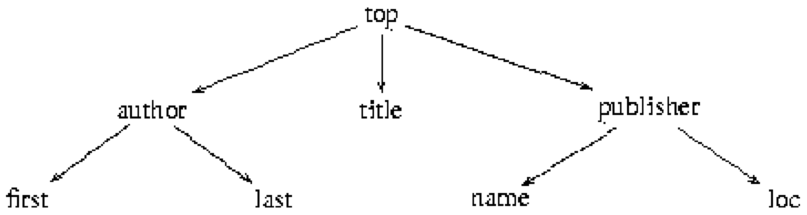
Definition 1: A **sign system**, or **semiotic system** or **semiotic theory**, consists of:

1. A **signature**, which declares sorts, subsorts, and operations (including constructors and selectors);
2. A subsignature of **data sorts**[2] and **data functions**;
3. Axioms (e.g., equations) as constraints;
4. A **level ordering** on sorts, including a maximum element called **top**; and
5. A **priority ordering** on constructors at the same level.

The non-data sorts classify signs and their parts, just as in grammar the "parts of speech" classify sentences and their parts. There are two kinds of operation: **constructors** build new signs from old signs as parts, while **selectors** pull out parts from compound signs. Data sorts classify a special kind of sign that provides values serving as attributes of signs. Axioms act as constraints on what count as allowable signs for this system. Levels indicate the whole/part hierarchy of a sign, with the top sort being the level of the whole; priorities indicate the relative significance of subsigns at a given level; social issues play a dominant role in determining these. The above definition follows Goguen (1999a), where the special treatment of data sorts follows Goguen and Malcolm (2000). The first four items constitute what is called an **algebraic theory** when all axioms are equations (e.g., Goguen & Malcolm, 1996; Goguen et al., 1978); it can be shown that this special case is sufficient for our needs.

The approach of Definition 1 differs from the more traditional set-based approaches of Gentner (1983) and Carroll (1982) in that it is axiomatic, that is, it does not present signs as particular models, but rather, a particular theory expressed in a formal language describes a space of possible signs, which are models of the theory, in the sense of that term in logic, providing concrete interpretations for the things in the theory: sorts are interpreted as sets; constant symbols are interpreted as elements; constructors are interpreted as functions, and so forth; that is, the theory is a *language* for talking about such models. This approach allows both multiple models and open structure, both of which are important for applications. The first point means, for example, that a semiotic space of books should allow anything having the structure of a book as a model; it also means that designers and implementers have the freedom to optimize implementations so long as they respect the constraints of the given axioms. The second point says that structure can be extended and combined without violating the specifications, which is not necessarily the case for models. For example, we might want to extend a basic sign system for books with some further structure pertaining to a certain series of books from a particular publisher. In addition, it

Figure 1. Levels for the book semiotic theory



is also more natural to treat levels and priorities in axiomatic theories than in set-based models.

For an example of axioms, in formalizing indices of books, we might well want to impose an axiom requiring that indexed items must be phrases of one, two, or three words, but not more. In a sign system for books, the top level might be occupied by the sort for books; the next level by author, title, publisher; and the third level by the first and last names of authors, and by the name and location of the publisher (see Figure 1). Here, last name has priority over first name, and publisher name has priority over publisher location. This is similar to the nesting structure used in XML documents.

The following are some further informal examples of sign systems: dates; times; bibliographies (in one or more fixed format); tables of contents (e.g., for books, again in fixed formats); newspapers (e.g., the *New York Times*); and a fixed Web site such as the CNN homepage (in some particular instance of its gradually evolving format). Note that each of these has a large space of possible instances, but a single fixed structure.

There is a basic duality between theories and models. We have already discussed one aspect: A semiotic theory determines the class of models that satisfy it, which we call its **semiotic space**.[3] The other aspect is more subtle: A class of models has a unique (up to equivalence), most restrictive theory whose models include it.[4] This duality helps to justify our occasional use of the term "space" when we really mean "theory"; this is mainly done for consistency of terminology when discussing conceptual blending theory.

Fauconnier (1985) introduced **mental spaces** for studying meaning in natural language from a cognitive point of view. The abstract mathematical structure of a mental space is a set of atomic elements together with a set of relation instances among those elements (Goguen, 1999a), and as such is a very special case of a sign system. Any such representation necessarily omits the qualitative, experiential aspects of what is represented (these aspects are often called "qualia"), since formal representations cannot capture meaning in any human sense. Moreover, mental spaces are not powerful enough for designing virtual worlds or other applications where structure and dynamics are important; obvious examples include wikis, Web sites, and music.

The conceptual spaces of Fauconnier and Turner (1998, 2002) are mental spaces, and hence share their limitations. For example, conceptual space theory can help us understand concepts about music, but semiotic spaces and structural blending are needed for an adequate treatment of the *structure* of music, for example, how a melody can be combined with a sequence of chords. Conceptual spaces are good for talking about concepts about (e.g., how we talk about) things, but are awkward for talking about the structure of things. It is also interesting to notice that greater cultural variation can be found in conceptual blending than in structural blending, because the former deals with concepts about something, whereas they latter deals with the structure of its instances and/or its representations. Mathematically, a conceptual space is a single model, consisting of items and assertions that certain relations hold of certain of those items; it is not a theory or a class of models.

Our suggested methods for determining semiotic spaces are grounded in ideas from sociology, especially ethnomethodology, but this chapter is not the right place to discuss such issues (see Goguen, 1997, 1994), beyond noting that semiosis, which is the creation of meaning, is always situated and embodied, and in particular always has a social context. Immersion arises in part through embodiment (even if only metaphorical embodiment, e.g. in text-based virtual worlds).

Representations

Mappings between structures became increasingly important in twentieth century mathematics and its applications; examples include linear transformations (and their representations as matrices), continuous maps of spaces, differentiable and analytic functions, group homomorphisms, and much more. Mappings between sign systems are only now appearing in semiotics, as uniform representations for signs in a source space by signs in a target space. Since we formalize sign systems as algebraic theories with additional structure, we should formalize **semiotic morphisms** as theory morphisms; however, these must be partial, because in general, not all of the sorts, constructors, and so forth are preserved in real-world examples. For example, the semiotic morphism that produces an outline from a book omits the sorts and constructors for paragraphs and sentences, while preserving those for chapters, sections, and so forth. In addition to the formal structure of algebraic theories, semiotic morphisms should also (partially) preserve the priorities and levels of the source space. The extent to which a morphism preserves the various features of semiotic theories is important in determining its quality, as the case studies to follow will show.

The design of virtual worlds, and more generally of user interfaces, is the art of creating representations, for example representing the functionality of an operating system using icons, menus, buttons, and so forth, or using haptics and virtual reality. The basic insight is that a representation is a mapping $M : S_1 \to S_2$ of sign systems that preserves as much as is reasonable. The following formalizes this insight:

Definition 2: A **semiotic morphism** $M : S_1 \to S_2$ from a semiotic system S_1 to another S_2 consists of the following partial mappings:

1. from sorts of S_2 to sorts of S_2, so as to preserve the subsort relations,
2. from operations of S_2 to operations of S_2, so as to preserve their source and target sorts,
3. from levels of S_1 sorts to levels of S_2, so as to preserve the ordering relation, and
4. from priorities of S_1 constructors to priorities of S_2 constructors, so as to preserve their ordering relations, so as to strictly preserve all data elements and their functions.

It is not always possible or even desirable for a semiotic morphism to preserve everything. For example, sometimes we just want to summarize some dataset, such as the table of contents of a book, in which case much of the structure and information are intentionally deleted. Another important observation is that not all representations are equally desirable. For example, one way to parse the sentence “Time flies like an arrow” in the following “bracket” (or “bracket-with-subscript”) notation, which is widely used in linguistics, is:

$$[[\text{time}]_N[[\text{flies}]_V[[\text{like}]_P[[\text{an}]_{Det}[\text{arrow}]_N]_{NP}]_{PP}]_{VP}]_S$$

However, this notation is not very satisfactory for humans, who would find it easier to discern the syntactic structure by examining a parse tree, or using the algebraic “constructor” notation given later. Some criteria for judging the quality of representations are discussed next in the section, “Quality of Representation.”

The duality between theories and models means that there is an inherent ambiguity about the direction of a semiotic morphism. For example, if *B* is a sign system for books and T is one for tables of contents, then books (which are models of B) are mapped to their tables of contents, which are models of T, but this map on models is determined by, and is dual to, the theory inclusion $T \to B$, which expresses the fact that the structure of tables of contents is a substructure of that of books. In informal discussions we will often take the direction to be that of the models, which is perhaps more intuitive; however, in formal discussions, it is much better to use the direction of the underlying theory morphism, which is opposite to that of the models.

There are at least three "modes" in which one might consider representations: analytic, synthetic, and conceptual. In the **analytic mode**, we are given one or more signs from the representation (i.e., the target) space, and we seek to reconstruct both the source space and the representation. In the **synthetic mode**, we are given the source space and seek to construct a good representation for the signs in that space, using some given technology (such as command line, or standard GUI widgets, or virtual reality) for the target space. In the conceptual mode, we seek to analyze the metaphorical structure of the representation, in the style of cognitive linguistics (Turner, 1997; Fauconnier & Turner, 2002); for example, how is Windows XP like a desktop, or how is a scrollbar like a scroll? A treatment in this mode will involve conceptual spaces, in the sense of cognitive linguistics (see the section "Metaphor and Blending"). In each mode, particular aspects of the cultures involved can be very significant.

Simple Examples

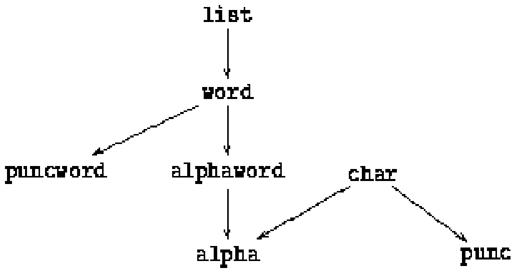
This subsection gives rather informal descriptions of some simple examples of semiotic theories and semiotic morphisms to illustrate the concepts, rather than to demonstrate their applicability to virtual world design, since these examples could only be considered virtual worlds in a trivial sense. The following sign systems are considered:

1. Lists of (potential) words with punctuation, denoted S_W.
2. Parse trees for sentences of a formal grammar G, denoted S_T.
3. A printed page format, denoted S_P.

Then the following are some interesting morphisms:

1. Let $P : S_W \to S_T$ give parse trees for lists from S_W that are G-sentences.
2. Let $H : S_T \to S_P$ give bracket representations of parse trees.
3. Let $F : S_W \to S_P$ give bracket representations of lists from S_W that are G-sentences.

Figure 2. Sorts and subsorts for S_T



The morphism F is "composed" from P and H, by first doing P and then doing H; we denote this by $P;H$, where ";" denotes the composition operation. Composing morphisms correspond to composing representations, which is the essence of iterative design, an important technique for any complex design task. By Definition 2, a semiotic morphism M has four component mappings for sorts, operations, levels, and priorities; let us denote these M_1, M_2, M_3, and M_4, respectively. Then the **composition** $M;M'$ of morphisms M and M can be defined by the formula $(M;M')_i = M_i;M'_i$ for $i=1,2,3,4$.

The sign system S_W for punctuated lists of words can be described roughly as follows: Its sorts are `char`, `alpha`, `punc`, `puncword`, `alphaword`, `word`, and `list`, where the sorts `alpha` and `punc` are subsorts of `char`, the sorts `alpha`, `puncword` and `alphaword` are subsorts of `word`, and the sort `word` is a subsort of `list`. These subsort relationships are shown in Figure 2. The sorts `char`, `alpha`, and `punc` are respectively for character, alphabetic character, and punctuation character; the sorts `puncword`, `alphaword`, and `word` are respectively for words consisting of alphabetic characters with a final punctuation sign, words with all alphabetic characters, and the union of these two.

The sort `list` is Level 1, the "top" sort of this system; `word` is Level 2; `alphaword` and `puncword` are Level 3; and `char`, `alpha` and `punc` are Level 4. The punctuation characters are comma and period (of course we could add more). The following defines concatenation constructors for constructing a list of alphabetic characters as a `alphaword`, a list `alphaword` followed by a punctuation as a `puncword`, and a list of `words` as a `list`; a functional notation is used:

```
_ _ : alpha alphaword → alphaword
_ _ : alphaword punc → puncword
_ _ : word list → list
```

Here the two underbars give the syntactic form of the function, which is the concatenatation of its two arguments, which respectively have the two types given between the colon and the arrow; the result type then comes after the arrow. These three operations also satisfy associativity equations, such as:

$$(\forall\ W\ L\ L')\ (W\ L)\ L' = W\ (L\ L')\ .$$

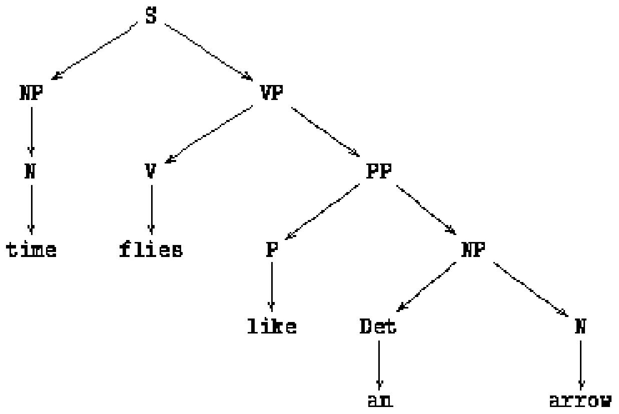
The context-free grammar *G* given below allows some lists of punctuated words to be recognized as legal sentences of that grammar; such sentences can be parsed, which means dividing them into **phrases**, which are sublists, each with its "part of speech" (or syntactic category) explicitly given. The grammar *G* will become the signature of the sign system S_T. The non-terminals of *G* are `S`, `NP`, `VP`, `N`, `Det`, `V`, `PP`, and `P`, which stand for sentence, noun phrase, verb phrase, noun, determiner, verb, prepositional phrase, and preposition, respectively. Then the rules of a simple example *G* might be the following:

```
S  → NP VP
NP → N
NP → Det N
VP → V
VP → V PP
PP → P NP
```

The non-terminals of this grammar (i.e., the "parts of speech" `S`, `NP`, `VP`, etc.) are the sorts of the sign system S_T, and the rules of the grammar will become its constructors. For example, the first rule says that a sentence can be constructed from an `NP` and a `VP`. There should also be some constants of the various sorts, such as:

```
N   → time
N   → arrow
V   → flies
Det → an
Det → the
P   → like
```

Figure 3. Parse tree for S_T



Then, for example, a parse tree for the sentence "Time flies like an arrow" is shown in Figure 3.

By the way, if we add the productions below to the grammar *G*, then the sentence gets another parse, a fact that the reader might enjoy checking.

```
NP → N N
V → like
```

There is a systematic way to convert context-free rules into constructor operations in the signature of a sign system; for the above grammar *G*, it is as follows, written in a functional notation:

```
sen : NP VP → S
nnp : N → NP
np : N Det → N
vvp : V → VP
vp : V PP → VP
pp : P NP → PP
time : → N
flies : → V
...
```

In this context, it is more elegant to regard `N` as a subsort of `NP`, and `V` as a subsort of VP, rather than to have monadic operations `N` $\rightarrow$ `NP` and `V` $\rightarrow$ `VP`. This sign system gives what computer scientists call abstract syntax for sentences; it gives an abstract algebraic representation for syntactic structure, in which the operations above generate a free algebra of terms that describe parses. For example, the term that represents the syntax of our example sentence is:

```
sen(time, vp(flies, pp(like, np(an, arrow)))).
```

Equations can be used in this algebraic setting to express constraints on sentences, for example, that the number of the subject and of the verb agree (i.e., both are singular or else both are plural). Each of the concrete ways to realize abstract syntax (trees, terms, bracket notation, and lists) can be considered to give a model of the sign system S_T providing a set of signs for each sort, and operations on those sets which build new signs from old ones.

The sign system S_P should have sorts for lines and pages, and could also have different fonts and subscripts in order to display the bracket notation to display parses. We omit the details, which are not very different from those above, except for an equation to limit the length of lines, for example to 80 characters, such as the following:

$$(\forall L\colon \texttt{line})\ \mathrm{length}(L) \leq 80\ .$$

The morphism $P : S_W \rightarrow S_T$ is very partial, since it is defined on a list l if l can be parsed using G; thus the subset of lists on which it is defined is the set of sentences generated by G, which is usually denoted $L(G)$. If $fr(t)$ denotes the **frontier**, or list of leaf nodes, of a parse tree t, then $fr(P(l)) = l$ for all $l \in X$, which is a strong preservation property, although it only holds on a small subset of lists of words. The morphism P also preserves all of the sort hierarchy in Figure 2.

We do not describe the morphism $H : S_T \rightarrow S_P$ in as much detail as we did the morphism $P : S_W \rightarrow S_T$. The morphism H is essentially a pretty printer for parse trees; it could use any of the representations we have been discussing, and it could even just print the frontier of the parse tree, although this would preserve much less structure (see the next subsection for more discussion of structure preservation). As already mentioned, the morphism $F : S_W \rightarrow S_P$ is the composition $P;H$; it prints the parse trees of those lists of words that can be parsed by G.

Quality of Representation

It is easy to define sort preserving, constructor preserving, level preserving, content preserving (where **content** refers to the values of selector operations, such as size and color), and so forth. But this is not as useful as one might hope, because in practice, these are often not preserved. Instead, we define the *comparative* notions of more sort preserving, more level preserving, more constructor preserving, and more content preserving (Goguen, 1999a). These notions define *orderings* on morphisms, which can be logically combined to get the right one for a given application (Goguen, 1999a). This is important because given morphisms M, M', one may preserve more levels, while the other preserves more content, and similarly for the other concepts. Empirical work has validated the following general principles:

1. It is more important to preserve structure than content (this is called **Principle F/C**).
2. It is more important to preserve level than priority.
3. Structure and content at lower levels should be sacrificed in favor of those at higher levels.
4. Lower level constructors should be sacrificed in favor of higher level constructors.

The first principle is perhaps the most important, and at first might seem counterintuitive, but many special cases can be found in the design literature (e.g., Tufte, 1983). It asserts that when a trade-off is necessary, form should be weighted more heavily than content; in general, the right balance between form and content can only be determined after knowing how a representation will actually be used. Also, we are fortunate that it is easier to describe structure than content.

These principles do not explain everything; for example, they do not explain why the tree representation of phrase structure is better than the bracket representation, since these two representations have exactly the same structure and content, but display them differently. In fact, the advantage of the tree representation arises from human visuo-cognitive capabilities, which prefer a more explicit diagrammatic representation of phrases and subphrase relations as nodes and edges, over a linear symbolic representation that requires counting brackets. Preservation of form and content can respectively be formalized as preservation of constructors and selectors, in the sense of abstract data type theory (Goguen et al., 1978; Goguen & Malcolm, 1996).

Fragments of a Calculus of Representation

The composition of semiotic morphisms has now been defined, and it was shown that this can be important for applications. It is easy to prove that this definition of composition obeys the following identity and associative laws, in which $A : R \rightarrow S$, $B : S \rightarrow T$ and $C : T \rightarrow U$:

$$A ; 1_S = A$$
$$1_S ; B = B$$
$$A ; (B ; C) = (A ; B) ; C$$

where 1_S denotes the identity morphism on S. These three laws are perhaps the most fundamental for a calculus of representation, since they imply that semiotic theories and their morphisms form what is called a "category" in the relatively new branch of mathematics called category theory (Mac Lane, 1998). The basic ingredients of a **category** are objects, morphisms, and a composition operation that satisfies the above three laws, and that is defined on two morphisms if and only if they have matching source and target. Perhaps surprisingly, many important mathematical concepts can be defined abstractly in the language of category theory, without reference to how objects are represented, using only morphisms and composition; moreover, many general laws can be proven about such concepts, and these automatically apply to every category.

Three of the simplest categorical concepts are isomorphism, sum, and product. A morphism $A : R \rightarrow S$ is an **isomorphism** if and only if there is another morphism $B : S \rightarrow R$ such that $A;B = 1_R$ and $B;A = 1_S$ in which case B is called the **inverse** of A and denoted A^{-1}; it can be proved that the inverse of a morphism is unique if it exists. The following laws can also be proved, assuming that $A : R \rightarrow S$ and $B : S \rightarrow T$ are both isomorphisms (and no longer assuming that B is the inverse of A).

$$1_R^{-1} = 1_R$$
$$(A^{-1})^{-1} = A$$
$$(A ; B)^{-1} = B^{-1} ; A^{-1}$$

Because sign systems and their morphisms form a category, these three laws apply to representations. In the section "Some Laws", we discuss sums of semiotic morphisms as a special case of blends of semiotic morphisms (blends are discussed in the next subsection), and we also give some laws for blends and sums. The standard mathematical reference for category theory is by Mac Lane (1999), but Pierce (1990) is one computer science-oriented introduction, among several others.

Metaphor and Blending

Research in cognitive linguistics by George Lakoff and others under the banner of "conceptual metaphor theory" (CMT) has greatly deepened our understanding of metaphor (Lakoff & Johnson, 1980; Lakoff, 1987), showing that many metaphors come in families, called **image schemas**, that share a common pattern. One example is BETTER IS UP, as in "I'm feeling up today," or "He's moving up into management," or "His goals are higher than that." Some image schemas, including this one, are grounded in the human body[5] and are called **basic image schemas**; they tend to yield the most persuasive metaphors, as well as to enhance the sense of immersion in virtual worlds.

Fauconnier and Turner (1998, 2002) study **blending**, or **conceptual integration**, claiming it is a basic human cognitive operation, invisible and effortless, but nonetheless fundamental and pervasive, appearing in the construction and understanding of metaphors, as well as in many other cognitive phenomena, including grammar and reasoning. Many simple examples are blends of two words, such as "houseboat," "jazz piano," "roadkill," "artificial life," "computer virus," "blend space," and "conceptual space." To explain such phenomena, blending theory (BT) posits that concepts come in clusters, called **conceptual spaces**, which consist of certain items and certain relations that hold among them. Such spaces are relatively small constructs, selected on the fly from larger domains, to meet an immediate need, such as understanding a particular phrase or sentence.[6] The abstract mathematical structure of a conceptual space consists of a set of atomic elements together with a set of relation instances among those elements (Fauconnier, 1985). **Conceptual mappings** are partial functions from the item and relation instances of one conceptual space to those of another, and **conceptual integration networks** are networks of conceptual spaces and mappings that are to be blended together.

We now describe our generalization of blending from conceptual spaces to semiotic theories. A simple example where this generality is needed is in the integration of a window with its scrollbar, which is structural, not conceptual, although conceptual aspects of this blend could also be studied; this example is discussed in considerable detail in Goguen (2003). To indicate this added generality, we will use the terms **structural blending** or **structural integration** for the blending of semiotic systems, which in general involves non-trivial constructors; but for consistency with BT, we use the phrase "semiotic space" instead of "semiotic theory" in this discussion. The simplest form of blend is as

Figure 4. A blend diagram

$$\begin{tikzcd}
& B & \\
I_1 \arrow[ur] & & I_2 \arrow[ul] \\
& G \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

shown in Figure 4, where I_1 and I_2 are called the **input spaces**, B the **blend space**, and G the **generic space**, which contains conceptual structure that is shared by the two input spaces.[7] Let us call I_1, I_2, and G together with the two morphisms $G \to I_1$ and $G \to I_2$ an **input diagram**. Then a **blendoid** over a given input diagram is a space B together with morphisms $I_1 \to B$, $I_2 \to B$, and $G \to B$, called **injections**, such that the diagram of Figure 4 **weakly commutes**, in the sense that both compositions $G \to I_1 \to B$ and $G \to I_2 \to B$ are **weakly equal** to the morphism $G \to B$, in the sense that each element in G gets mapped to the same element in B under them, provided that both morphisms are defined on it.[8] The special case where all four spaces are conceptual spaces gives conceptual blends. This diagram is "upside down" from that used by Fauconnier and Turner, in that our arrows go up, with the generic G on the bottom, and the blend B on the top. Our convention is consistent with duality mentioned earlier, as well as with the way that such diagrams are usually drawn in mathematics, and with the image schema MORE IS UP (since B is "more"). Also, Fauconnier and Turner do not include the map $G \to B$. By definition, the maps $G \to I_1$ and $G \to I_2$ are total, not partial, and if the input spaces were minimal, then the maps $I_1 \to B$, $I_2 \to B$, $G \to B$ would also be total.

Usually an input diagram has many blendoids, only a few of which are interesting. Weak commutativity of the blend diagram, which is included in the definition, is a good first step, but still leaves too many possibilities. Therefore additional principles are needed for identifying the most interesting possibilities, so that we can define a **blend** to be a blendoid that is optimal with respect to these principles. Fauconnier and Turner suggest a number of "optimality principles" that serve this purpose (see Chapter 16 of Fauconnier & Turner, 2002), but they are too vague to be easily formalized. A tentative and difficult but precise mathematical approach is given in Appendix B of Goguen and Malcolm (1996), based on a modification of the category theoretic notion of "pushout" (Mac Lane, 1998); this modification takes advantage of an ordering relation on morphisms, along the lines discussed above. The intuition is that nothing can be added to or subtracted from such an optimal blendoid without violating consistency or simplicity in some way. However, there can still be more than one blend in this sense, as an example discussed below will make very clear. It should also be noted that this notion of blend easily generalizes to any number of semiotic spaces, and even to arbitrary diagrams of semiotic spaces and morphisms, for which there are many significant applications. Thus, the emphasis of Fauconnier and Turner (2002) on blends having the form of Figure 4 seems inappropriate for algebraic semiotics, because its major applications typically involve multiple spaces and multiple morphisms among them.

It has perhaps not been sufficiently emphasized in the BT literature that blending does not always give a unique result. For example, the following are four different blends of conceptual spaces for “house” and “boat”:

1. houseboat;
2. boathouse;
3. amphibious RV; and
4. boat for moving houses.

The last may be a bit surprising, but I once saw such a boat in Oban, Scotland, transporting prefabricated homes to a nearby island. There are also some other, even less obvious blends (Goguen & Harrell, 2004).

In the UCSD Meaning and Computation Lab, Fox Harrell and I have been experimenting with a blending algorithm, which has generated novel metaphors, which in turn were used in generating poems (Goguen & Harrell, 2004) with some success before a live audience. The algorithm uses dynamic programming to generate blends in approximate order of optimality, and if requested, can generate all possible blends, including even very bad ones. One surprise was that there were so many blends, for example, 48 for the (small) house and boat spaces.

The CMT view of metaphor associates aspects of one domain to another, and describes this association using a mapping, of which the target domain concerns what the metaphor is “about.” On the other hand, BT views metaphors as “cross-space mappings” that arise from blending conceptual spaces extracted from the domains involved. For example, the metaphor “my love is a rose” arises from blending conceptual spaces for “my love” and “rose,” such that the identification of the two items “love” and “rose” in the blend space gives rise to a correspondence between certain items in the rose space and the target love space. Such **metaphoric blends** are *asymmetric*, in that as much as possible of the target space is imported into the blend space, whereas only key aspects from the source space, associated with elements that have been identified with elements of the target space, are imported, for example, sweet smell and attractive color; moreover, names from the top space take precedence over those in the source space, so that relations in the source space become “attributed” to items in the target space. Our approach differs from orthodox BT not only in that we allow many more kinds of structure in our spaces, but also in that we do not first construct a minimal image in the blend space and then "project" that material back to the target space, but instead we construct the entire picture in the blend space. Thus it is not the case for us that, in forming the blend, elements are preferentially omitted from the target space, only to be restored upon projection, as described in Grady et al. (1999). Since CMT has been mainly concerned with families of metaphors having a shared pattern, and BT has been more concerned with how novel metaphors can be understood, the two theories are compatible and can both play a role in understanding complex language. This and related issues are discussed with many interesting details and examples in Grady et al. (1999).

Algebraic semiotics also goes beyond conceptual spaces in allowing entities that have dynamic states; this is necessary for applications to the dynamic entities that appear in user interface design and virtual worlds. Actually, two kinds of dynamics are involved in blending: the process of blending itself, and entities with internal states. Whereas cognitive linguistics has so far focused mainly on the former, algebraic semiotics is more concerned with the latter. Another difference from BT is that relations like causality are represented as ordinary relations rather than being given a special *ad hoc* status.

The conceptual spaces, mappings, and blending of cognitive linguistics seem well adapted for treating many aspects of literature, as in Turner (1997), as well as some recent trends in art, including (the very aptly named) conceptual art movement, and with the conceptual aspects of works in many other styles, which are often designed to provoke conceptual conflicts or to force unusual conceptual blends. One important application is the combination of music with lyrics, as skillfully studied using cross-domain mappings by Zbikowski (2002). Unfortunately, the framework of conceptual blending seems too restricted for studying blending *within* music, for example, harmony, polyphony, polyrhythm, and so forth, because musical structure is inherently hierarchical, and hence cannot be adequately described using only atomic elements and relation instances among them. Understanding how a particular melody, chord sequence, and rhythm can work together requires attention to the component notes, phrases, chords, and beats, as well as to their subcomponents. However, it seems that the added generality of semiotic spaces and semiotic morphisms is adequate for such purposes.

Some Laws

This subsection gives some further fragments of a calculus of representation (see Goguen, 1999a, for more detail). Here a,b,c are semiotic morphisms, and

$\Diamond$ denotes some choice of a blend that is maximal with respect to some optimality criterion:

$$a \Diamond b \cong b \Diamond a$$
$$a \Diamond (b \Diamond c) \cong (b;a) \Diamond c$$
$$(a \Diamond b) \Diamond c \cong a \Diamond (b;c)$$

In the following, *A,B,C* may be either semiotic morphisms or just semiotic systems. Sums, denoted +, are the special case of blend where the base theory is **1**, which is the theory having exactly one constant, its top element, and nothing else.

$$A + \mathbf{1} \cong A$$
$$\mathbf{1} + A \cong A$$
$$A + B \cong B + A$$
$$A + (B + C) \cong (A + B) + C$$

It should be noted that products of models correspond to sums of theories, that is, a model of a sum of theories is a product of models of the summand theories, and *vice versa*, or even more formally, there is an isomorphism of categories of models:

$$Mod(A+B) \cong Mod(A) \times Mod(B) ,$$

where *A, B* and are semiotic theories (see Goguen, 1999a, for details).

Case Studies

This section surveys some case studies applying algebraic semiotics. Noting that we have already discussed blending and metaphor, the following additional case studies are considered:

1. Information visualization,
2. Proof presentation,

3. Humor, and
4. User interaction.

The first category in this list actually contains three small case studies, and the second can also be considered a special case of it; proof visualization is our most extensive case study, part of a large project to produce a Web-based system to support theorem proving. The study of humor is somewhat of a digression, but it is hoped that the reader will find it amusing. Proof navigation is used to illustrate how interaction is treated in algebraic semiotics, although many details are left out, because the formal theory of dynamic signs is technically rather complex.

Information Visualization

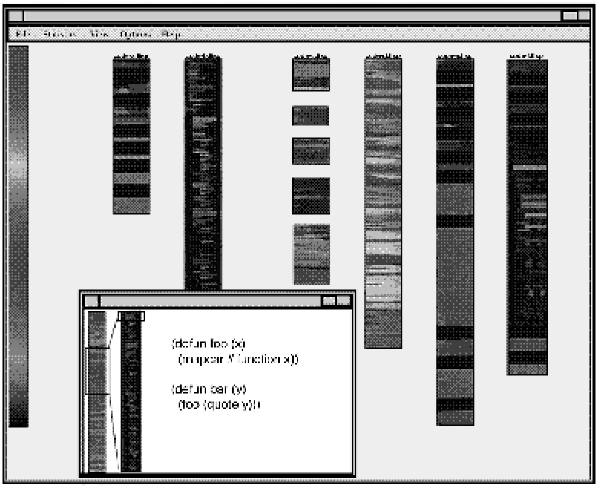
Visualizing complex data can help to discover, verify, and predict patterns, and to quickly locate specific information; but it can be difficult to construct the appropriate visualizations for these purposes. Because visualizations are representations, our theory applies to them, and in particular, our quality measures apply. The following subsections analyze three real visualization systems as semiotic morphisms, and on that basis, suggest some improvements. We found it convenient to use algebraic semiotics in a semi-formal style, letting the ideas and results guide the re-design, and introducing formal details only to the degree that they actually help with decisions. Many aspects of these discussions follow (Goguen & Harrell, 2003).

Code Visualization

A visualization tool for code developed at ATT Bell Labs, and discussed in Shneiderman (1997), displays the large grain structure of code by omitting details (see Figure 5). This is an excellent illustration of Principle F/C: commands are indistinguishable lines, but files and procedures are easily distinguished, and the age of code is highlighted with color (though it shows up as shades of gray in this figure), presumably because code age is so important for software maintenance, which accounts for most of the cost of large software systems. Moreover, code at the command level can be viewed in a separate window, which is activated by “zooming in” from the main overview window. However, software engineers often need to find other specific features of code, such as:

1. Occurrences of particular variables,
2. Certain uses procedure calls, and
3. Certain uses of pointers.

Figure 5. A code browser



```
(defun foo (x)
  (mapcar #'function x))

(defun bar (y)
  (foo (quote y)))
```

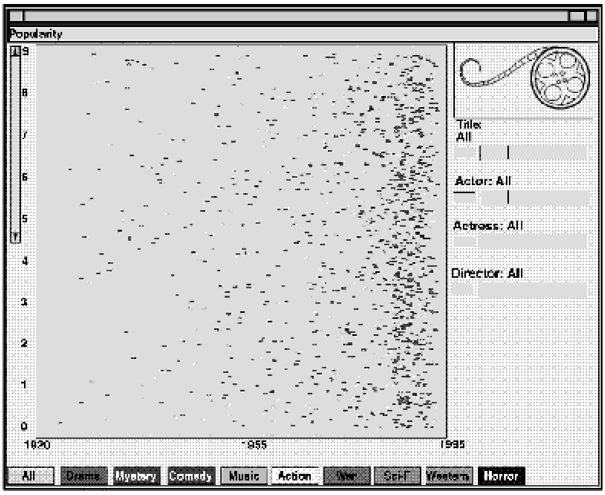
Or consider what would be needed to work on the Y2K problem. To support this kind of flexibility, the system should allow users to select and highlight a variety of features to be displayed with color, not just code age; indeed, each feature listed above could be highlighted with a different color, because these features are binary (i.e., they either occur or do not occur, at any given point in the program), rather than, like age, being measured on a (nearly) continuous scale.

A Film Visualizer

Figure 6 shows FilmFinder, a system to help consumers find films, designed in Ben Shneiderman's group at the University of Maryland, as described in Shneiderman (1998). The vertical axis indicates popularity, the horizontal axis indicates the release date, and the color[9] indicates the genre; the area on the right side of the display is for controlling the system. This complex sign is the image under an appropriate semiotic morphism of a sign in a space of information about films. From this, we infer that the designer of the system thought users would consider the popularity, date, and genre to be the most important attributes of films.

Instead of thinking of it as a consumer product, it is interesting to think of this system as a scientific tool for displaying data about the movie industry. Using it in this way, we can see that the density of films increases rapidly in the most recent years displayed, except perhaps for those genres that are the least popular; and we can also easily see some other facts, such as that there has

Figure 6. FilmFinder display



Title: All

Actor: All

Actress: All

Director: All

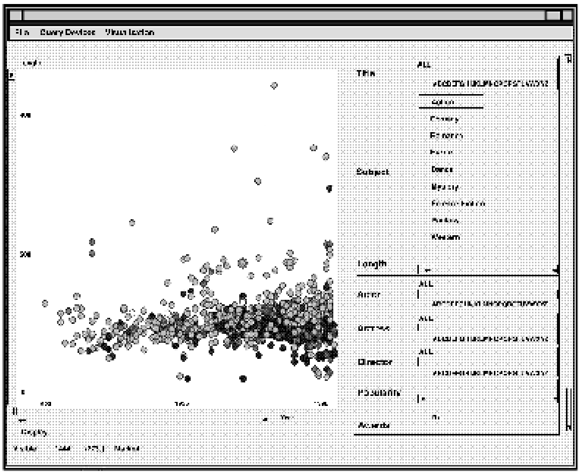
always been a higher percentage of drama, and that there are increasing percentages of action and horror.

However, this representation is less useful than it could be for this purpose. The problem is again that too much content and not enough structure have been preserved. For example, it would be better to aggregate all films having approximately the same attributes of interest into one blob, and then display the number of films in a blob using a distinct visual attribute, such as size or brightness. Successive blobs of the same kind could then be connected by lines having the same color as the blobs. Users could click on a blob to see what's in it, preferably displayed graphically in a new popup window. These revisions would facilitate hypothesis formation, and would also make the tool more useful for consumers, especially when (as in the most recent years that are not represented in the figure) there are many more films.

A Later Version

Figure 7 shows a later version (SpotFire from ivee Development in Sweden) of the FilmFinder tool in Figure 6; the main improvement is that users have more control over what is displayed and how it is displayed. This particular display has length and date as its axes, and again uses color for genre, although the genre color coding scheme is not explicitly shown; prize winning films are highlighted by having a larger size. It is interesting to observe a clustering at around 90 minutes length. But once again, the display is difficult to use because there are

Figure 7. SpotFire version of FilmFinder



too many dots, even though this display cuts off at 1990! If the user is seeking a particular film or class of films, she will want to narrow the search focus by imposing additional constraints, but from this single display, it is difficult to know how easily that could be done. We are presumably supposed to assume that the (possibly imaginary) user who created this display considered these particular attributes the most interesting at a certain point during a sequence of displays, constituting a search; but in fact, they do not seem especially useful for any particular purpose.

We can also infer what the designer of this version thought would be most important, by examining the controls on the right of the display; we can hope that these were determined by polling an adequate pool of typical users. But the key issue is how convenient these controls are for scenarios that typical users find particularly important; most likely, those typical users are looking for a good video to rent, rather than analyzing trends in the movie industry, and so the controls should reflect the key actions involved in those searches, rather than just the most important general attributes of films. It would take some experimental work to determine these most relevant search attributes, but we can still criticize the design of the control console, because of its exclusive focus on simple attributes instead of structure. And we can also criticize the fine grain control that it gives users over length and year, and suggest instead that soft constraints would be more appropriate; it also seems doubtful that length is a highly significant attribute for search. Moreover, we can criticize the design philosophy, advocating instead a more social approach that relates the profile of one user to the profiles of other users to select films that similar users have found interesting (there are numerous variations on this, such as listing films that a user's friends have liked; Amazon has exploited similar strategies very successfully). Finally, we can note that the design ideas proposed to improve the previous version of this system also apply to the new version.

Proof Representation and Understanding

It is well known (perhaps too well known to many unhappy students) that understanding mathematical proofs can be very difficult. But why is it difficult? And how can this situation be improved? The UCSD Tatami project (Goguen et al., 2000) aims to make proofs more interesting and even enjoyable to read, by viewing them as representations of their underlying mathematics, so that we can apply algebraic semiotics, including the theory of representation quality.

The Kumo system generates proof Web sites, based on user-provided sketches in a language called Duck. The pages are in XML, displayed using XSL style sheets, and can be viewed over the Web using any browser. The complex signs that users actually see are called **proofwebs**, consisting of English phrases and sentences, mathematical signs, navigation buttons, formal input and output for a mechanical theorem prover, and so forth (Goguen et al., 2000; Goguen, 1999b).

Our view of what constitutes the underlying mathematics to be displayed is unusual: we consider it to include not just the tree structure of proofs, decorated with formal sentences and rules, as is common among computer scientists and logicians, but also:

1. A dramatic structure, following Aristotle (see below);
2. A narrative structure (following ideas of Labov (1972) and Linde (1981), as briefly described below);
3. Hyperlinks to related material, including tutorials for proof rules used, input and output to a formal theorem prover (if available), and motivation and explanation for proof strategies and steps; and
4. Image schemas (in the sense of Lakoff & Johnson, 1980; Lakoff, 1987). As with any virtual world, image schemas can make the language more direct and powerful, and hence easier to follow.

Aristotle (1997) said, "*Drama is conflict,*" which suggests providing conflict to add drama to proofs. Finding a non-trivial proof usually requires exploring many dead ends, errors, and misconceptions, some of which may be very subtle. Therefore the process of proving can be full of disappointed hopes, unexpected triumphs, repeated failures, and even fear and interpersonal conflict. All this is typically left out when proofs are written up, leaving only the map of a path that has been cleared through the jungle. But proofs can be made much more interesting and understandable if some of the conflicts that motivated their difficult steps are integrated into their structure; proof obstacles are exactly what is needed for drama. Of course, this must be done with care, and it should not be overdone, just as in a good novel or movie. Our Kumo theorem proving system (Goguen et al., 2000) used these ideas to structure the Web sites that it generates to display proofs. Aristotle also gave other useful suggestions, including unity of time and place, and having a beginning, middle, and end to a drama (Aristotle, 1997).

Labov (1972) showed that oral narratives of personal experience have a precise internal structure, which includes the following:

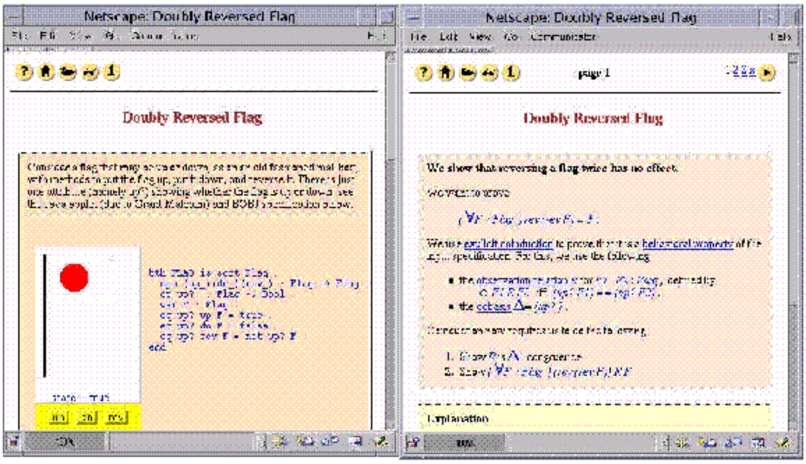
1. An optional **orientation section**, which provides basic orientation information, such as the time and place of the story, and perhaps some major characters;
2. A sequence of **narrative clauses** that describe the events of the story;
3. The **narrative presupposition**, which by default assumes that the ordering of the narrative clauses corresponds to the temporal ordering of the events that they describe;
4. **Evaluative material** integrated with the narrative clauses, which "evaluates" the events, in the sense of relating them to socially shared values; and finally
5. An optional **closing** section, which may contain a "moral" or a summary for the story.

The above follows Linde (1981, 1993), who describes developments subsequent to the classic treatment of Labov (1972). Although this empirical research used oral narratives of personal experience as data, its results apply much more broadly (though in general less precisely), since the class of narratives is the core around which many discourse types are built.

To aid our discussion of proofs as representations, we introduce terminology for the source and target semiotic spaces: let us call their elements **abstract proofwebs** and **proof displays**, respectively, and perhaps also use the term **display proofweb** for target signs. In addition, the term **unit** refers to a block of information of the same kind in a proof display. The display proofwebs generated by Kumo adhere to the following style guidelines, called the **tatami conventions** (Goguen et al., 2000). The first eight are justified mainly by narratology:

1. **Homepages** are provided for every major proof part; homepages introduce and motivate the problem to be solved and the approach taken to the

Figure 8. A typical tatami homepage and proof page


Netscape: Doubly Reversed Flag
Netscape: Doubly Reversed Flag
File Edit View Go Communicator
page 1
Doubly Reversed Flag
Doubly Reversed Flag
Explanation

We show that reversing a flag twice has no effect.

solution for that part, and correspond to the orientation sections of Labov's narrative structure; they may contain graphics, applets, and of course text. Homepages appear in the same window as their tatami pages (see the next item) because they are part of the same narrative flow (see Figure 8).

2. **Tatami pages**, also called **proof pages**, are the basic constituents of display proofwebs; they are XML pages containing one or more proof units, with its inference rule applications, interleaved with one or more explanation units. This integration follows the interleaving of narrative and evaluative material in Labov's theory. Limiting the number of non-automatic proof steps on tatami pages to approximately seven is justified by the classic work of Miller (1956) on limitations of human cognitive capacity; this limitation also makes it feasible to place both proof and explanation units on the same proof page (see Figure 8).
3. The **explanation units** of tatami pages are prover-supplied informal discussions of proof concepts, strategies, obstacles, and so forth. They correspond to the evaluative material in Labov's theory, and motivate important proof steps by relating them to values shared in the appropriate community of provers.
4. Tatami pages can be browsed in an order designed by the prover to be helpful and interesting to the reader; if possible, they should tell a story about how obstacles were overcome (or still remain). This narrative order again comes from Labov's theory, while including obstacles comes from the work of Campbell (1973) and others on "heroic" narratives.

5. Major proof parts, including lemmas, have their own subsites, each with the same structure as the main proof, including homepage and explanation units. These appear in a separate dedicated persistent popup window. Having separate hyperlinked Web sites for major proof parts is similar to the way that flashbacks and other temporal dislocations occur in stories. It is helpful to have them in a separate window in order to clarify their relation to the main sequence of proof steps.
6. Tatami pages also have associated formal proof scores, which appear in another separate popup window when summoned from a tatami page. The separate window is convenient because users typically want to look at the formal proof and its motivation at the same time as the proof score. Users can also request proof score execution, and the result is displayed in the same window as the score, so that one can easily alternate between them. (The proof score is sent to an OBJ server and the result is returned for display.) This hiding of routine details is similar to human proofs, which use it to highlight the main ideas (Livingston, 1987).
7. Major proof parts can have an optional closing page, to sum up important results and lessons, again following Labov's theory. They appear in the same window as proof pages, again because they are part of the same narrative flow.
8. A menu of open subgoals appears on each homepage, and error messages are placed on appropriate pages. Open subgoals are important to provers when they read a proof, since proving new results is a major value within this community.

Now we give further style guidelines, with justifications based on algebraic semiotics:

1. **Windows:** The main contents of a display proofweb are its proof steps, informal explanations, tutorials, and mechanical proof scores. These four are also the main contents of abstract proofwebs, and their preservation has much to do with the quality of their representation. These four basic sorts of the abstract data type for proofwebs are reflected in our choice of windows for displaying them. Because tatami pages are the main constituent of proofwebs, theirs is the master window, and because explanation pages are so closely linked, they share that window; each unit is enclosed is its own "box." Tutorial and machine proof score pages each have a separate window. All this preserves the hierarchical structure and priorities of the underlying mathematics.
2. **Backgrounds:** Each major sort of unit has its own background color: proof units have light beige, explanations have light yellow, tutorials have yellow

marble, and proof scores have light purple. Although the choice of colors is somewhat arbitrary, and is easily changed by editing the XSL style file, their distinctness reflects the importance of distinguishing these four units.

3. **Navigation:** Similar considerations hold for navigation. Each page has a title, supplied by the user in the Duck script (or a simple default if no title is supplied). Buttons are used to move to other pages of the same sort, and to open widows that display information of other sorts. Each persistent window has a somewhat different layout and navigation buttons, reflecting its different typical uses. For example, the master tatami window has buttons to step through the narrative ordering of tatami pages, both forward and backward, and a button to return to the homepage.
4. **Mathematical Formulae:** gif files are used for mathematical symbols, in a distinctive blue color, because mathematical signs come from a domain that is quite distinct from that of natural language.

Some additional applications of semiotic morphisms to the user interface design of the Tatami system are described in Goguen (1999b), in a more precise style than here, although they are based on an older version of the system. For example, Goguen (1999b) shows that certain early designs for the status window were incorrect because the corresponding semiotic morphisms failed to preserve certain key constructors.

Humor

We have studied a corpus of over 50 “humorous oxymorons” (phrases like “military intelligence,” “good grief,” and “almost exactly”). Dictionaries say an “oxymoron” is a phrase having contradictory (or incongruous) components. But this is not what happens in a humorous oxymoron: instead, there are two distinct meanings, one of which is conventional, and the other of which has some contradictory components; that is, there are two different blends, one of which has conflicts. When we are told that something is an oxymoron, we seek out that second, conflictual blend, and we feel pleasure when we find it.

We also studied more than 40 newspaper cartoons and found that about 75% have a similar pattern, but instead of two blends existing simultaneously, the reader is first led to form one blend, and then led by new information to form a completely different blend, usually in partial conflict with the first; that is, there is a kind of dynamic reblending.

Thus in each case, it is not just the existence of more than one blend, but rather the process of reblending that produces the humorous effect, and I conjecture that reblending in fact characterizes humor. This is relevant to HCI and the design of virtual worlds, because humor is sometimes used in computer system interfaces, often very badly. For example, the paperclip in Microsoft Office creates a poor impression in part because the sensation of reblending loses its effect if it is repeated many times, and eventually becomes “stale” or even unpleasant (see Goguen, 2004a, for additional details). These observations, which go back to about 1999, seem to have potential for fascinating new application areas.

Interaction

Classical semiotics is concerned with static signs; it does not allow for signs that change in response to user input, or that move on their own. This section sketches how algebraic semiotics handles dynamics, by extending its foundation from classical algebra to hidden algebra. As a simple example, consider the problem of designing that part of the Kumo interface that supports browsing proofs. Kumo provides buttons to traverse in the proof author’s chosen narrative order, labeled with iconic triangles to indicate forward and backward motion, as well as buttons to return to the homepage, to view the specification, and so forth (see Figure 8). Common practice would suggest constructing an automaton with a state for each proof tree node, and a transition label for each traversal button. But this does not allow for the fact that different proofs have different structures, and thus different automata, nor does it account for the different displays that are produced in each state, nor for the variety of possible implementations of transition lookup, for example, using lists, arrays, or hash tables. An automaton can describe how a single proof instance can be navigated, but it cannot describe the general method that generates proof navigation support for any given proof, nor the way that this method is implemented, nor the quality of the resulting interface.

In fact, despite the formal character of the model itself, the construction and use of transition diagrams (or the corresponding automata) in user interface design is intuitive, and does not provide an adequate basis for a rigorous mathematical analysis of possible designs. In order to address the display, implementation, and quality questions raised above, the automaton model must be supplemented in various *ad hoc* ways, whereas hidden algebra can handle all of these within a single unified framework. Another example of dynamics in Kumo that would be difficult to handle with traditional user interface modeling techniques is the facility to execute the proof script for a proof part by downloading it to a BOBJ proof server and then viewing the result on the local browser as it executes.

This is not the place for details (see Goguen, 2003, for that), but we can say that hidden algebra provides a precise way to handle both the display and implementation aspects of examples like that described above, and the corresponding extension of semiotic morphisms gives a precise basis for comparing the quality of interface designs realizing the desired dynamics, without bias towards any particular implementation. The dynamics of a window with a scrollbar is discussed in considerable detail in Goguen (2003).

Summary, Future Research, and Social Implications

This chapter has presented theory and case studies to support the claim that algebraic semiotics is a promising foundation for virtual world design, in both theory and practice. The case studies on information visualization, proof presentation, metaphor, humor, and interaction are encouraging, and suggest that design problems can be successfully confronted directly, without unreliable *ad hoc* methods and assumptions, such as analyses based on prior systems that are only remotely related, or expensive, time-consuming methods of experimental psychology and usability testing. These studies also confirm our views that taking account of key social and cognitive factors is crucial for success, and that formal methods can play a very helpful role, if applied pragmatically rather than dogmatically. However, much more work is still needed, such as:

- Combining Gibsonian affordances (Gibson, 1977) with algebraic semiotics, to provide a socio-cognitive dimension for the interaction formalism.
- Studying immersion in virtual worlds, for example, how closure and embodiment relate to representational coherence, image schemas, affordances, choice of media, and so forth.
- More work on social foundations and the processes of semiosis.
- More work on narrative structure, including flashbacks and flashforwards.
- More work on how to choose quality orderings on representations that are appropriate to their actual use.
- More case studies, done more thoroughly.

Only the second of these is specific to virtuality, though all are related. We hope that readers of this chapter may find some benefit to the algebraic semiotic approach, and will contribute to its further development.

I close this article with some words of warning, along lines perhaps most closely associated with Jean Baudrillard (1994), who wrote:

> *"Simulation is no longer that of a territory, a referential being, or a substance. It is the generation by models of a real without origin or reality…. By crossing into a space…no longer that of the real, nor that of truth, the era of simulation is inaugurated by a liquidation of all referentials—worse: with their artificial resurrection in the systems of signs, a material more malleable than meaning, in that it lends itself to all systems of equivalences, to all binary oppositions, to all combinatory algebra. It is no longer a question of imitation, nor duplication, not even parody. It is a question of substituting the signs of the real for the real, that is to say of an operation of deterring every real process via its operational double, a programmatic, metastable, perfectly descriptive machine that offers all the signs of the real and short-circuits all its vicissitudes. Never again will the real have the chance to produce itself—such is the vital function of the model in a system of death…."*

If we translate this out of the stylistic conventions of recent French intellectualism, the danger is that the virtual can replace the real in our affections, so that we lose touch with our communities, our values, even the very living quality of our lives. Baudrillard claims that exactly such alienation is already characteristic of the contemporary world, and that it is growing like a cancer. He does not offer any solution to this dilemma, but I would like to suggest that compassion (Goguen, 2004b) is one way out of an enervating absorption in virtuality. A sympathetic feeling for the suffering of others, and action on their behalf, can generate positive emotionality and re-engagement with real experience. And, contrary to Baudrillard, it seems quite possible that technology, including virtual world technology, can assist with such projects.

Acknowledgments

I thank Drs Kirstie Bellman and Christopher Landauer of Aerospace Corp. for inviting me to the 2001 Virtual Worlds and Simulation Conference, and for their valuable comments. I thank Fox Harrell for many valuable conversations and insights on topics related to this chapter, and for the renderings of Figures 5, 6,

and 7. I also thank students in my UCSD classes CSE 171 and 271 for their feedback and patience.

References

Aristotle. (1997). *Poetics*. Dover. Translation by S.H. Butcher; original from approximately 330 B.C.

Barthes, R. (1968). *Elements of semiology*. Hill and Wang. Translation by A. Lavers & C. Smith.

Barwise, J., & Perry, J. (1983). *Situations and attitudes*. Bradford: MIT.

Baudrillard, J. (1994). *Simulacra and simulation*. Michigan. Translated by S.F. Glaser.

Campbell, J. (1973). *The hero with a thousand faces.* Princeton.

Carroll, J. (1982). Learning, using, and designing filenames and command paradigms. *Behavior and Information Technology, 1*(4), 327-246.

Fauconnier, G. (1985). *Mental spaces: Aspects of meaning construction in natural language*. Bradford: MIT.

Fauconnier, G., & Turner, M. (1998). Conceptual integration networks. *Cognitive Science*, *22*(2), 133-187.

Fauconnier, G., & Turner, M. (2002). *The way we think*. Basic Books.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*(2), 155-170.

Gibson, J. (1977). The theory of affordances. In R. Shaw & J. Bransford (Eds.), *Perceiving, acting and knowing: Toward an ecological psychology*. Lawrence Erlbaum.

Goguen, J. (1994). Requirements engineering as the reconciliation of social and technical issues. In M. Jirotka & J. Goguen (Eds.), *Requirements engineering: Social and technical issues* (pp. 165-200). Academic.

Goguen, J. (1997). Towards a social, ethical theory of information. In G. Bowker, L. Star, W. Turner, & L. Gasser (Eds.), *Social science, technical systems and cooperative work: Beyond the great divide* (pp. 27-56). Lawrence Erlbaum.

Goguen, J. (1999a). An introduction to algebraic semiotics, with applications to user interface design. In C. Nehaniv (Ed.), *Computation for metaphors, analogy and agents* (pp. 242-291). Springer. *Lecture Notes in Artificial Intelligence, 1562.*

Goguen, J. (1999b). Social and semiotic analyses for theorem prover user interface design. *Formal Aspects of Computing, 11*(Special Issue on User Interfaces for Theorem Provers), 272-301.

Goguen, J. (2003). Semiotic morphisms, representations, and blending for interface design. *Proceedings of the AMAST Workshop on Algebraic Methods in Language Processing* (pp. 1-15), Verona, Italy, August 25-27. AMAST Press.

Goguen, J. (2004a). CSE 275 homepage: Social issues in science and technology. Retrieved from: *www.cs.ucsd.edu/users/goguen/courses/275/*

Goguen, J. (2004b). Groundlessness, compassion and ethics in management and design. In R. Boland & F. Callopy (Eds.), *Managing as designing*. Stanford.

Goguen, J., & Harrell, F. (2003). Information visualization and semiotic morphisms. In G. Malcolm (Ed.), *Visual representations and interpretations*. Proceedings of a workshop held in Liverpool, UK. Elsevier.

Goguen, J., & Harrell, F. (2004). *Foundations for active multimedia narrative: Semiotic spaces and structural blending*. To appear in Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems.

Goguen, J., Lin, K., Rosu, G., Mori, A., & Warinschi, B. (2000). An overview of the Tatami project. In K. Futatsugi, A. Nakagawa, & T. Tamai (Eds.), *Cafe: An industrial-strength algebraic formal method* (pp. 61-78). Elsevier.

Goguen, J., & Linde, C. (1984). *Optimal structures for multi-media instruction*. Technical report, SRI International. To Office of Naval Research, Psychological Sciences Division.

Goguen, J., & Malcolm, G. (1996). *Algebraic semantics of imperative programs*. MIT.

Goguen, J., & Malcolm, G. (2000). A hidden agenda. *Theoretical Computer Science*, *245*(1), 55-101. Also UCSD Dept. Computer Science & Engineering Technical Report CS97-538, May 1997.

Goguen, J., Thatcher, J., & Wagner, E. (1978). An initial algebra approach to the specification, correctness and implementation of abstract data types. In R. Yeh (Ed.), *Current trends in programming methodology IV* (pp. 80-149). Englewood Cliffs, NJ: Prentice-Hall.

Grady, J., Oakley, T., & Coulson, S. (1999). Blending and metaphor. In R. Gibbs & G. Steen (Eds.), *Metaphor in cognitive linguistics*. Benjamins.

Labov, W. (1972). The transformation of experience in narrative syntax. *In Language in the inner city* (pp. 354-396). Philadelphia: University of Pennsylvania.

Lakoff, G. (1987). *Women, fire and other dangerous things: What categories reveal about the mind*. Chicago.

Lakoff, G., & Johnson, M. (1980). *Metaphors we live by*. Chicago.

Linde, C. (1981). The organization of discourse. In T. Shopen & J.M. Williams (Eds.), *Style and variables in English* (pp. 84-114). Winthrop.

Linde, C. (1993). *Life stories: The creation of coherence.* Oxford.

Livingston, E. (1987). *The ethnomethodology of mathematics*. Routledge & Kegan Paul.

Mac Lane, S. (1998). *Categories for the working mathematician (2nd ed.)*. Springer.

Miller, G.A. (1956). The magic number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Science*, *63*, 81-97.

Peirce, C.S. (1965). *Collected papers* (in six volumes; see especially Volume 2: Elements of Logic). Harvard.

Pierce, B.C. (1990). *Basic category theory for computer scientists.* MIT.

Saussure, F. (1976). *Course in general linguistics*. Duckworth. Translated by R. Harris.

Shneiderman, B. (1997). *Designing the user interface (2nd ed.)*. Addison-Wesley.

Shneiderman, B. (1998). *Designing the user interface (3rd ed.)*. Addison-Wesley.

Tufte, E. (1983). *The visual display of quantitative information.* Graphics Press.

Turner, M. (1997). *The literary mind.* Oxford.

Zbikowski, L. (2002). *Conceptualizing music.* Oxford.

Endnotes

[1] Due to the nature of this chapter, sign systems are not fully formalized, and in particular, signatures are treated rather informally, because they are sufficiently complex that a formal definition would distract from the flow of ideas; see Goguen and Malcolm (1996) and Goguen et al. (1978) for the

formal definition of signature, and see Goguen (1999a) for the formal definition of sign system.

[2] These are for fixed data types like integers, Booleans, and colors, which are always interpreted in a standard way.

[3] This use of the word “space” conflicts with the conceptual spaces of cognitive linguistics, which are discussed below.

[4] This duality is a Galois connection between algebraic theories and their models; it does not involve the levels or priorities.

[5] The source UP is grounded in our experience of gravity, and the schema itself is grounded in everyday experiences, such as that when there is more beer in a glass or more peanuts in a pile, the level goes up, and this is a state we often prefer; therefore the image schema MORE IS UP, discussed in Lakoff (1987), is even more basic.

[6] However, we do not assume that they are necessarily the **minimal** such spaces needed to understand a given blend, since that can only be determined after the blend has been understood. Moreover, different blends may ignore different elements of the input spaces, and it may also be necessary to recruit additional information from other spaces in order to understand a blend.

[7] The term “**base space**” is used in Goguen (1999a) because it is considered to better describe how this theory is used in applications to user interface design.

[8] Strict commutativity, usually called just **commutativity**, means that the compositions are strictly equal, that is, one morphism is defined on an element if and only if the other is, and then they are equal.

[9] But as before, gray tones appear in our rendition of the display.

Chapter VI

Conceptual Modeling of Virtual Environments Using Hypermedia Design Techniques

Paloma Díaz
Universidad Carlos III de Madrid, Spain

Susana Montero
Universidad Carlos III de Madrid, Spain

Ignacio Aedo
Universidad Carlos III de Madrid, Spain

Juan Manuel Dodero
Universidad Carlos III de Madrid, Spain

Abstract

Traditionally, the development of virtual environments has been tightly dependent on the programmer's skills to manage the available toolkits and authoring systems. In such a scenario, the discussion of different design alternatives, future changes and maintenance, interoperability, and software reuse are all of them costly and quite difficult. In order to overcome this *unsystematic and technology-driven process, conceptual modeling has to be included just before the implementation phase to provide a shared representation language that facilitates the communication among the different team members, including stakeholders, as well as the reuse and redesign for future requirements since conceptual models hide implementation details and constraints, and are cheaper and easier to produce than prototypes. As a first attempt to attain these aims, this chapter presents the basis of a constructional approach for the VE conceptual modeling through a set of complementary design views related to the VE components and functions. Moreover, we explore how these design issues might be addressed by hypermedia modeling techniques, given the similarities between these two kinds of interactive systems and the maturity reached in hypermedia development.*

Introduction

When facing the development of virtual environments (VEs), most developers turn to toolkits or authoring tools like those reported in Kessler, Bowman, and Hodges (2000) in which they pick different components out of a repository and build their environment from scratch in an unsystematic fashion driven by technology rather than by requirements. In such a scenario, abstraction is absolutely despised. Instead of describing the VE using concepts and relationships that describe the problem to be solved in terms of the universe of discourse (such as rooms, collections, or paintings in a virtual museum) as it is done in conceptual models (Hofstede & van der Weide, 1993), it is expressed using technical terms and implementation units, such as cylinders, spheres, or textures. But such a technology-driven development strategy brings a number of disadvantages. Firstly, development is boiled down to 3D modeling and programming, so that the stakeholders can only take part in the evaluation, whether formative or summative, of prototypes. Involving stakeholders in all the phases of the development process, including design, is a basic requirement for any kind of interactive systems, as they know which objects, facts, concepts, and relationships are relevant in the domain of the application (Preece, Rogers, & Sharp, 2002). Secondly, the lack of a conceptual design process leads to little flexibility for changes with a high cost in resources when the environment does not meet the user requirements or when technological evolution suggests the addition of new services. And finally, this implementation-driven approach makes maintenance, interoperability, and software reuse difficult or nearly unfeasible. A conceptual model, which is independent from the implementation units, provides a picture of the system that can be understood by different people and that can serve as an intermediate level among different technological options. Each concept and relationship in the real world has a correspondence with an element or more in the conceptual model, which in turn can be translated into different implementation platforms.

The scarce use of conceptual models in VE development can be due to the fact that there are no broadly accepted development methodologies and design techniques that apply a software engineering perspective to this domain. A VE software crisis, in the line of the phenomenon identified in the ’60s as the software crisis (Gibbs, 1994) and in the ’90s as the hypermedia/Web software crisis (Lowe & Hall, 1999), can be diagnosed just to try and raise concern for the way most VE software is being produced.

Compared to traditional software systems, VEs make use of a richer variety of complex types of objects, behaviors, interactions, and communications in order to provide users with more and more realistic and useful environments and, consequently, typical conceptual modeling tools, such as E-R diagrams (Chen, 1976) or UML models (Booch, Jacobson, & Rumbaugh, 1998), do not seem appropriate enough to encompass all their requirements. On the other hand, VEs share with hypermedia systems a number of features and problems. Consequently, hypermedia modeling techniques that address the aforementioned design issues and have reached a certain level of maturity may be considered in the development of VEs as well.

In this chapter, the use of hypermedia design techniques for VEs will be discussed in order to:

- Make evident the relationship between the ideas, concepts, and principles underlying different modeling techniques in both disciplines, and
- Analyze how existing hypermedia techniques are suitable to deal with the development process of VEs.

The approach assumed in this chapter has a number of advantages. Firstly, it does not require all the members of the development team to be experts in implementation technologies, insofar as the system is specified at a conceptual level. Secondly, from the resulting conceptual design, rapid prototypes can be generated using any toolkit or authoring tool, so that the system usability can be assessed and modifications and improvements can be made directly on the design and not diving into the final code. As a result, maintenance and reuse of VEs will become easier. Finally, designers can benefit from the experience underlying hypermedia techniques in such aspects as navigation tools definition, interaction and dynamics modeling, or space and time-based relationships.

The remainder of the chapter is organized as follows. The next section discusses VE conceptual modeling and introduces a constructional approach that proposes six complementary design views to deal with the VEs requirements: structure, presentation, behavior, navigation, users, and access. We then analyze the similarities between hypermedia and VEs to justify the application of hypermedia techniques in this domain and describe how to use a specific development method, called ADM, in the conceptual modeling of VEs, highlighting the main benefits of this approach. Then, various related works are surveyed and compared with our approach. Finally, a summary and some considerations for future work are outlined.

Conceptual Modeling of VEs

Most VEs have been developed using both toolkits and authoring systems in unsystematic fashion. Toolkits such as Vortex (cm-labs.com/products/index.php) or MAVERIK (aig.cs.man.ac.uk/maverik/) provide an application programming interface with which a skilled programmer can create VEs from scratch. In authoring tools like SENSE8 (www.sense8.com/index.html) or DIVISION Reality (www.ptc.com/products/), the access to programming libraries is done through a graphical interface with support for user customization. Although they help in the virtual environment construction, basically affording a rapid prototyping process, the usability of the resulting system can be compromised since end-user requirements are sacrificed to get the prototype working according to the capabilities of the development environment. Even an unfortunate choice of a tool can be a critical factor in the success or failure of the final system (Smith & Duke, 2000). Concerning the development process itself, when all design decisions are taken in the implementation phase and hidden into the code, usability, maintainability, and reusability are compromised. There is not room for the discussion of different design alternatives; any change or future maintenance has to be made diving into the final code, and the success of the system depends only on the skill of the developer to manage the toolkit. In order to provide a more flexible development process, a high-level modeling phase, proposing a series of mechanisms to express the system features in an abstract and complete way, can be introduced just before the implementation phase, so that implementation details are hidden in a first specification stage. This approach enriches the technology-driven development method in several ways, including: VE development is no more constrained by a particular toolkit; reuse and redesign for future designs are cheaper and easier to afford; it can be verified if the VE meets the requirements before implementing the system or prototype; and there is a shared representation language (the conceptual model) that will facilitate the communication among the different team members, including stakeholders.

There still is little knowledge about how VEs are designed, what issues have to be addressed, and, what is worse, little guidance about how design should be carried out. Some works have been aimed at these issues, but in this chapter we will focus on the ones that deal with a very specific question related to the goals of our proposal: *Which different design perspectives or products have to be considered in the conceptual modeling of VEs?*

Kim, Kang, Kim, and Lee (1998) state three different perspectives needed for the VE modeling:

- *Form* determines the appearance of virtual objects and the scene structure of the virtual world.
- *Function* encodes what virtual objects do, whether autonomously or in response to external stimuli.
- *Behavior* determines how virtual objects react to events.

From the point of view of Smith and Duke (2000), five views make up VE design:

- *Object component decomposition* identifies the objects that are required in a VE. The output is a tree structure showing the decomposition of virtual object and any associated behavior.
- *Object appearance* determines the level of rendering for each of the objects of an environment according to their realism.
- *Object behavior* defines the functions and triggers that cause the object to react to events.
- *Embodiment* determines the virtual representation of the user and tasks that she can carry out in the environment.
- *Navigation* identifies the possible paths of transitional or orientational navigation for the user, including navigation aids.

Finally, Tanriverdi and Jacob (2001) present a methodology based on a design model for developing reality interfaces. In the high-level design phase of the methodology, they take into account the following design aspects:

- *Graphics* specify a high-level description of graphic requirements for virtual objects.
- *Behaviors* identify object behaviors, classifying them into simple physical, simple magical, or composite behavior categories.
- *Interactions* identify interaction requests to objects and behavioral changes caused by these requests.

- *Internal communications* specify control and coordination among the components of objects.
- *External communications* specify control and coordination among objects.

In the following section we thoroughly discuss the inclusion of a conceptual modeling phase into the traditional development process of VEs. We will consider conceptual modeling in a broad sense as a mechanism to specify all the features of the system, both static and dynamic, and not just as a data modeling technique. In this sense, we will consider a set of complementary design views that can help designers acquire a better understanding of the skills required to develop VEs and the issues that need to be addressed in the development of this kind of interactive systems.

Conceptual Modeling as a Software Developing Technique

Conceptual modeling consists of applying a high level of abstraction to describe the static and dynamics of software system, so that designers are compelled to translate the application requirements into logical solutions not biased by technical issues. This property, usually referred to as *Conceptualization Principle* (Hofstede & van der Weide, 1993), puts the stress on the need to produce platform-independent design entities, assuming the definition of design entity given in IEEE (1990), that are used to gather the characteristics and dynamics of the universe of discourse as well as the system requirements.

Abstraction is a key activity in software development and in computer science in general, but it is also a quite typical problem. Technical details and constraints often blur design solutions, making them difficult to understand for a multidisciplinary audience and, what is worse, difficult to reuse and evaluate. In the case of interactive systems and VEs, this situation worsens with the massive use of rapid prototyping techniques that tend to shift the user's attention from the system structure and services to interface features. Moreover, the high cost of developing VE prototypes often results in little flexibility to changes and poor quality and usability. However, as usability is considered one of the most critical quality criteria of any interactive system, the system success will heavily depend on how the VE system components and tasks are represented (Stanney, Mollaghasemi, Reeves, Breaux, & Graeber, 2003).

Conceptual modeling has been massively used in software design since it provides a number of products readable enough to analyze the structure and function of a system, even when the development team is made up of people with different backgrounds, as happens in VEs. A conceptual model of a specific VE will represent virtual spaces, objects, behaviors, and services using concepts that are widely accepted from the users' perspective and, at the same time, that are independent from any implementation platform. Conceptual modeling is then a powerful abstraction tool that helps in disregarding irrelevant structures by building relationships between idealized concepts that focus on what is essential to produce a specification of a system that describes how the system is and how it should work. Thus, the conceptual model of a VE will concentrate on relevant entities, relationships, and behaviors, and will not take into account the computational technology used to implement the virtual environment, such as hardware (i.e., computer architectures), operating systems, input (i.e., HDMs or gloves), and output devices (i.e., visual or tactile displays).

When the development of an interactive system is addressed, whether a VE or any other kind of interactive system, two points of view can be adopted: *behavioral*, taking upon the user perspective and the interaction with the software application, and *constructional*, turning over the software developer view and the software system design (Gabbard, Hix, & Swan, 1999). In the next section, we present the basis of a constructional approach for the conceptual design of VEs.

A Constructional Approach for VE Conceptual Modeling

A VE can be conceptualized in terms of the following components and functions defined in Díaz and Fernández (2000): (1) *a virtual space*: most VEs are built upon a spatial metaphor such as a building or a city; (2) *inhabitants*: objects are populated within the virtual environment space; (3) *user embodiments*: users have a body image representation as avatars or software agents; (4*) mobility*: users and objects browse through virtual spaces; (5) *behavior:* users interact with the virtual environment.

In order to specify all these static and dynamic features and requirements in a progressive and integrated way, complementary design perspectives related to the VE components and functionalities have to be managed in the conceptual modeling process. In particular, in the constructional approach here introduced, where HCI issues will be considered as a cornerstone, we propose six design views: structural representation of the domain, virtual objects presentation, virtual objects behavior, navigation through the VE, user modeling, and VE access. These design views, their underlying requirements, and other relevant issues concerning VE design are discussed in the next paragraphs.

Structural Representation of the Domain

A VE is the representation in a computer of a particular domain, whether real or not, for instance, a virtual art museum where a user can browse its rooms and watch its collections following different exploration styles. Such a domain has an underlying structure that is expressed in terms of *concepts* or *"things"* (e.g., the entities of an E-R model or the classes and meta-classes of an object-oriented UML model) and *structural relationships* (e.g., the relationships among E-R entities or the structural relations and associations in an UML model), producing what are usually known as data models. Thus, the virtual art museum can be conceptually structured as a collection of rooms that in turn are composed of a number of artworks performed by authors who are often organized into schools or periods. Moreover, *things* have properties or attributes (Wand, Storey, & Weber, 1999), depending on whether they are concrete *things* or conceptual ones, which have to be incorporated to their model. Properties can be intrinsic, when they depend on just one thing (for example, a workart has a type—such as painting, sculpture—and a date); or mutual/relational, when they depend on two or more things (for example, the workart inclusion in a specific school depends on the existence of both a workart and a school). In addition, properties and attributes may have an explicit value (e.g., workart dating) or a calculated one (e.g., workart number of visits).

Although VEs are traditionally designed just taking into account the spatial structure of virtual spaces, this data modeling perspective can help to analyze and acquire a deeper knowledge on the domain to be represented. Different data modeling techniques have been proposed in the literature to capture the structural features of software systems, such as E-R model, semantic modeling, or object-oriented approaches, whose expressiveness chiefly depends on the richness and semantics of their constructs. Indeed, the study of the kinds of relationships among entities has given place to several works, such as Wand et al. (1999), where conceptual models are analyzed from an ontological perspective, or Welty and Guarino (2001), who explore different ways of defining taxonomies according to the actual mereological or parthood relationships. Concerning VE structural modeling, some authors like Xiaoguang, Dongmu, and Bingrong (1999) propose the use of an object-oriented approach for data modeling, while De Troyer, Bille, Romero, and Stuer (2003) make use of ontologies to describe the virtual world domain. Either one technique or another is applied, the important thing is that all these approaches provide a high-level description of the application domain using well-known terms that can be understood by both domain experts and stakeholders, so that they can actively collaborate with developers, from the very beginning taking part in a user-centered process oriented towards enhancing usability.

Virtual Objects Presentation

Virtual objects are placed in the structures to make up the virtual world. The appearance of such virtual objects according to their geometric structures and the rendering level determines the realism of a VE. This information may be too concrete as to be specified during a conceptual design. However, the purpose of conceptual modeling of the virtual objects presentation is to organize and harmonize each virtual object and its different components in different dimensions, such as the time and the two- or three-dimensional space, to produce a high-level picture of the VE rendering features. The goal is not to produce a 3D prototype, but a sketch representing the world or part of it that can be discussed with users to improve design decisions before implementing the system. The same as it is done in hypermedia presentation modeling (Díaz, Aedo, & Montero, 2001b), designers can perform the following activities maintaining a reasonable level of abstraction: creating templates or interface mock-ups by placing abstract objects into spaces; defining visual clues or human-computer interaction rules to increase the system usability; and setting space and time-based relationships among components to create aesthetic and dynamic multimedia compositions. Thus, within a room of a museum, workarts can be placed as blackboxes into the visualization and interaction space, to indicate a relative location and orientation independent of users' viewpoints.

Virtual Objects Behavior

Another important and critical issue in the conceptual modeling of virtual environments is the specification of the interaction and behavior, insofar as these systems are intrinsically interactive, and the high level of realism of their appearance leads users to a high expectation about the system fidelity with virtual object behavior. For example, a door that has been rendered with realistic textures gives the feeling that it can be opened and closed.

Therefore behaviors, whether reactive, interactive, or proactive, should be considered during conceptual modeling. In such behaviors, one or more agents can be involved, assuming as agent any object or user that can initiate a dynamic process, giving place to any kind of result affecting to the environment and/or interacting with the user. Taking into account the involved agents, we can identify two types of behaviors: intrinsic behaviors, depending only on one object (for example, when the attribute "number of visits" of a workart is higher than 100, it is marked with a visual clue), and mutual behaviors, depending on two or more agents (for example, whenever a user visits a workart, the attribute "number of visits" of that workart is increased). The latter can be further divided into user-object behavior (as the example shown before) and object-object behavior (for example, if two moving objects meet a “crash” object will be shown).

Interaction modeling can be tackled from a conceptual point of view, creating high-level descriptions of the system reactions and completing their specifications in a further detailed design. This design perspective makes it possible to model complex behaviors, including interactive behaviors, reactions, proactive actions, access to external applications (e.g., inference engines to support adaptive interfaces, databases to create objects dynamically), or the inclusion of virtual objects and structures that are created or modified at runtime (Díaz, Aedo, & Panetsos, 2001a). The advantages of this approach are twofold. First, complex behaviors can be broken down into more simple ones, providing a library of behaviors that designers can reuse to create new behaviors. Second, this high-level description enables designers to assess the level of detail required in communicating the behavior specification to software developers being programming language independent.

Some approaches to model interaction and behavior include the use of state machines (Tanriverdi & Jacob, 2001) or DFDs and statecharts (Kim et al., 1998).

Navigation Through the Virtual Environment

Unfamiliar and large-scale virtual environments are difficult to navigate. There are two key aspects concerning the process of navigating or browsing the virtual space. One is wayfinding, which consists of determining a path within the environment which satisfies the user expectations and needs. The other one is motion control, which is much more dependent on the user interface provided by the virtual environment software (Volbracht & Domik, 2000). In this work, we will focus on the first issue, since the second one is a technological and not conceptual aspect.

An appropriate spatial structure design describing the relationships among virtual spaces can help users to identify and locate objects. But navigation modeling has to deal not only with the conventional navigation through a continuous 3D space, but also with what is called discontinuous movements (or hyperlinks) that allow a user to jump from one location to another one which is not related spatially, in order to reduce the distance and the navigation time (Ruddle, Howes, Payne, & Jones, 2000). For example, from a workart in the museum, we can move to pieces of the same author, school, style, or theme, for which we will be offered different links.

The use of these associative hyperlinks can derive on disorientation problems as it happens in most hypermedia systems where this kind of semantic movement represents the basic information access paradigm. To try and avoid disorientation, VEs can benefit from the experience in hypermedia navigation modeling (Nielsen, 1995). In fact, they can incorporate appropriate visual clues and navigational aids, such as maps, indexes, footprints, search engines, and so on. Moreover, they can facilitate users' spatial awareness, permitting them to apply their real-world navigational experience. For instance, Vinson (1999) presents guidelines for the design and placement of landmarks in VEs.

User Modeling

There is a key feature that makes VEs different from any other software application, that is the user embodiment, which will play an important role in the realism and usefulness of the system. This issue involves different aspects such as the degree of immersiveness that the user experiments with depending on its physical representation by means of avatars, the availability and degree of presence, the location, the identity, the activity, viewpoints and actionpoints, gesture and facial expression, history of activity, representation across multiple media, autonomous and distributed body parts, efficiency, manipulating one's view of other people, and truthfulness (Benford, Bowers, Fahlén, Greenhalgh, & Snowdon, 1995). Most of these issues are technology-dependent so that they go beyond the scope of conceptual modeling. However, we consider avatars as virtual objects representing the participants in the virtual environment, and, consequently, considerations about virtual objects' presentations and behavior should be taken into account.

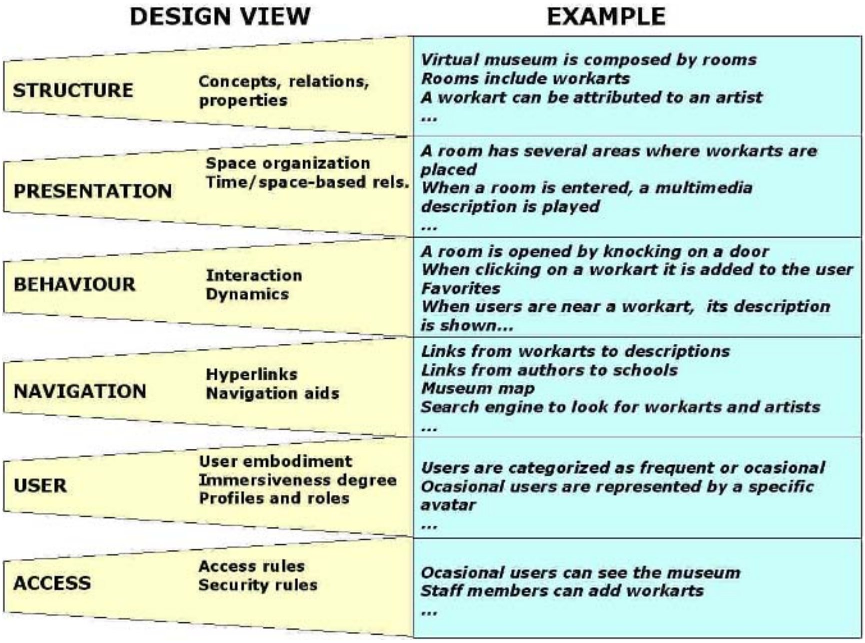
Moreover, we will point out other issues such as access needs and preferences that different kinds of users may have, involving both private objects and spaces. In VEs, there is a need to model the user profile with all of her abilities such as to collect, to move, or to own objects.

In addition, in cooperative or collaborative virtual environments (CVEs) such as MASSIVE (Greenhalgh & Benford, 1995) or DIVE (Hagsand, 1996), there is a need to model groups of users to assist them in communicating, in collaborating, and in coordinating their activities.

Virtual Environment Access

Access is an essential requirement in most multi-user information systems inasmuch as different users have different responsibilities, needs, and preferences that determine their ability to access information. Thus, stereotypes or

Figure 1. Design views in conceptual modeling of VEs



DESIGN VIEW		EXAMPLE
STRUCTURE	Concepts, relations, properties	*Virtual museum is composed by rooms* *Rooms include workarts* *A workart can be attributed to an artist* *...*
PRESENTATION	Space organization Time/space-based rels.	*A room has several areas where workarts are placed* *When a room is entered, a multimedia description is played* *...*
BEHAVIOUR	Interaction Dynamics	*A room is opened by knocking on a door* *When clicking on a workart it is added to the user Favorites* *When users are near a workart, its description is shown...*
NAVIGATION	Hyperlinks Navigation aids	*Links from workarts to descriptions* *Links from authors to schools* *Museum map* *Search engine to look for workarts and artists* *...*
USER	User embodiment Immersiveness degree Profiles and roles	*Users are categorized as frequent or ocasional* *Ocasional users are represented by a specific avatar* *...*
ACCESS	Access rules Security rules	*Ocasional users can see the museum* *Staff members can add workarts* *...*

user profiles are used to provide a kind of personalized access in large information spaces, so that disorientation problem falls off as far as users face up a reduced and supposedly familiar space.

Another issue that has to be considered is information security, taking for granted that security is not only related to confidentiality or privacy, as commonly believed, but also to integrity and availability (Aedo, Díaz, & Montero, 2003). For example, some authors point out that in a CVE there is sensitive information that should be protected (Pettifer & Marsh, 2001). For instance, for each shared object, control rights to manage the object have to be authorized.

As a summary of this section, Figure 1 illustrates how the six design views tackle with different but complementary aspects of a same system.

Using Hypermedia Techniques for the Conceptual Modeling of VEs

Hypermedia provides mechanisms for structuring and navigating large quantities of multimedia and highly interactive information. There are a number of similarities between VEs and hypermedia systems that can be exploited to apply hypermedia design methods into the virtual environments arena.

Hypermedia systems organize information as a net of interrelated nodes which hold multimedia content. These nodes can be freely browsed by users selecting links and making use of other advanced navigation tools, such as indexes or maps (Nielsen, 1995). In a similar way, VEs can be characterized by a net of interrelated “rooms” or “virtual spaces,” which hold virtual objects, where users can move freely. Moreover, VEs and hypermedia environments have to deal with interactive behaviors, such as responses to specific events that occur, as well as to include complex multimedia compositions that have to be usable and aesthetic at the same time. For readability shake, Table 1 includes the definitions of the different hypermedia components that will be referred to in this chapter and which are based on the Labyrinth hypermedia reference model (Díaz et al., 1997, 2001a).

With regard to typical problems, hypermedia users, the same as VE users, suffer from the same well-known navigation problem: the users’ lack of knowledge about their relative position, the disorientation into the system, and a general lack of familiarity with the system operation (Conklin, 1987; Ruddle, Payne, & Jones, 1998).

Most differences between hypermedia systems and VEs concern technological issues, since from a conceptual perspective we can identify equivalences among

Table 1. Hypermedia components

Component	**Description**
Simple node	Abstract container of information
Composite node	Node made up of other simple or composite nodes according to a specific structural relationship
Multimedia content	Information item
Composite content	Content made up of other simple or composite contents according to a specific structural relationship
Structural relationship	Relation settled upon a composite and its components
Location	Position of a content into a node
Space-based constraint	Relative spatial position of a content depending on the position of other content
Time-based constraint	Relative temporal position of a content depending on the position of other content
Link	Navigational connection defined between two sets of source and target anchors
Anchor	Reference to a part of a node, content, or contextual content (a content presented in a node) used to set up links
Attribute	Property that can be assigned to a user, node, content, or link to add useful meta-data
Event	Set of actions executed when a condition is fulfilled (e.g., the mouse is over a content or a timeout expires); events are used to model interactive behaviors (e.g., pop-up windows, simulations) and to include procedural specifications of the hyperdocument elements (e.g., adaptive links)
User	Individual or group of users (profiles, stereotypes, or collaborative group) who access the system under certain conditions

Table 2. Conceptual components in hypermedia and virtual environments

Conceptual Elements	**Hypermedia Elements**	**Virtual Elements**
Simple composite class or type	Simple node	Virtual spaces
	Composite node	
Simple composite thing	Multimedia content	Virtual object User embodiment
	Composite content	Virtual composite object
Relationship and mutual properties	Structural relationship	Structural relationship
	Location	Spatial structure
	Space-based constraint	
	Time-based constraint	
Intrinsic properties	Attribute	Attribute
Navigational relationship	Link	Link
	Anchor	Component part
Derived properties, things, and relations Interactive, reactive, and proactive behaviors	Event	Virtual object behavior
User and groups	User and groups	User and groups

concepts and constructs in both domains. Table 2 summarizes the different conceptual components in a domain and its interpretation in hypermedia and VE.

Any application domain has simple or composite classes or types of *things* (Wand et al., 1999) which turn into simple/composite nodes in hypermedia and virtual spaces or worlds into the VE. For example, a virtual museum is converted into a hypermedia composite node, which will aggregate the different components of the system.

There are also *things* that appear in the domain (Wand et al., 1999), which are the objects and the users' embodiments of VEs and the hypermedia contents, whether simple or composite according to their nature. For example, Albrecht Dürer's "Self-Portrait" or Velazquez's painting "The Surrender of Breda" are *things* that appear in the Prado Museum in Madrid. If we consider just the paintings, they will be treated as simple objects, but if we add a description of the workart, this aggregation will be a composite object.

Structural relationships among *things* (Wand et al., 1999) appear in both domains at an abstract level. Mutual properties not related with navigation, which is considered as a special case, are gathered in the spatial structure of the VE and in several relationships in the hypermedia domain (location, spatial, and time-based constraints).

Navigation relationships are considered in hypermedia and VEs through links and anchors, the latter referring to a virtual component or part of it in a virtual world.

Dynamics, in the form of derived or calculated components and all kinds of behaviors, are represented by means of events in hypermedia which can be triggered by user-dependent or system-dependent conditions. This powerful specification mechanism, thoroughly discussed in Díaz et al. (2001a), can be translated to VEs to model the virtual objects behavior.

Finally, the existence of users and groups has been considered in hypermedia systems from the very beginning since they were envisaged as cooperation enabling tools. Similarly, users have to be considered in VEs, not only for user embodiment, which is a special case of multimedia object, but as an entity whose goals, preferences, needs, or responsibilities have to be analyzed to determine the access capabilities of each kind of user.

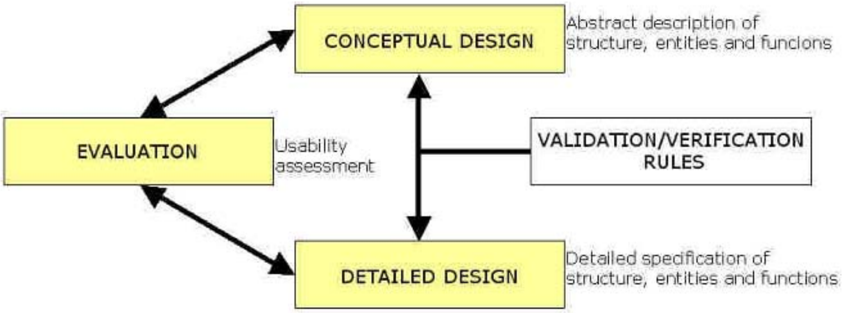
Given these similarities between these two kinds of interactive systems, we propose to apply hypermedia tools to deal with VE conceptual modeling, since hypermedia techniques are mature enough to contribute in the development process. In particular in this chapter, we suggest the use of Ariadne Development Method (ADM; Díaz et al., 2001b) to provide conceptual and methodological guidance to VE designers.

Overview of ADM

ADM proposes a systematic, integrative, and platform-independent process to specify and produce hypermedia and Web applications. In order to cover the modeling process of hypermedia and Web applications, three phases are proposed—conceptual design, detailed design, and evaluation—as shown in Figure 2.

Conceptual design is focused on identifying abstract types of components, relationships, and functions; *Detailed design* is concerned with specifying the system features, processes, and behaviors in a so detailed way that the application might be semi-automatically generated; and, finally, *Evaluation* is concerned with using prototypes and specifications to assess the system usability. Arrows in Figure 2 represent relationships among phases and do not represent any kind of sequence among them. ADM does not impose a rigid process model, letting developers decide the best way to face their work according to their needs. Moreover, the method provides a number of *validation and integrity rules*, both at the intra and inter phase level, to check completeness, consistency, and integrity among the various design products. All of these features have as foundation the Labyrinth reference model (Díaz et al., 1997, 2001a) that provides the core components of any hypermedia application and are supported by a design toolkit called AriadneTool (Montero, Díaz, & Aedo, 2003).

Figure 2. The ADM process model

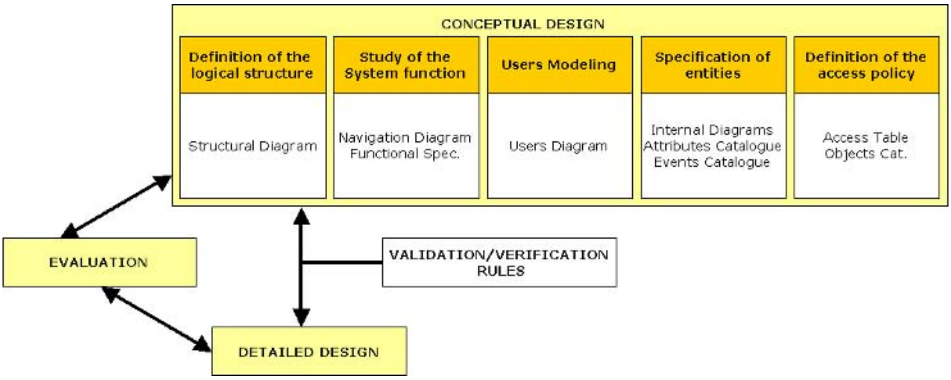

CONCEPTUAL DESIGN
Abstract description of structure, entities and funcions
EVALUATION
Usability assessment
VALIDATION/VERIFICATION RULES
DETAILED DESIGN
Detailed specification of structure, entities and functions

ADM fits the incremental, iterative, and user-centered nature of the VE development process. Taking into account the scope of this chapter, we will only focus on how the conceptual design phase of ADM can be accommodated in the design of virtual spaces, the definition of objects, users and their functionality and privileges, understanding the need for specific mechanisms that deal with their detailed design (e.g., 3D visual space, a multi-modal interface, and an immersive environment), and evaluation (e.g., usability criteria (Hix & Gabbard, 2001).

Modeling VEs Using ADM Conceptual Design Phase

ADM conceptual design approaches development from a high level of abstraction where solutions are expressed in terms of expected types of elements. The activities performed in this phase, as well as the design products generated, are summarized in Figure 3.

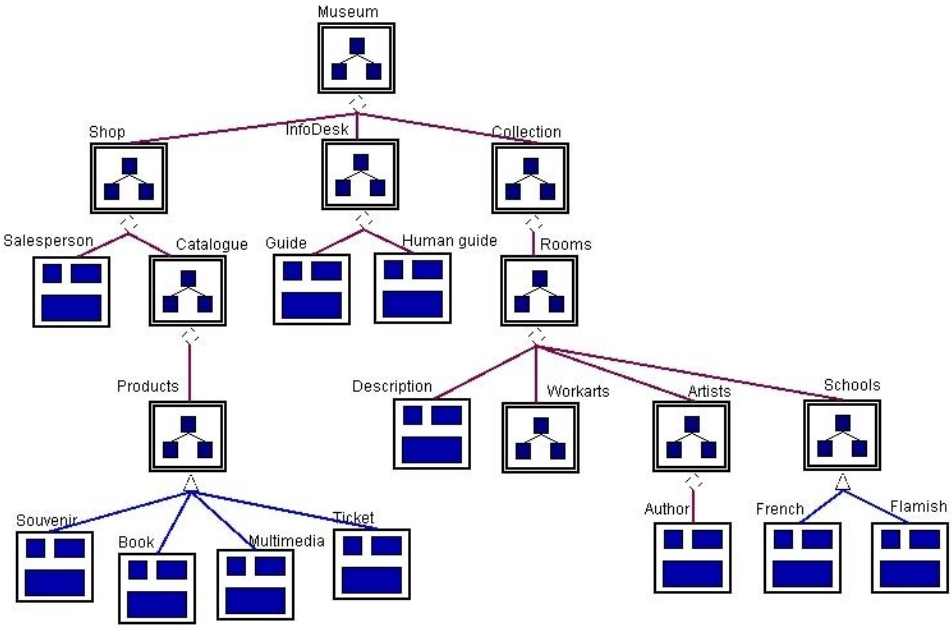
Figure 3. The ADM conceptual design phase



Definition of the Logical Structure

VEs represent a particular domain, whether real or not, such as a virtual museum where a user can explore its rooms and watch its collections from different styles. These structural relationships that appear in the domain of the application can be represented by means of composite nodes that are connected to their components (simple or composite nodes) by means of two possible structural relationships: aggregation, which is a composition relation used as a mechanism to refer to a set of nodes as a whole; and generalization, that represents an inclusion relation involving inheritance mechanisms. For example, Figure 4 shows a possible structural diagram created during this activity for a virtual museum using aggregations and generalizations to represent a complex structure. Thus, the museum aggregates a shop, an information desk, and the collection. In turn the collection is made up of a number of rooms, each of which holds a description, an area where workarts are shown, and information on the corresponding authors and schools. The school can be specialized into particular cases such as the French or the Flamish Schools. As can be seen in Figure 4, this schematic representation is understandable enough to be assessed with stakeholders, and it gathers the environment structure. Objects populating the world will be defined in the "Specification of Entities" activity (below), so that a progressive level of detail is supported.

Figure 4. ADM structural diagram for a virtual museum

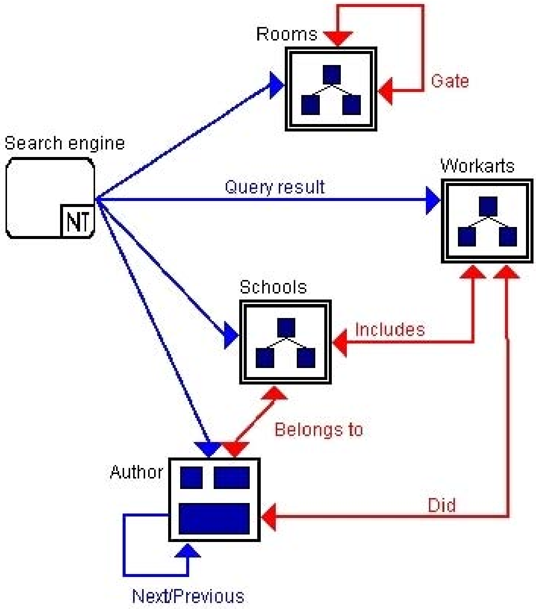


Study of the System Function

This activity is oriented towards describing the different functions offered to the users. Inside the VE, users can browse through different rooms, which can be connected by gates and interact with objects. Moreover, users of virtual environments, like hypermedia ones, have to maintain knowledge of their location and orientation; thus, navigation aids are needed. Therefore, two different kinds of functions are specified: the navigation options offered to the users are represented in a navigation diagram, including both links and other navigation aids (for instance, the navigation structure in the virtual museum is made up of each path connecting two rooms that can be followed by users); and information concerning other services, such as searching for a specific painting or chatting with other users, is documented in the functional specifications.

Figure 5 shows an example of navigation modeling for the virtual museum. The use of hypermedia techniques makes it possible to deal with special kinds of useful links, including: n-ary links (see the connection among the search engine and the possible destinations), bi-directional links (see links "Gate," "Includes," "Belong to," or "Did" in the figure), or reflexive links (see links "Gate" or "Next/Previous"), improving the browsing capabilities of the system.

Figure 5. Part of the ADM navigation diagram of a virtual museum



Specification of Entities

In this activity nodes are specified including their contents (virtual objects), semantics (attributes or properties), and behavior (event-based specifications). Thus, the different spaces identified in the structural diagram are now composed of a number of objects (the components of the virtual world) using the internal diagrams. For example, the internal diagram of the node "Workart" in Figure 4 will contain a number of generic workarts organized in the presentation and visualization space in a specific way. The use of ADM makes it possible to place objects not only in the space but also in a timeline. For instance, we can decide to place a textual explanation in the hole assigned to a workart and a period of time after displaying the painting itself. Moreover, not only absolute positions can be defined as in Usaka, Yura, Fujimori, Mori, and Sakamuram (1998), but also space and time-based constraints can be expressed. For example, we can establish that two paintings on a wall are aligned by their tops by means of an alignment (Díaz et al., 2001a) instead of calculating their (*x*,*y*,*z*) position. Similarly, we could define that when entering a room, a video describing the contents is shown and as soon as it finishes the virtual workarts scene is displayed.

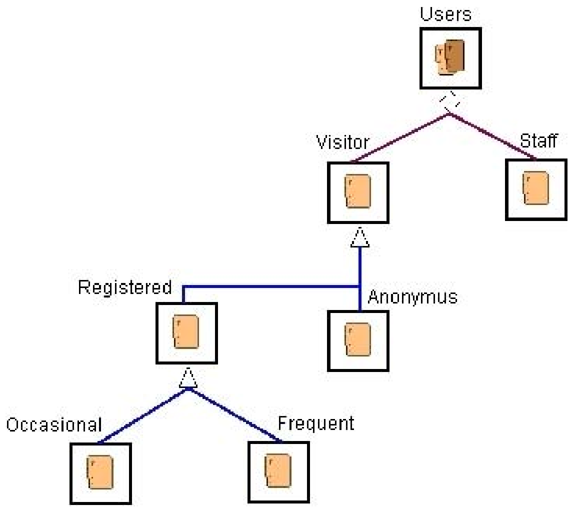
A special case of objects are user embodiments. Each avatar will have an internal diagram that will be associated to a specific kind of user (see next activity) so that this object will be placed at runtime according to the position of the corresponding user.

Moreover, attributes or properties that will increase the node, and contents semantics and events that will model its behavior are defined in the attributes and events catalogue respectively. For example, all the objects representing an avatar will be associated with the same event that puts that object into the right position according to the user's location in the world. Thus, the same event can be reused to model the same behavior. An event can be specified using any process modeling technique, including statecharts, DFDs, and UML activity and sequence diagrams.

User Modeling

This activity is devoted to identifying the expected types of users of the system. In VEs, there is a need to model the user profile with all of her abilities such as to collect, to move, or to own objects. Moreover, in CVEs there is a need to model groups of users to assist them in communicating, in collaborating, and in coordinating their activities. To model the user structure, an RBAC model is assumed (Aedo et al., 2003), which means that roles (stereotypes or responsibilities) and teams (group of roles) are identified in the user diagram.

Figure 6. ADM user diagram for a virtual museum



As an example, Figure 6 shows an ADM user diagram for a virtual museum. The set of users is represented as a team made up of two kinds of people: visitors and staff. In turn, visitors are specialized into other roles to offer more services to registered users. This product makes it possible to analyze the different kinds of users of the system to achieve a deeper understanding of their needs and abilities to access the virtual environment and its components.

Definition of the Access Policy

This step is intended for defining which actions are permitted for each subject, that is, for each role and team defined in the user diagram. With this purpose an RBAC model for hypermedia is assumed (Aedo et al., 2003). According to this model, access rights are assigned in terms of hypermedia objects (nodes and contents) and subjects (teams and roles). Access rights include the ability to see, personalize, and edit.

In the case of a virtual museum, this activity can be used to describe adaptative virtual environments or security rules. For example, using the structural diagram of Figure 4 and the user diagram of Figure 6, it can be established that when a *frequent* user accesses the system, she will be presented with her list of favorites. An access rule can establish that staff members can modify the workarts included in a room or that frequent users can add their own paths into the virtual world.

Conclusions and Future Trends

This chapter has proposed and described a set of integrated design views to set up the basis of a constructional approach for the VE conceptual modeling, and how hypermedia modeling techniques can help to provide mechanisms for structuring and navigating large quantities of multimedia and highly interactive information in the VE arena. Therefore, stakeholders, together with designers and developers, can describe the main specifications of the system using a high-level description, so that immediately afterward specified elements might be semi-automatically generated in a straightforward way to produce prototypes. Moreover, these prototypes can be used to enhance the system usability in terms of a number of well-defined criteria (Hix & Gabbard, 2001) as a way of improving the design, whether conceptual or detailed.

Concerning the design views, a key difference between our approach and other works (Kim et al., 1998; Smith & Duke, 2000; Tanriverdi & Jacob, 2001) is that they primarily focus on behavior and interaction issues, whereas our scope is much wider. Firstly, structural representation and navigation are issues that should be considered to produce useful environments since they are concerned with how the application domain is organized and how that domain is explored, respectively. Secondly, we have not considered the division between interaction and behavior since both issues can be modeled using the same conceptual mechanisms. And thirdly, as virtual environments can be collaborative or simply stereotypes, or user profiles can be required, the need for establishing access rules is a relevant issue addressed in ADM.

That is not to say that only hypermedia modeling techniques should be used for the VE conceptual modeling, but designers should take advantage of the experience gained in years of research in the design of interactive systems, the same as hypermedia engineering made with software engineering (Lowe & Hall, 1999).

Moreover, many research and development issues are still open:

- Develop conceptual models whose components describe the problem domain in terms of virtual components.
- Specify a number of stages and products that make it possible to guide the development process in a systematic and integrated way. As a result, systems will have better quality, usability, maintainability, and reusability.
- Merge the conceptual design of VEs with current toolkits in order to generate the code automatically, as well as documentation about the system development.
- Enhance the design process with the use of design patterns for virtual environments (Díaz & Fernández, 2000). Design patterns capture knowledge of how and when to apply the solution to a recurring problem, besides providing

a shared vocabulary for expressing and communicating such a knowledge. This kind of pattern could help to match design models with implementations.

Acknowledgment

The work presented here is part of the Ariadne project (TIC2000-0402) funded by MyCT (Spain).

References

Aedo, I. , Díaz, P., & Montero, S. (2003). A methodological approach for hypermedia security modeling. *Information and Software Technology, 45*(5), 229-239.

Benford, S., Bowers, J., Fahlén, L.E., Greenhalgh, C., & Snowdon, D. (1995). User embodiment in collaborative virtual environments. *Proceedings of CHI 95: Human Factors in Computing Systems* (pp, 242-249). ACM Press.

Booch, G., Jacobson, I., & Rumbaugh, J. (1998). *The Unified Modeling Language.* Addison-Wesley.

Chen, P. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36.

Conklin, J. (1987). Hypertext: An introduction and survey. *IEEE Computer, 20*(9), 17-41.

De Troyer, O., Bille, W., Romero, R., & Stuer, P. (2003). On generating virtual worlds from domain ontologies. *Proceedings of the 9th International Conference on Multi-Media Modeling* (pp. 279-294).

Díaz, P., Aedo, I., & Panetsos, F. (1997). Labyrinth, an abstract model for hypermedia applications. Description of its static components. *Information Systems, 22*(8), 447-464.

Díaz, P., Aedo, I., & Panetsos, F. (2001a). Modeling the dynamic behavior of hypermedia applications. *IEEE Transactions on Software Engineering, 27*(6), 550-572.

Díaz, P., Aedo, I., & Montero, S. (2001b). Ariadne: A development method for hypermedia. *Proceedings of Dexa 2001.* Berlin: Springer Verlag (LNCS 2113, 764-774).

Díaz, A., & Fernández, A. (2000). A pattern language for virtual environments. *Journal of Network and Computer Applications, 23*(3), 291-309.

Gabbard, J.L., Hix, D., & Swan, J.E. (1999). User-centered design and evaluation of virtual environment. *IEEE Journal of Computer Graphics & Applications, 19*(6), 51-59.

Gibbs, W.W. (1994). Software's chronic crisis. *Scientific American,* 72-81.

Greenhalgh, C., & Benford, S. (1995). MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction, 2*(3), 239-261.

Hagsand, O. (1996). Interactive multiuser VEs in the DIVE system. *IEEE Multimedia*, *3*(1), 30-39.

Hix, D., & Gabbard, J.L. (2001). Usability engineering of virtual environments. Chapter 39 in K. Stanney (Ed.), *Handbook of virtual environments: Design, implementation, and applications.* Publisher.

Hofstede, A.H.M., & van der Weide, Th.P. (1993). Expressiveness in conceptual data modeling. *Data and Knowledge Engineering, 10,* 65-100.

IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990.

Kessler, G., Bowman, D., & Hodges, L. (2000). The simple virtual environment library: An extensible framework for building VE applications. *Presence: Teleoperators and Virtual Environments, 9*(2), 187-208.

Kim, G.J., Kang, K.C., Kim, H., & Lee, J. (1998). Software engineering of virtual worlds. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 131-138).

Lowe, D., & Hall, W. (1999). *Hypermedia and the Web: An engineering approach.* New York: John Wiley & Sons.

Montero, S., Díaz, P., & Aedo, I. (2003). A design toolkit for hypermedia applications. *Proceedings of Web Engineering* (ICWE 2003). Berlin: Springer Verlag (LNCS 2722, 214-217).

Pettifer, S., & Marsh, J. (2001). Collaborative access model for shared virtual environments. *Proceedings of International 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WET ICE 2001) (pp. 257-262).

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design: Beyond human computer interaction.* New York: John Wiley & Sons.

Rolland, C., & Prakash, N. (2000). From conceptual modeling to requirements engineering. *Annals of Software Engineering, 10,* 151-176.

Ruddle, R., Payne, S., & Jones, D. (1998). Navigating large-scale “desk-top” virtual buildings. *Presence, 7*(2), 179-192.

Ruddle, R.A., Howes, A., Payne, S. J., & Jones, D. (2000). The effects of hyperlinks on navigation in virtual environments. *International Journal of Human-Computer Studies, 53*(4), 551-581.

Smith, S.P., & Duke, D.J. (2000). Binding virtual environments to toolkit capabilities. In M. Gross & F.R.A. Hopgood (Eds.), *Computer Graphics Forum, 19*(3), C81-C89. Blackwell Publishers.

Stanney, K.M., Mollaghasemi, M., Reeves, L., Breaux, R., & Graeber, D.A. (2003). Usability engineering of virtual environments (VEs): Identifying multiple criteria that drive effective VE system design. *International Journal of Human-Computer Studies, 58*(4), 447-481.

Tanriverdi, V., & Jacob, R.J.K. (2001). VRID: A design model and methodology for developing virtual reality interfaces. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 175-182).

Usaka, T., Yura, S., Fujimori, K., Mori, H., & Sakamuram, K. (1998). A multimedia MUD system for the digital museum. *Proceedings of the 3rd Asia Pacific Computer Human Interaction* (pp. 32-37). IEEE CS Press.

Vinson, N.G. (1999). Design guidelines for landmarks to support navigation in virtual environments. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 278-285).

Volbracht, S., & Domik, G. (2000). A model for developing effective navigation techniques in virtual 3D environments. *Guiding Users Through Interactive Experiences: Usability Centred Design and Evaluation of Virtual 3D Environments,* Workshop in Paderborn, Germany.

Wand, Y., Storey, V.C, & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems, 24*(4), 494-528.

Welty, C., & Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering, 39,* 51-74.

Xiaoguang, Z., Dongmu, W., & Bingrong, H. (1999). An object-oriented data framework for virtual environments with hierarchical modeling. *ACM SIGSOFT Software Engineering Notes, 24*(1), 65-68.

Chapter VII

Design of Believable Intelligent Virtual Agents

Pilar Herrero
Universidad Politécnica de Madrid, Spain

Ricardo Imbert
Universidad Politécnica de Madrid, Spain

Abstract

Virtual environments (VEs) have a set of characteristics that make them hard to be designed and implemented: distributed nature, high-level graphical design, technology novelty, and so forth. Because of the criticism or the repetitiveness of some roles played in them, some of the characters of the VEs usually must be automated. The risk is to pay a too high price, losing attractiveness, usability, or believability. The solution proposed in this chapter is to control the automated avatars by associating them with software agents, becoming intelligent virtual agents (IVAs). With this aim, an architecture to manage the perception and cognition of the agent is described. On one hand, the perceptual module of this architecture consists of a human-like model, based on one of the most successful awareness models in computer-supported cooperative work (CSCW), called

the Spatial Model of Interaction (SMI). On the other hand, the cognitive module proposes an easy-to-configure structure, providing it with the precise mechanisms to exhibit reactive, deliberative or, even, more sophisticated social behaviors, incrementing the believability of the IVA in the VE.

Introduction: The Problem of Usability with Immersive Systems

To face up properly to the development of *virtual environments* (VEs) from a designer point of view, it should be first perceived from the perspective of their users' expectations. This principle, which in fact could be extended to the design of any other kind of system, gathers special importance to VEs, as far as their users are actual participants in the system, beyond their classical external role.

On their interaction with VEs, users test the sensation not only by *actively perceiving* the environment, but also by *being perceived* by other peers. This kind of interaction could be considered, therefore, closer to human experience than the one of traditional software. Obviously, this human experience is not simply restricted to pure physical and realistic interactions, but rather to believable expectations of behavior—although that behavior is not achievable in real life.

Thus, VEs imply a major breakthrough in the realm that has been called *presence* of the user in the virtuality. Presence is the feeling of being *inside* and *a part of* the system, even identifying himself/herself with the virtual character that represents him/her in the VE, generically known as *avatar*. That presence, nevertheless, is more related to the interactions among the actors within the VE than to the technology with which it is implemented (Morningstar & Farmer, 1990).

As systems complexity is increased, two new challenges emerge for presence. The first one is related to the appearance in these environments of some roles that should not be performed by a human user, either due to their repetitive and monotonous nature or because of the specific skills they require. This leads to the incorporation of automatic *synthetic characters* able to develop those tasks.

The second one deals with the believability of the user-controlled character behaviors. On one hand, the user must experience the presence in the VE perceiving in other users and, even more, in his/her own avatar, the behaviors that he/she expects to be appropriate in that situation. That implies the management of all the details of every one of those behaviors, with the risk of overloading the user interface.

On the other hand, when the user has decided to use the system, he/she is looking for some goal or functionality, and that aim must not be put down by the management of a complex user interface. Again, some kind of personalized automation must be provided to release the user of everything he/she is not willing to manage.

Paradoxically, to date, most of the proposals and methodologies for the designing of VEs are mainly focused on issues related to the structure, functionalities, and appearance of the virtual world. Even more, if the designing of the virtual inhabitants is considered, it is made from an external and aesthetical point of view, being unaware of their potential complexity.

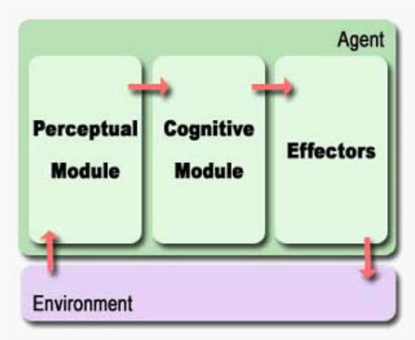
The Agency Meets the Virtual Environments

The problem stated suggests the joining of autonomous or semi-autonomous characters to the VEs. Those individuals should be provided with some kind of perceptual mechanism to acquire all the necessary information to manage their behavior; they should be able to be personalized according to the requirements of their role in the VE or the willingness of their user; they should be able to react to changes or events in their surroundings, as well as acting autonomously every time they consider it appropriate; autonomous characters should be able to interact with other characters, synthetic or user directed.

Those requirements are pretty close to the essential characteristics that some of the most commonly accepted definitions of *agent* attribute to them: **autonomous acting** to reach some set of goals or to accomplish some kind of tasks (Maes, 1995; Wooldridge & Jennings, 1994); **temporal continuity**, throughout a cycle in which the agent perceives, fulfills some kind of cognitive procedure (reactive or deliberative), and determines the actions to be carried out (Bratman, Israel, & Pollack, 1988; Hayes-Roth, 1995; Wooldridge, 2000); sometimes the **specialization** of the agent is highlighted to solve a problem or an specific aspect of a problem (Sycara, 1998); also sometimes stressed as a unquestionable feature of agents is their ability to **interact** with other peers, systems, or human users (Genesereth & Ketchpel, 1994); and always uses to be **situated** in a dynamic, complex environment, over which it acts (Jennings, Sycara, & Wooldridge, 1998; Maes, 1994).

Therefore, the agency seems to be a promising technology to hide from the user the complexity of the interaction procedures without reducing their believability. Even more, the classical internal structure of agents decomposing their operation

Figure 1. Internal structure of an agent



in three consecutive processes—perceptual, cognitive, and effective—fits quite well into the kind of procedures required above.

Thus, thinking in an absolutely automated agent, it would consist of a *perceptual module* able to perceive the state of the environment (Wooldridge, 2002), and restricted by the specific perceptual limitations assigned to that agent in that VE; a *cognitive module,* which deals with the selection of the appropriate actions to be executed depending on the environment perceived, the goals of the agent, and the nature of the agent itself; and finally, a set of *effectors* that execute the selected actions exhibiting the behaviors that could make the agent *believable* (see Figure 1).

For a semi-autonomous agent, this structure is very convenient, given that the automation could be identified in a subset of the three modules, leaving to the user management the rest of them.

Chapter Structure

Obviously, every one of the three previous main modules plays an important role in the consecution of an intelligent believable behavior. However, while the importance of the effectors focuses on the accuracy of the output—in a VE mostly work of the graphical designers—the other two modules must have in mind a wide variety of different concepts and considerations, which makes them more attractive and intricate from the design point of view. This is the reason why the rest of the chapter concentrates on these two modules.

Since most of the agent architectures developed to date do not cover both modules in depth, the following sections will describe a generic architecture for designing intelligent virtual agents (IVAs) with such characteristics. Although it is out of this chapter scope, some practical results of the application of the proposed model can be found in Herrero (2003) and Imbert, de Antonio, and Segovia (2001).

In the following section, the *perceptual module* design is detailed. The reader is first provided with an overview of the studies related to visual perception in IVAs. Subsequently, we describe how the model of perception has been designed: analyzing the factors that make the perceptual model more human-like; re-defining and reinterpreting the set of key concepts introduced by the Spatial Model of Interaction (SMI); and introducing a set of mathematical functions to describe the agent's clarity of perception.

The next section deals with the design of a *cognitive module* for achieving believable intelligent behaviors. That comprises tasks from pure reflex responses to events, to deliberation processes to elaborate, more long-term plans, always having in mind the social nature of agents, which allow them to look for alternatives of solution throughout cooperation or negotiation.

The final section of the chapter summarizes the main conclusions highlighted and states some of the main lines of ongoing work.

Perceptual Module

While some years ago the aim of an agent's perception was just seeking information from the environment, requirements have changed and, currently, a wide range of applications require a relatively high-fidelity model of perception.

This trustworthiness is especially important in order to simulate realistic situations such as military training, where soldiers must be trained for living and surviving risky situations. A useful training would involve endowing soldier agents with a human-like perceptual model, so that they would react to the same stimuli as a human soldier. Agents lacking this perceptual model could react in a non-realistic way, hearing or seeing things that are too far away or hidden behind an object. A different situation could happen, for example, in a museum where, if an agent is too close to a painting, it cannot get a clear impression of the image that is on the painting. The perceptual model we propose in this dissertation introduces these limitations inside the agent's perceptual model with the aim of reflecting a human-like perception.

In this chapter we will concentrate on visual perception, but we have also designed, developed, and implemented a similar model for simulating human-like hearing perception in IVAs (Herrero, 2003; Herrero & de Antonio, 2003a).

Visual Perception in IVAs

Many approaches have been employed to implement the visual process of perception in IVAs, oriented to different kind of applications, such as *artificial creatures* (Blumberg, 1997; Terzopoulos & Rabie, 1995) or *virtual humans* (Chopra-Khullar & Badler, 2001; Hill, Han, & van Lent, 2002; Noser, 1997; Thalmann, 2001).

Perception in those agents has been modeled in diverse ways, depending on what they were designed for. Most of the perceptual models to date have focused on providing methods and techniques for modeling the cognitive process of perception, such as the Cognitive Vision Systems (CogVis).

The CogVis is a project sponsored by the European Union IST-2000-29375. The objective of this project is to provide the methods and techniques that enable construction of vision systems that can perform task-oriented categorization, and recognition of objects and events in the context of an embodied agent. Cognitive vision systems include facilities for *understanding*, *knowing*, and *learning*. *Understanding* implies an ability to generate an explicit description of the perceived world in terms of objects, structures, events, their relations and dynamics that can be used for action generation or communication. *Knowing* implicitly specifies a need to consider memory as a common basis for representation and maintenance of information, including methods for associate access. *Learning* implies an ability to generate open-ended models and representations of the world. That is, the model of the system and its use cannot be based on a closed-world assumption, but rather on a model that allows automatic generation of new representations and models. Cognitive vision only makes sense in the context of a system where there is a user that provides task information and which uses the information generated by the system. In addition a fundamental assumption is that such systems are embodied so that they interact with the world and have the potential for interaction with the world using active vision, manipulation, or similar facilities.

As humans "see" the places they visit with some precision, the cognitive mapping techniques have focused on providing us with a description of each local space visited. While cognitive mapping has been examined in the context of mobile robotics, very little work has been done to enable virtual humans to build and use cognitive maps. An example of this work is the cognitive mapping technique implemented by Hill et al. (2002). Their implementation is based on a computational framework that represents a local environment as a structure called an Absolute Space Representation (ASR). Building an ASR involves perceiving the environment, which is the area immediately surrounding the viewer, building up a mental model of the space, and computing the *boundaries*—which prohibit movement through the space—and *exits*—which are gaps in the boundaries that permit the agents to leave this space. The cognitive mapping algorithms used are an extension of those presented by Yeap and Jefferies (1999). Hill has applied this theoretical computational framework of cognitive mapping to a training application that includes virtual humans in a virtual environment.

Cognitive mapping is not limited to places that have been physically explored. Virtual humans map the environment by continuously perceiving a scene, constructing a sketch of the surfaces, and building a local map. This local map is connected with other local maps that have been constructed while exploring the virtual town. Instead of focusing on only the immediate surroundings, virtual humans gather information about other regions perceived through the exits in the local environment and, therefore, virtual humans build cognitive maps in anticipation of the next space they will enter. To do this, agents perceive through the exits in the local environment and construct the new ASRs before the areas are visited.

Several models of visual attention for virtual humans have been proposed by Chopra-Khullar and Badler (2001) and Hill et al. (2002). Chopra's work, based on human psychological research, specifies the types of visual attention required for a variety of basic tasks (e.g., locomotion, object manipulation, and visual search), as well as the mechanisms for dividing attention among multiple such tasks. In the Soar Virtual Pilot, Hill also focuses on providing a model of perceptual attention for virtual humans in a synthetic battlefield.

Even though cognitive perception plays a very important role in each and every perceptual model, the agent's perception would not be complete without taking the sensorial part of the process of perception.

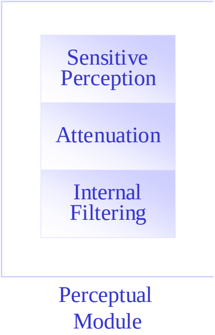
Most of the research carried out on modeling sensory inputs has been focused on using cameras, sensors, and so forth, and different techniques of computational vision, gathering information about the dynamic environment without taking into account *human factors* (Terzopoulos & Rabie, 1995; Thalmann, 2001). A classification of current approaches can be found in Herrero (2003).

Since the current studies on the agent's perception do not consider the most useful and representative human perceptual factors such as sensorial acuity, the following sections will describe a generic architecture for designing a perceptual model for IVAs with these perceptual factors. Some practical results of the application of the proposed model can be found in Herrero and de Antonio (2002, 2003b) and Herrero (2003).

An Architecture for the Agent's Perception

As previously mentioned, our architecture has three main blocks (Figure 1), representing the agent's perceptual module, the agent's cognitive module, and

Figure 2. Agent's perceptual module



the agent's effectors. The perceptual module, or perceptual engine, operates concurrently with the cognitive module, and some of the interpretations of the perceived data or some of the parameters of the agent's internal model can in turn modify the perceptual process. The perceptual engine will manage the interaction with the environment, and it will be composed of the following three sub-modules (Figure 2): *sensitive perception*, *attenuation*, and *internal filtering* (Herrero, 2003).

Although in the following sections we concentrate on describing the sensitive perception module of this perceptual engine, we also have to bear in mind that perceptual sensations are subjectively attenuated with time. The attenuation module will introduce a reduction experienced by the signal coming from the sensitive perception. On the other hand, the internal filtering module will make the selection of the most relevant objects within the focus of perception.

Designing the Visual Model of Perception

One of the most important characteristics of IVAs is the ability to be aware of current situations in the environment where they reside and operate. Following Endsely studies on "situational awareness," physical perception can be understood as the first level of an awareness model (Endsley, 1998, 1993; Shively, Brickner, & Silbiger, 1997).

Awareness is very broad concept with different meanings in different areas of application. In this way, the Spatial Model of Interaction (Benford & Fahlen, 1993) is an awareness model designed for CSCW (computer-supported collaborative work) applications which uses the properties of the space to get knowledge of the environment.

The SMI was based on a set of key awareness concepts, which could be extended to introduce some human-like factors, and it had been tested with successful results in CSCW multi-user environments; this model has been selected because it has the essential qualifications for our purposes.

The aim of our research is not just to extend the SMI to mIVAs, but also to make it more realistic, introducing some concepts typical in human-like perception. In order to get that, a reinterpretation of the meaning of "awareness" has to be made—quite different from the definition used in CSCW literature (Dourish & Bellotti, 1992)—as well as a reinterpretation of the key concepts of the SMI to introduce them as key concepts of a perceptual model, applying this model to IVAs.

The proposed perceptual model seeks to introduce more coherence between IVAs' perception and human being perception. In this way, the psychological "coherence" between the real life and the virtual environment experience will be incremented. Although this section describes a visual model of perception, an auditory model of perception has also been developed following the same structure (Herrero & de Antonio, 2003a).

Key Concepts in the SMI

As mentioned in previous sections, the key concepts in the visual model of perception are based on the main concepts of SMI. The spatial model, as its name suggests, uses the properties of space as the basis for mediating interaction. It was proposed as a way to control the flow of information of the environment in CVEs (collaborative virtual environments). It allows objects in a virtual world to govern their interaction through some key concepts: medium, aura, awareness, focus, nimbus, adapters, and boundaries.

Aura is the sub-space that effectively bounds the presence of an object within a given medium and acts as an enabler of potential interaction. In each particular medium, it is possible to delimit the observing object's interest. This area is called *focus:* "the more an object is within your focus, the more aware you are of it." The focus concept has been implemented in the SMI as an "ideal" triangle limited by the object's aura.

In the same way, it is possible to represent the observed object's projection in a particular medium. This area is called *nimbus*: "the more an object is within your nimbus, the more aware it is of you." The nimbus concept, as defined in the SMI, has always been implemented as a circumference in a visual medium. The radio of this circumference has an "ideal" infinite value, although in practice it is limited by the object's aura.

The implementations of these concepts—focus and nimbus—in the SMI did not have in mind human aspects, thus reducing the level of coherence between the real and the virtual agent behavior.

The main concept involved in controlling interaction between objects is *awareness*. One object's awareness of another object quantifies the subjective importance or relevance of that object. The awareness relationship between every pair of objects is achieved on the basis of quantifiable *levels* of awareness between them, and is unidirectional and specific to each medium. Awareness between objects in a given medium is manipulated via *focus* and *nimbus*. Moreover, an object's aura, focus, nimbus, and hence, awareness can be modified through *boundaries* and some artifacts called *adapters*.

Making the Visual Perceptual Model More Human-Like

There are many factors that contribute to our ability as humans to perceive an object, some of which are directly working on the mental processes, being not easily modeled or reproduced in a virtual world. Some key concepts of human perception have been analyzed before determining which one of them could be introduced in our visual agent's perceptual model. These concepts, selected for being the more representative of human visual perception, are (Herrero & de Antonio, 2002, 2003b; Herrero, de Antonio, Benford, & Greenhalgh, 2002; Herrero, 2003):

- *Visual Acuity*: Representing the general "sense acuity" in a visual medium, the visual acuity is a measure of the eye's ability to resolve fine detail and is dependent upon the person itself, the accommodative state of the eye, the illumination level, and the contrast between target and background (Howarth & Costello, 1997). Virtual agents that exhibit this property would be able, for instance, to perceive a message written on a notice board, only if the distance from the agent to that notice board is within the visual range of perception.
- *Lateral Vision*: Representing the general "sense transition region" (STR), the lateral vision corresponds to the visual perception towards the extremes of the visual focus. Virtual agents should exhibit this characteristic to avoid anomalous behaviors, for example, those that will happen if an agent is not aware of and cannot interact with another agent who is inside the lateral vision area, but out of the visual focus.
- *Visual Filters*: These allow the selection from all the objects in an extensive focus of only those that the agent is especially interested in.

From these three factors, this section concentrates on the two first concepts (visual acuity and lateral vision) without eluding the third one (visual filters) when the perceptual model has been developed and implemented for IVAs.

Reinterpreting the SMI's Key Concepts

Neither the SMI nor its implementations are considered aspects of human perception. Thus, if the SMI were applied just as it was defined by Benford, the level of coherence between real and virtual agent behavior would be minimum. Some factors concerning human-like perception have been identified, providing a more believable perception.

In this section we describe the human factors considered and how the key concepts defining the SMI have been modified to introduce these factors.

Visual Focus—Benford introduced the focus concept in 1993 as: *"The more an object is within your focus, the more aware you are of it"* (Benford & Fahlen, 1993). This concept meant that the observing object's interest for each particular medium could be delimited. According to this definition, the focus notion is the area within which the agent perceives the environment. In previous sections, the work of sensitive perception in human beings has been analyzed, and from this analysis, some physical factors—which should have an effect on the physical area delimiting the observing object's interest—have been selected. These factors are *sense acuity* and the *sense transition region*.

Starting from the focus concept in the spatial model, and bearing in mind previous implementations, for example by Greenhalgh (1997), where focus was implemented as a cone, sense acuity and sense transition regions have been introduced. We will define a new mathematical function to represent the human-like focus concept. This mathematical function (Equation 1) will be described by the following set of variables and parameters, and is represented in Figure 3:

- (μ_x, μ_y, μ_z) Represents the agent's eye position in a 3D system of reference.
- Dm Represents the agent's visual resolution acuity distance.
- (x,y,z) Represents any point inside the focus.
- θ' Represents the angle delimiting human foveal vision.
- θ Represents the angle delimiting human vision: foveal and peripheral vision.
- s Represents the object's size.

In the implementation of the model, we have separated global focus, which has infinite length, from specific focus, associated with each agent. The length of

Equation 1.

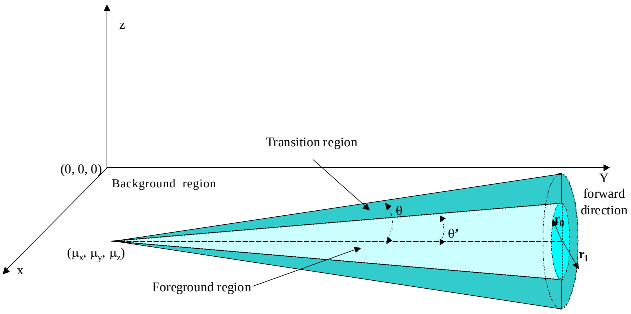
$$\mu_y \le y \le \mu_y + D_m$$

$$(x - \mu_x)^2 + (z - \mu_Z)^2 \le (\mathrm{tang}\,(\theta))^2 * (y - \mu_y)^2$$

$$(x - \mu_x)^2 + (z - \mu_Z)^2 \le (\mathrm{tang}\,(\theta'))^2 * (y - \mu_y)^2$$

$$D_m \le \frac{s}{\mathrm{tang}\,(MAR)}$$

Figure 3. Physical focus with lateral vision



global focus is limited by the aura, while the length of specific focus is limited by each agent’s physical factors.

When the agent perceives an object in the environment, perception will be different depending on the area in which the object is located. Both the object’s orientation (Figure 4) and the area in which it is located (Figure 5) play an important role in determining the perception of the object.

As Figure 3 shows, two different cones can be distinguished, the internal cone (with angle θ’) represents the agent’s field of vision without lateral vision, and the external cone (with angle θ) represents the agent’s field of vision with lateral vision (STR). Both cones have been implemented as functions delimiting the agent’s visual perception area. Starting from some experiments run for “The Old Man” (Herrero, 2003), the origin of the cones will be placed at an eighth part of the object’s height (above the nose and in between the agent’s two eyes).

In Figure 5, the Area of Perception (AP) indicates whether an object is within the focus, and, in this case, within which area it is located. For our purposes, we have implemented a function that checks whether an object is inside the agent’s

Figure 4. Agent's eye orientation and object's position

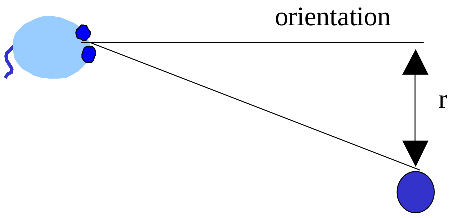
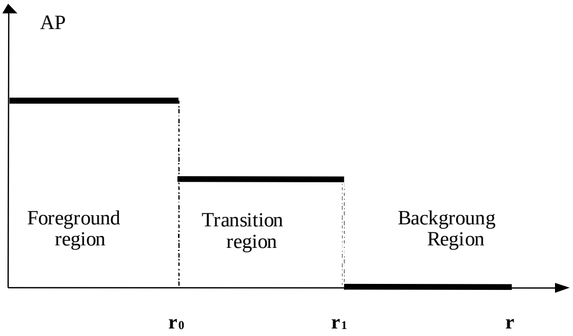


Figure 5. Physical focus with lateral vision (distance component)



focus, and if it is, then this function will indicate the area within which the object is located (foreground or transition region). This function will allow us to determine whether the agent can detect an object because it is inside its field of vision. If the agent's objective is not just to detect the object, but also to perceive some details of the object, we will also be interested in the clarity of the perception that the agent has of the object.

Considering medium homogeneity, it has been found that, while in a homogeneous medium, the focus shape is uniform and corresponds to a cone, whereas in a heterogeneous medium, it could have discontinuous transitions between regions with different densities. We are not going to deal with heterogeneous media in our model.

Our initial equation considers that the cone orientation is parallel to the y-axis. Otherwise, this approach will be a valid subject to the previous rotation of the axes according to *Euler's Rotation Theorem*.

- **Visual Nimbus:** Benford introduced the nimbus concept in 1993 as: *"The more an object is within your nimbus, the more aware it is of you"* (Benford & Fahlen, 1993). This concept meant that the observed object's projection for each particular medium could be delimited.

The nimbus concept, as defined in the Spatial Model of Interaction, has always been implemented as a circumference in both visual and hearing media. The radius of this circumference has an "ideal" infinite value, although in practice it is limited by the object's aura. Just as with the above-mentioned focus concept, the nimbus concept in the SMI does not consider any human factors, thus hypothetically reducing the level of coherence between real and virtual agent behavior. We are going to represent the nimbus of an object as an ellipsoid (Equation 2, Figure 7) or a sphere (Equation 3), depending on the conic by which it is circumscribed (Figure 6), centered on the object's geometrical center. The way of determining which conic has to be associated with each object in the environment is to look for the bounding box that has been associated to this object in the environment. If the bounding box is a rectangle, the nimbus has been approximated as an ellipsoid; if the bounding box is a circle, then the nimbus will be approximated as a sphere.

The nimbus radius, or its eccentricity if it is an ellipsoid, will depend on two factors: the object's shape and the furthest distance at which a human being would be able to distinguish the object. This distance is determined by visual acuity, which depends on the object's size; thus, indirectly, the nimbus conic will depend on the object's size as well.

Where (m_x, m_y, m_z) represents the object's geometrical center, (a,b,c) represents the ellipsoid parameters and R represents the sphere radius (when $a=b=c=R$).

Figure 6. Nimbus representations for geometric objects

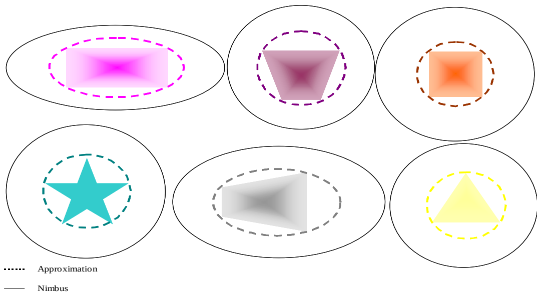
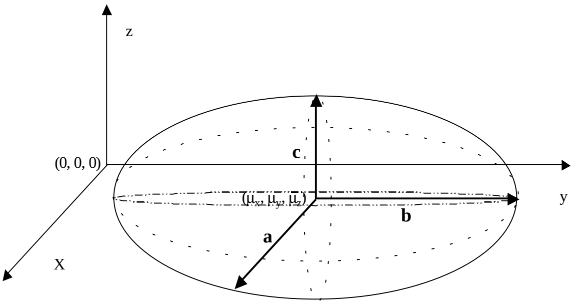


Figure 7. Physical nimbus representation



Equation 2.

$$\left(\left(\frac{x-\mu_x}{a}\right)^2+\left(\frac{y-\mu_y}{b}\right)^2+\left(\frac{z-\mu_z}{c}\right)^2\right)\leq 1$$

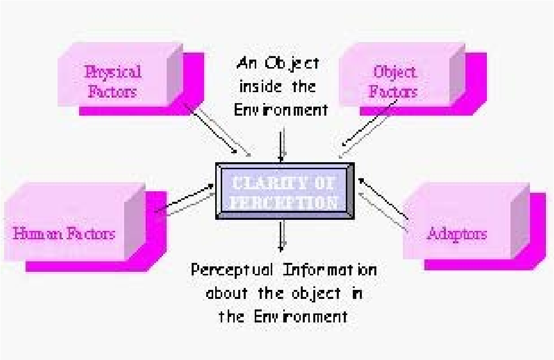
Equation 3.

$$(x-\mu_x)^2+(y-\mu_y)^2+(z-\mu_z)^2\leq R^2$$

Visual Clarity of Perception

This section concentrates on the sensitive perception block introduced previously. The sensitive perception module simulates the typical process by which organisms receive sensations from the environment. Sensation usually refers to the immediate, relatively unprocessed result of stimulation of sensory receptors in the eyes, ears, nose, tongue, or skin. Sensitive perception depends on some relevant sensorial concepts (Figure 8): *human factors* such as visual acuity, lateral vision, and visual filters; *physical factors* such as the distance between the object and the position of the agent's eye (*deye-object*); *object's factors*; and *adaptors* (Herrero & de Antonio, 2002, 2003b). The *deye-object* distance and clarity of perception, in general, should be considered key concepts in an agent's perception because it introduces more realism, believability, and efficiency. For example, it will be necessary to check its value to know if an agent can read a notice board at a fixed distance. Moreover, making awareness dependent on this factor is totally new; no other model had it in mind before.

Figure 8. Sensitive perception

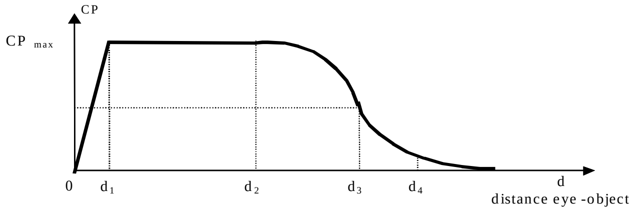

Physical Factors
An Object inside the Environment
Object Factors
CLARITY OF PERCEPTION
Human Factors
Adaptors
Perceptual Information about the object in the Environment

Clarity of perception is a measurement of the ability to perceive an object inside the agent's visual focus, as well as the clearness of this perception. Once an object's nimbus intersects with the agent's focus, the sensitive perception module will calculate the *clarity of perception* for this object.

The process of human visual perception is continuous, and the size of the image on the retina will continuously depend on the distance between the eye and the object to be perceived. Therefore, from the sensorial point of view, if clarity of perception is the ability to distinguish what kind of object is being perceived, then it should depend on the object image that human beings have on the retina. Moreover, as the retinal image decreases continuously with the eye-object distance, then the clarity of perception will decrease continuously with this distance as well. But the size constancy phenomenon has also been taking into account, by means of which the object's size tends to appear constant in spite of it changing with distance. This factor will imply that the clarity of perception will fall still more smoothly. Following the research conducted by Levi, Klein, and Hariharan (2001a), a Gaussian has been proposed as the function to describe the variation in the clarity of perception with the eye-object distance (Figure 9, Equation 4) for a fixed object's size in the foreground region, where $d1$ represents the minimum distance necessary to have a clear perception of an object and $d2$ represents the maximum distance necessary to have a clear perception of an object. In Figure 9 it is possible to appreciate that the level of detail starts decreasing (between $d2$ and d_3), and starting from $d4$ the eye cannot perceive almost any detail from any object. More details are given in Herrero and de Antonio (2002, 1003b) and Herrero (2003).

The clarity of perception function in the transition region has to take into account the presence of peripheral vision. Peripheral vision, as mentioned above, is paying attention to what is happening at the periphery of your field of vision. In

Figure 9. Clarity of perception relative to distance inside the focus foreground region

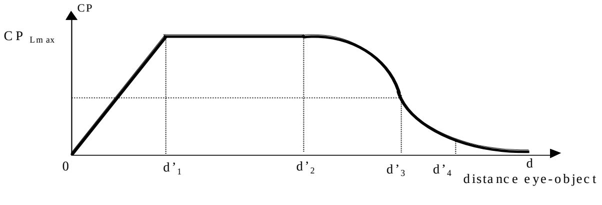


Equation 4.

$$
\begin{aligned}
&0.0 \le d \le d_1 \quad CP(d) = \lambda d \\
&d_1 \le d \le d_2 \quad CP(d) = CP_{\max} \\
&d \ge d_2 \quad CP(d) = \frac{1}{\sigma * \sqrt{2 * \pi}} * \exp\left\{ - \frac{(d - d_2)^2}{2 * \sigma^2} \right\}
\end{aligned}
$$

this area you may become aware of movement, but you are less aware of color and contrast distinctions. Following the research by Levi et al. (2002b), another Gaussian has been proposed as a function to describe the variation that the clarity of perception has with the distance eye-object (Figure 10, Equation 5) in the lateral region. More details are given in Herrero and de Antonio (2002, 2003b) and Herrero (2003).

Figure 10. Clarity of perception relative to distance inside the focus transition region



Equation 5.

$$
\begin{aligned}
&0.0 < \mathrm{CP}_{\mathrm{Lmax}} < \mathrm{CP}_{\max} \\
&\mathrm{d'}_1 > d_1 \quad \mathrm{d'}_2 < d_2 \quad \mathrm{d'}_3 > d_3 \quad \mathrm{d'}_4 \approx d_4
\end{aligned}
$$

Cognitive Module

Gaining Believability Through Cognition

Perhaps, the **OZ project** (Bates, Loyall, & Bates, 1994) may have been the first real project on believable agents in interactive environments. Bates introduces the concept of *believable agents*, meaning that a viewer or a user can suspend his or her disbelief (Loyall & Bates, 1997).

The Oz project simulates a small world whose inhabitants, spherical avatars named *Woggles*, are built through a goal-directed, behavior-based architecture for action. This architecture is coupled to a distinct component for generating, representing, and expressing emotion, based on *Ortony*'s *Cognitive Theory of Emotions* (Ortony, Clore, & Collins, 1988). By and large, they claim that personality and emotions are the most important aspects of believability to add to social behaviors (Reilly & Bates, 1995; Reilly, 1997). This project intends to make it easy to build characters with specific personalities, using the minimal representation for them.

In this framework, individuals are assumed to behave differently in the same emotional state or the same interpersonal relationship, or may feel different emotions in similar situations because of current behavioral features.

One of the issues that they have faced is that many emotions of various intensities often exist simultaneously (Bates, 1994), and they have had to find ways to combine these to get one or two primary overall emotions of adequate clarity to express a coherent internal state.

On the other hand, **The CyberCafe** (Rousseau & Hayes-Roth, 1997) introduces the concept of *synthetic actors*. A synthetic actor may be autonomous or a user's avatar. An autonomous actor receives directions from the scenario and other actors, and decides on its own behavior on the virtual stage with respect to those directions (Hayes-Roth, Brownston, & Sincoff, 1995). An avatar is largely directed by a user who selects actions to perform, although it also receives directions from the scenario and from the other actors. In fact, the user chooses the actions to be developed by the avatar, but the way to be carried out is chosen by the avatar, depending on the character personality traits. These actors are able to improvise their behavior in an interactive environment, and they own a repertoire of actions that are automatically planned to achieve each activity. They even reflect aspects of personality traits and mood.

The **ALIVE project** (Maes, 1995) points out that how fancy graphics are may be less important than how meaningful the interactions in which the user engages can be. However, although one of the aims of ALIVE project is to visualize the motivational and emotional state of an agent in the external features of the agent, no serious personality model has been developed (it uses ethological mechanisms to maximize the actor's ability to reorganize its own personality, based on its own perception and accumulated experience). In fact, personality is not the main interest of this project.

Another interesting system, **Bodychat** (Vilhálmsson, 1997), also tries to automate the communicative behavior in avatars. Here, the concept of *intention* for this context is introduced. Intentions are described as a *"set of control parameters that are sent from the user's Client to all Clients, where they are used to produce the appropriate behaviour in the user's Shadow avatars."*

Bodychat proposes an avatar as a partially autonomous entity, providing an automated facial expression and gaze that depends on the user's current intentions, the current state and location of other avatars, its own previous state, and some random tuning to create diversity.

Improv (Perlin & Goldberg, 1996) offers an environment where an avatar can generate motions in real time; however, the conversation between avatars is not addressed. Improv provides tools to create actors that respond to users and to each other in real time, with personalities and moods consistent with the author's goals and intentions.

Improv is more than a simple tool for designing virtual actors. It allows actors to have certain information about it and his relationship to his environment stored in an actor's properties. With these properties, one may describe an actor's personality, current mood, attitudes, and his relationship to other actors or objects. The system uses decision rules to generate weighted decisions.

The **Cognition and Affect Project** proposes a much elaborated model for describing human emotionality. Sloman and Logan (1998) conjecture that human mental concepts (e.g., belief, desire, intention, experience, mood, emotion, etc.) are grounded in implicit assumptions about an underlying information processing architecture. They claim that the normal adult human architecture involves three main layers, each supporting different sorts of *mental concepts*: the first layer is also the oldest in evolutionary terms, and is entirely reactive; the second layer is deliberative; the third—perhaps the more original and distinctive feature of this architecture—is a reflective layer.

The **Byrne** system is another interesting related work (Binsted, 1999). It presents an animated talking head for generating appropriate affective speech and facial expressions, while retransmitting soccer matches, based on the character's personality, emotional state, and the state of the play.

The system does not make a great effort to make the emotion component of our system cognitively plausible. The goal of *Byrne* is not to develop a psychologically realistic personality, but to generate a consistent character. Therefore, Byrne is more folk psychology than modern cognitive science.

Inside the character architecture, an *emotion generation* module can be found, containing rules that generate simple emotional structures. These structures consist of a type (a mood), an intensity (scored from 1 to 10), a target (only when the emotion is directed at some person or object), a cause (what caused the emotion in the virtual world), and a decay function (sharing Elliott's opinion that moods naturally return to default values over time (Elliott, 1997)).

Still, the authors consider moods as independent concepts, and face the issue of having simultaneously contradictory, inconsistent values for opposite concepts (e.g., *high* simultaneous values for both *happiness* and *sadness*).

Finally, the way of generating an output emotion is choosing that mood with the highest intensity, in order to eliminate the possible inconsistencies. The problem lies in the discarding of the rest of the not-so-high value moods. To stand up to this situation, the authors propose to mix consistent emotional expressions, but that is not always possible.

In summary, most of the above-mentioned models achieve a high degree of expressiveness and believability at the expense of the user: when he/she has few parameters to manage, his/her avatar loses in both expressiveness and believability; when he/she has an awful lot of them, it becomes impossible to properly control all of them, even having a powerful model.

An Archetypal Architecture for the Cognitive Module

In order to identify an architecture for this module, the first issue to be taken into account is the kind of behavior control desired for the whole agent. Initial trends, mainly influenced by systems such as Newell and Simon's (1963) GPS, gave rise to **deliberative architectures**. These architectures are based on the following premise: "a system able to manipulate a symbolic representation of its environment, describing the goals and means to satisfy them, could be able to exhibit an intelligent behaviour."

This kind of "reasoning" yielded excellent results in small simulations or very specific contexts, but showed some scalability limitations in more realistic scenarios. The main reason lies in the necessary assumption of the *calculative rationality property* (Russell, Subramanian, & Parr, 1993). *Grosso modo*, this property states that, from an observation of the environment carried out in a moment *t*, an action decided as a result of a deliberative process would be optimum always, since it was executed in that moment *t*. That is, during the process of deliberation, the environment should not change to maintain the action optimality.

These limitations make this kind of architecture somewhat suitable for time-demanding applications, such as most VEs. For this reason, in the early 1980s arose the first trends questioning the viability of symbolic reasoning-based approaches to obtain “intelligent” behaviors.

Brooks (1991) assesses that problem solving through the use of symbolic abstractions of the universe provides simplistic representations of it, only valid for *toy prototypes* and impossible to scale. Thus—always according to Brooks—representation is a wrong abstraction unit to build the core of intelligent systems. The VE itself would be a better model instead of a representation of it. This led him to the proposal of **reactive architectures**, based on a model of stimulus-response.

However, despite the fact that this approach has provided excellent results in empirical applications, it is not free of important disadvantages, which make it not appropriate for a number of contexts (Jennings, Sycara, & Wooldridge, 1998).

Viewing the lack of both alternatives, at the beginning of the 1990s appeared the trend of using *hybrid architectures*, the most common alternative hitherto in VEs. This kind of approach combines reactive skills—so to provide fast critic responses—and deliberative processes—to elaborate complex, less time-demanding plans.

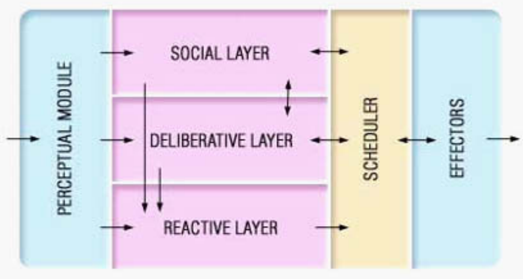
Hybrid architectures commonly present an internal layered structure. The number and design of these layers is very much dependent on the context of the system, but it is usual to find three layered architectures, generally horizontally arranged, with a distribution similar to the one proposed in the following (also see Figure 11):

- **Reactive Layer**, which provides with dynamic responses according to the changes perceived in the VE.
- **Deliberative Layer**, to analyze the current situation, taking into account the agent goals and interests, and its personal skills to structure plan-shaped solutions.
- **Social Layer**, to cope with the current situation, taking into account the potential interactions with other agents in the system, to structure also plan-shaped solutions.

According to how one accesses the input information and to the output generation, the layer distribution of the proposed architecture could be classified as *horizontal*, given that all of them are able to simultaneously perceive the input data from the perceptual model and, also, all of them produce concurrently their action proposals towards the effectors.

As a matter of fact, that action proposals communication is not directly performed to the effectors, as shown in the Figure 11. From our point of view, it is more convenient to carry it out through an intermediate element, noted as *scheduler*, shared by the three mentioned layers, and whose responsibility is to

Figure 11. General architecture of the agent, centered in the cognitive module



sort and sequence the actions towards the effectors. Maybe this scheduler could have been part of the effectors, but the dimension of the decisions to be made in it makes it more convenient to include it inside the cognitive module.

Regarding the interaction among every architecture block, *grosso modo*—detailed later—we could identify the following basic communication flows:

- The three layers—reactive, deliberative, and social—access the following from the perceptual module notifications: state changes, events, and information.
- The social and deliberative layers process the input and, according to certain factors analyzed later, produce new goals and propose potential plans to satisfy them, each from their own particular perspective.
- In turn, according to the established goals, indicate to the reactive layer the general guidelines of it behavior to be coherent to those goals.
- This reactive layer, mainly from the perceived input by the perceptual module and from the behavior guidelines received, also can propose to the scheduler the actions considered appropriate.
- The scheduler collects the plans and actions proposed and structures them to avoid conflicts among them, trying to maximize the agent behavior.
- From the *agenda* generated, the scheduler is able to provide the effectors with the selected actions in the convenient order.

One of the main features of the architecture, as will be seen further on, is that it is valid for any kind of IVA, avatar, semi-autonomous and autonomous. In fact, the autonomy of the agent depends on the number of actions it is able perform on its own in the VE, from none—avatar—to all its possible actions—pure autonomous agent. The semi-autonomy is possible since explicit mechanisms of agent behavior characterization are provided, to perform user-delegated functionalities, whereas the user's activity can be processed as a high priority input.

Therefore, the coherence of the agent behaviors will be based on three fundamental pillars: (1) the agent *knowledge/beliefs* maintenance, to support and justify its behavior; (2) the automated update of the values of those information structures throughout the precise *correlations* among them, designed *ad hoc*; and (3) the *management* of the agent needs and beliefs jointly, in order to originate the appropriate actions in every one of the layers.

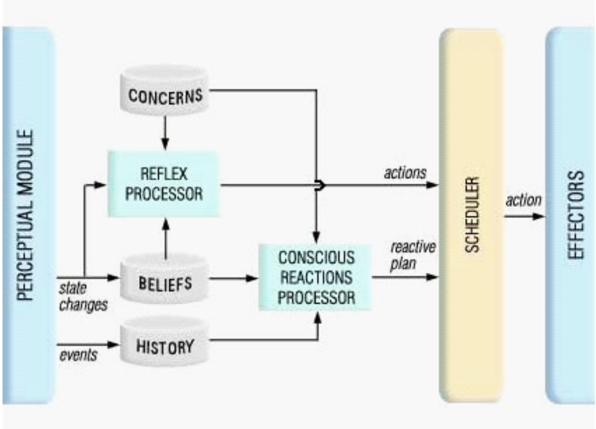
Reactive Layer

The aim of the *reactive layer* is to respond quickly to changes in the state perceived by the agent. These changes are normally produced by internal or external events and/or the acquisition of new information. Its importance arises from the existence of some situations in which an immediate response is required. The time available for such a reaction is so limited that it will be insufficient to run a planning process. In fact, at this level, neither the elaboration of new plans nor the explicit evaluation of alternative behaviors is considered at all.

Inside of the reactive layer, two kind of different reactive processes could be distinguished, depending on the voluntary nature implicit in the responses:

- **Reflex Processing:** Under certain changes in the environment, from the current beliefs and concerns of the agent, this one is able to produce appropriate responses with a very low level of voluntariness. This is the kind of *pre-attention* reactions that Allen (1999) considers just enough for an agent to survive in environments in which these generic solutions often do not fail. However, by definition, the environment in which the agent is situated and acts may not be deterministic, and its knowledge about it may be local (Rao & Georgeff, 1995), and even could be incomplete or wrong. Therefore, the execution of a certain action in a given moment will not necessarily lead to the situation (the environment state) hoped/desired by the agent.
- **Conscious Reactions Processing:** The agent is also able to react against situations not directly triggered by outcomes or arrivals of information, but rather owed to the existent beliefs about the current state (*beliefs*) and about past states (*history*), always according to their specific concerns. Thus, this kind of behavior implies a slightly upper degree of reaction

Figure 12. Agent's reactive layer architecture



consciousness. It could also mean the proposal of one or more actions to be executed, sometimes as preconceived *reactive mini-plans*.

The schema of the specific architecture corresponding to the reactive layer proposed is depicted in Figure 12. Besides the two main processes described before, the rest of its components and interfaces are analyzed below.

Beliefs

Conceptually, these kinds of beliefs are quite similar to those of the BDI (Belief-Desire-Intention) architectures (Georgeff, Pell, Pollack, Tambe, & Wooldridge, 1999). They represent the agent information about the more likely state of the world. They are essential because of the world dynamism and the local perspective of the environment—many events out of the agent perception sphere must also be taken into account and remembered.

It is preferable to deal with *beliefs* instead of managing *knowledge*, because it is assumed that an agent's beliefs could be wrong or incomplete, whereas knowledge should be correct. Besides, agents in the past perceived a limited view of the environment: they are neither prescient nor omniscient.

Taking into account its situation in a VE, the kinds of beliefs that a IVA should manage include not only information—both the essential and the transitory one—about objects, places, individuals, and the current situation of the IVA itself, but also emotional or emphatic information itself and others, such as personality traits, moods, physical states, attitudes, and so forth (Imbert & de Antonio, 2000).

The reactive layer is provided by the perceptual module with pertinent information, mainly *changes of the perceived environment*. These changes are appropriately stored as beliefs, becoming the input source for both reflex and conscious reaction processing. Of course, given the assumption of the incomplete nature of beliefs, some parameters concerning the data confidence degree and temporal validity could be associated with the information saved.

History

History refers to all the knowledge managed by the agent dealing with circumstances that took place in the past. Considering the history of the agent as a directed graph, linear from the present point towards the past, and branched from the present towards the future (Wooldridge, 2002), history is indispensable to maintain coherence in its behaviors with regard to past events. Thus, it is a cornerstone to obtain believability in the agent behavior (incoherence in the agent behaviors during the time is a major cause of unbelievability, quantitatively worse than not having fancy avatars or scenarios).

History is maintained by the events perceived and filtered in the perceptual module, stored in the appropriate format to allow reasoning from them. This information, at this level, is only relevant for the conscious reaction processing, as far as reflex processing involves exclusively immediate, instinctive reactions.

Concerns

Concerns deal with the behavior interests of the agent. Concerns are not goals: a goal is an aim for the agent, something that the agent intends to achieve; concerns, however, describe behavior guidelines. Their essence is similar to the concerns in MINDER1 (Wright, 1997), but here they are considered more process-independent (in fact, the agent could work even if no concern is defined, whereas for Wright the existence of concerns is unavoidable).

Due to their structure, concerns allow the control of the rest of the agent layers over the reactive one. In the reactive layer, concerns are only consulted, never modified. Hence, depending on the active agent concerns, introduced at design time, or by other layers at run time, the reactive layer is able to exhibit believable behaviors according to their deliberative and social strategies.

Thus, for instance, a logical reflex reaction against an arm prick would be to take away the arm just to avoid the pain; however, when the goal is to be vaccinated, the concern of “suffering some pain” may be warranted. In other words, the new goal could raise the associated threshold of pain. These thresholds could be determined by default through the values of the personal characteristics—personality traits—that determine the guidelines of the specific behavior of the agent (stored among the beliefs).

Therefore, at the reactive layer, the concerns will transmit the desires of the agent designer to its behavior and adjust the reactive behaviors—reflexes and conscious—according to the deliberative and social strategies.

Deliberative Layer

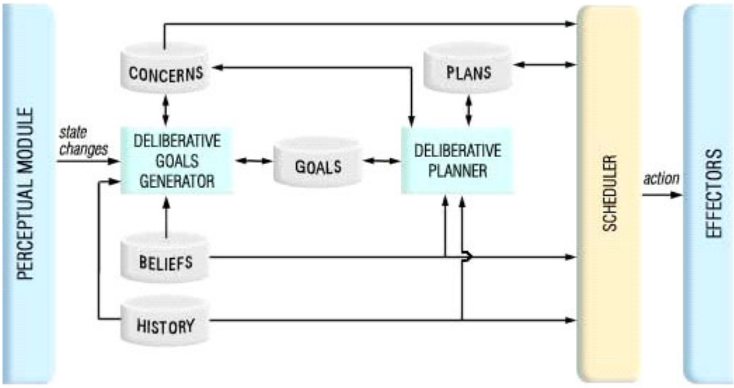
Given that having an agent with only a reactive layer is not always enough to exhibit a believable complex behavior in most situations, it is usual to incorporate a *deliberative layer* to respond to long-term goals and deal with elaborated plans. That means that the deliberative layer works with goals without extreme time limitations—that is, assuming the calculative rationality property.

In the agent deliberative layer, the scope of the planning is to achieve some goals from the personal skills of the agent, that is, without taking into account the capabilities of other avatars to solve its own aims.

Two main processes work sequentially inside this layer, as shown in Figure 13:

- **Deliberative Goals Generator:** This opportunistic process takes into account possible changes in the environment, local beliefs—mainly regarding its personal sphere information—past history, active goals, and current concerns to propose new goals to lead the future agent behavior. In addition, at the same time that new goals are generated, new concerns could be produced or the values of existing concerns could be modified to enable the achievement of the goal. Thus, in the previous example, when

Figure 13. Agent's deliberative layer architecture



the new goal of "being vaccinated" is incorporated into the agent's goals, the threshold of the concern "pain resistance" could be raised a bit. This process is also able to perceive whether an existing goal is still interesting for the agent or not, and to maintain or suppress it according to its considerations.

- **Deliberative Planner:** Once a set of goals is available, the second process inside the deliberative layer begins its operation. The deliberative planner builds a complete *plan* to achieve every goal using as fact base its beliefs, information about past history, and concerns. Whenever new intermediate goals emerge during the planning process and the deliberative planner is unable to build plans to achieve them, this planner will include them in the *goals base* to allow other layers to look for a plan for them. This process also supervises the set of available goals to eliminate non-executed plans from the *plans base* whenever their original corresponding goals have been removed from the *goals base*.

This deliberative layer, like the reactive one, manages the information contained in the beliefs, history, and concerns. Regarding the concerns, this layer is, unlike the previous one, able to add and modify them. But also two new structures appear at this level.

Goals

Goals represent a desired final state of the agent, that is, the motivation of the agent. Obviously, goals may be added during the execution of the agent and, therefore, dynamism in its motivation is considered. Technically speaking, consistence among goals is presupposed, given that dealing with inconsistence is a hard issue, avoidable if the aim is just to achieve believable behaviors.

This *base of goals* contains goals with diverse scope of resolution. It is the planner at every layer (in this case, the deliberative planner) who will decide for which goals a plan may be built. It will trust other layers to manage the remaining goals. Thus, the goals managed at this level could be achieved by using only the agent's own information and skills.

Plans

Every time the deliberative planner builds a plan or subplan for one of the goals of the *goals base*, it is incorporated to the *plans base*. As far as different layers are building plans from the same set of goals, alternative plans could co-exist inside this set of plans.

It is important to note that plans are conceived taking into account the agent beliefs, history, and concerns. So, personal characteristics of the agent are considered in that process, and the kinds of plans resulting will show coherent behaviors, increasing believability.

These plans will be collected by the *scheduler*, which will sort them to compose the *agent's agenda*.

Social Layer

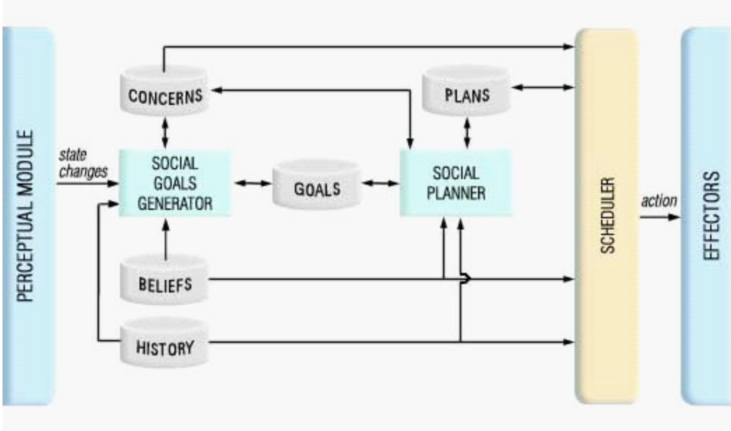
The upper layer of the proposed architecture for the cognitive module deals with the fact that VEs are usually crowded with avatars, and that those avatars once interacted among them. That is, this layer exploits the coincidence of the social nature of both agents and avatars in VEs.

From its general definition, an agent is able to solve a specific kind of problem autonomously. However, a social behavior—interactions, negotiations, cooperations—is considered a way of increasing the agent performance. It is also one alternative to achieving higher goals, out of the scope of every single agent.

Therefore, the first main difference of this layer regarding the deliberative one is its social nature, visible in the existence of components to generate goals and plans from a social point of view:

- **Social Goals Generator:** This goal generator has a similar functionality to its peer in the deliberative layer, but taking as input beliefs upon the other ones—their perceived personality traits, moods, physical state—and the attitude of the avatar towards them, along with memories about past interactions (history), other active goals, and current concerns. Thus, the goals generated—goals with an evident social nature—will be "personalized" for the agent and, therefore, closer to the goals expected for that agent by a human user. Naturally, the social goals generator possesses the same attributes as its deliberative peer with regards to proposal or modification of concerns and existing goals management.
- **Social Planner**: The aim of the social planner is to generate plans for every goal present in the goals base—note that, at the end, all goals, regardless of their origin, are maintained in a common goal base, allowing the different planners to propose alternative plans from their particular perspective, but always according to the agent's social skills. The plans built will be maintained in the same structure as the deliberative ones, as long as the scheduler is indifferent against the origin or nature of the plan.

Figure 14. Agent's social layer architecture



The proposed architecture for the social layer is outlined in Figure 14. There are obvious similarities between the general structure of this layer and the deliberative one, as far as their main differences lie in the data managed and the processes' internal working.

Conclusions

The design and implementation VEs is an intricate task. To the usual—and not trivial—difficulties of any big system, many particular characteristics of this kind of software (distributed nature, complex graphics design and handling, technology novelty) are added. To top it all, the need of increasing the system usability by automating some of its roles forces the inclusion of some sort of synthetic characters ruled by agents provided with an autonomous behavior.

This decision entails two additional difficulties: (1) the agent should verbatim substitute a human user, and all or a part—depending on the agent's degree of autonomy—of his/her perceptual, cognitive, and behavioral skills should be replicated or, at least, simulated; and (2) any human user inside the VE should feel everything perceived is believable, even those agent-ruled avatars, in order to improve his/her presence in the virtuality, one of the reasons of using this kind of system.

Therefore, the design of these IVAs should be supported by a believability-oriented solid architecture. The believability in an IVA must be strengthened in the design of all of its modules—perceptual module, cognitive module, and effectors—but the main charge lies in the first two.

Hence, the architecture proposed develops a human-like *perceptual model* for IVAs based on one of the most successful awareness models in computer-supported cooperative work (CSCW), called the Spatial Model of Interaction (SMI) (Benford & Fahlen, 1993). This perceptual model extends the key concepts of the SMI, introducing some factors typical from human being perception such as sense acuity, sense transition region, and filters, as well as reinterpretating the key concepts with the aim of using them as the key concepts of an IVA's human-like perceptual model.

This perceptual model also introduces a new concept, which has been called *clarity of perception* (CP), as a way of having a measurement of the ability to perceive an object inside the agent's area of perception, as well as the clearness of this perception that it is possible to get from it.

Unlike other perceptual proposals, limited to 2D environments and/or not facing up to some relevant human perceptual factors, the proposed perceptual model allows an IVA to perceive its 3D environment and surrounding objects in real-time with a human-like clarity of perception, giving it the chance to react to stimuli in its environment, as well as to respond to interactions with the real world, making it more believable.

With regards to the *cognitive process*, a three-layered horizontal architecture has been proposed, based on the most usual model of hybrid architectures for agents and extended with indispensable characteristics for IVAs. The architecture allows personalized behaviors in three layers of abstraction, from instinctive reactive behaviors to interactive social behaviors, also giving a chance to several kinds of planning processes.

For these purposes, the cognitive module makes use of diverse sources of information, each of them with its particular degree of trustworthiness. Thus, some concepts like beliefs, past history, goals, and plans are adapted to this kind of software. Also, the concept of concern is introduced as a key feature to personalize the behaviors and to allow an easy-to-use coordination mechanism between layers.

Still, most of the cognitive architectures proposed to date are focused on the understanding of the human reasoning processes, forgetting the significant influence of emotion and affection on human behavior. The proposed cognitive model allows one to take the input information filtered and processed by the perceptual module, and structure it to act reactive or proactively, exhibiting personalized and believable behaviors not affordable by pure rational processes.

References

Allen, S. (1999). Control states and motivated agency. In E. André (Ed.), *Behavior planning for life-like characters and avatars* (pp. 43-61). Spain: Sitges.

Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM, 37*(7), 122-125.

Bates, J., Loyall, A.B., & Reilly, W.S. (1994). *An architecture for action, emotion, and social behavior* (Volume 830). Berlin: Springer-Verlag.

Benford, S., & Fahlen, L.E. (1993). A spatial model of interaction in large virtual environments. *Proceedings of the Third European Conference on Computer Supported Cooperative Work* (ECSCW) (pp. 109-124). Milan, Italy: Kluwer Academic Publishers.

Binsted, K. (1999). Character design for soccer commentary. In M. Asada & H. Kitano (Eds.), *Lecture Notes on Artificial Intelligence, 1604,* 23-35. Berlin: Springer-Verlag.

Blumberg, B. (1997). Go with the flow: Synthetic vision for autonomous animated creatures. In W.L. Johnson & B. Hayes-Roth (Eds.), *Proceedings of the First International Conference on Autonomous Agents (Agents'97)* (pp. 538-539). New York: ACM Press.

Bratman, M.E., Israel, D., & Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence, 4*(4), 349-355.

Brooks, R.A. (1991). Intelligence without representation. *Artificial Intelligence, 47,* 139-159.

Chopra-Khullar, S., & Badler, N. I. (2001). Where to look? Automating some visual attending behaviors of human characters. *Publication, 4*(1/2), 9-23.

Cohn, A., Magee, D., Galata, A., Hogg, D., & Hazarika, S. (2003) Towards an architecture for cognitive vision using qualitative spatio-temporal representations and abduction. In C. Freksa, W. Brauer, C. Habel, & K.F. Wender (Eds.), *Spatial Cognition III, 2685,* 232-248.

Dourish, P., & Bellotti, V. (1992). Awareness and coordination in shared workspaces. *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (CSCW'92) (pp. 107-114). Toronto, Ontario: ACM Press.

Elliott, C. (1997). I picked up catapia and other stories: A multimodal approach to expressivity for "emotionally intelligent" agents. In W.L. Johnson & B. Hayes-Roth (Eds.), *Proceedings of the First International Conference on Autonomous Agents* (Agents'97) (pp. 451-457). New York: ACM Press.

Endsley, M. (1988). Design and evaluation for situation awareness enhancement. *Proceedings of Human Factors Society 32nd Annual Meeting* (pp. 97-101), Anaheim, California.

Endsley, M. (1993). *Towards a theory of situation awareness* (Tech. Rep.). Texas Technical University. Department of Industrial Engineering.

Genesereth, M.R., & Ketchpel, S.P. (1994). Software agents. *Communications of the ACM, 37*(7).

Georgeff, M., Pell, B., Pollack, M., Tambe, M., & Wooldridge, M. (1999). The belief-desire-intention model of agency. In J. Müller, M. P. Singh, & A.S. Rao (Eds.), *Proceedings of the 5th International Workshop on Intelligent Agents: Agent theories, architectures, and languages* (ATAL-98), *1555,* 1-10. Heidelberg, Germany: Springer-Verlag.

Greenhalgh, C. (1997). *Large scale collaborative virtual environments*. Unpublished doctoral dissertation, University of Nottingham, UK.

Hayes-Roth, B. (1995). An architecture for adaptive intelligent systems. *Artificial Intelligence: Special Issue on Agents and Interactivity, 72*, 329-365.

Hayes-Roth, B., Brownston, L., & Sincoff, E. (1995). *Directed improvisation by computer characters.* Technical Report No. KSL-95-04, Knowledge Systems Laboratory, Stanford University, Stanford, California, USA.

Herrero, P. (2003). *A human-like perceptual model for intelligent virtual agents*. Unpublished doctoral dissertation, Universidad Politécnica de Madrid, Spain.

Herrero, P., & de Antonio, A. (2002). A human=based perception model for cooperative intelligent virtual agents. *Proceedings of the Tenth International Conference on Cooperative Information Systems* (CoopIS'02). *Confederated International Conferences DOA/CoopIS/ODBASE 2002, 2519*, 195-212. Irvine, California: Springer-Verlag.

Herrero, P., & de Antonio, A. (2003a). A hearing perceptual model for intelligent virtual agents. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 733-740). Melbourne, Australia: ACM Press.

Herrero, P., & de Antonio, A. (2003b). Keeping watch: Intelligent virtual agents reflecting human-like perception in cooperative information systems. *Proceedings of the Eleventh International Conference on Cooperative Information Systems* (CoopIS'03). *Confederated International Conferences DOA/CoopIS/ODBASE 2003*. Catania, Sicily, Italy: Springer-Verlag.

Herrero, P., de Antonio, A., Benford, S., & Greenhalgh, C. (2002). Increasing the coherence between human beings and virtual agents. *Proceedings of* *the First International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 354-355). Bologna, Italy: ACM Press.

Hill, R., Han, C., & van Lent, M. (2002). Perceptually driven cognitive mapping of urban environments. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*. Bologna, Italy: ACM Press.

Howarth, P.A., & Costello, P.J. (1997). *Contemporary ergonomics* (pp. 109-116).

Imbert, R., & de Antonio, A. (2000). The bunny dilemma: Stepping between agents and avatars. In A. Nijholt, D. Heylen, & K. Jokinen (Eds.), *TWLT 17-CEvoLE 1 learning to behave. Proceedings of the 17th Workshop on Language Technology* (pp. 145-159), Enschede, Holland.

Imbert, R., de Antonio, A., & Segovia, J. (2001). Improving communication in 3D virtual environments by means of task delegation in agents. *Proceedings of the Fifth International Workshop on Cooperative Information Agents,* Modena, Italy.

Jennings, N.R., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, *1*(1), 7-38.

Levi, D., Klein, S., & Hariharan, S. (2002a). *Suppressive and facilitatory spatial interactions in foveal vision: Foveal crowding is simple contrast masking* (pp. 140-166).

Levi, D., Klein, S., & Hariharan, S. (2002b). *Suppressive and facilitatory spatial interactions in peripheral vision: Peripheral crowding is neither size invariant nor simple contrast masking* (pp. 167-177).

Loyall, A.B., & Bates, J. (1997). Personality-rich believable agents that use language. In W.L. Johnson & B. Hayes-Roth (Eds.), *Proceedings of the First International Conference on Autonomous Agents* (Agents'97) (pp. 106-113). Marina del Rey, CA: ACM Press.

Maes, P. (1994). Modeling adaptive autonomous agents. *Artificial Life I, 1&2*(9), 135-162.

Maes, P. (1995). Artificial life meets entertainment: Lifelike autonomous agents. *Communications of the ACM, 38*(11), 108-114.

Morningstar, C., & Farmer, F.R. (1990). The lessons of Lucasfilm's habitat. In M. Benedikt (Ed.), *Cyberspace: First steps* (pp. 273–301). Cambridge, MA: MIT Press.

Newell, A., & Simon, H.A. (1963). GPS: A program that simulates human thought. In E.A. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 279-293). New York: McGraw-Hill.

Noser, H. (1997). *A behavioral animation system based on l-systems and synthetic sensors for actors*. Unpublished doctoral dissertation, École Polytechnique Fédérale de Lausanne, France.

Ortony, A., Clore, G., & Collins, A. (1988). *The cognitive structure of emotions*. Cambridge, UK: Cambridge University Press.

Perlin, K., & Goldberg, A. (1996). Improv: A system for scripting interactive actors in virtual worlds. *Computer Graphics, 30*(Annual Conference Series), 205-216.

Rao, A.S., & Georgeff, M.P. (1995). BDI agents: From theory to practice. In V. Lesser (Ed.), *Proceedings of the First International Conference on Multi-Agent Systems* (ICMAS-95) (pp. 312-319). San Francisco: MIT Press.

Reilly, W.S. (1997). A methodology for building believable social agents. In W.L. Johnson & B. Hayes-Roth (Eds.), *Proceedings of the First International Conference on Autonomous Agents* (Agents'97) (pp. 114-121). Marina del Rey, CA: ACM Press.

Reilly, W.S., & Bates, J. (1995). *Natural negotiation for believable agents.* Technical Report No. CMU-CS-95-164, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Rousseau, D., & Hayes-Roth, B. (1997). *Improvisational synthetic actors with flexible personalities.* Technical Report No. KSL 97-10, Knowledge Systems Laboratory, Computer Science Department, Stanford University, Stanford, California, USA.

Russell, S.J., Subramanian, D., & Parr, R. (1993). Provably bounded optimal agents. In R. Bajcsy (Ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (IJCAI-93) (pp. 338-344), Chambéry, France. San Mateo, CA: Morgan Kaufmann.

Shively, R.J., Brickner, M., & Silbiger, J. (1997). *A computational model of situational awareness instantiated in MIDAS.* Technical Report.

Sloman, A., & Logan, B. (1998). Cognition and affect: Architectures and tools. In K.P. Sycara & M. Wooldridge (Eds.), *Proceedings of the 2nd International Conference on Autonomous Agents* (Agents'98) (pp. 471-472). New York: ACM Press.

Sycara, K. (1998). Multiagent systems. *AI Magazine*, *19*(2), 79-92.

Terzopoulos, D., & Rabie, T. (1995). *Animat vision: Active vision in artificial animals* (pp. 801-808).

Thalmann, D. (2001). The foundations to build a virtual human society. *Proceedings of the 3rd International Workshop on Intelligent Virtual Agents, 2190,* 1-14, Madrid, Spain.

Vilhjálmsson, H.H. (1997). *Autonomous communicative behaviors in avatars*. Unpublished master's thesis, MIT, Cambridge, Massachusetts, USA.

Wooldridge, M. (2000). *Reasoning about rational agents*. Cambridge, Massachusetts: MIT Press.

Wooldridge, M. (2002). *An introduction to multi-agent systems*. Chichester, UK: John Wiley & Sons.

Wooldridge, M., & Jennings, N. (1994). Agent theories, architectures, and languages: A survey. In M. Wooldridge & N.R. Jennings (Eds.), *Intelligent agents—theories, architectures, and languages. Proceedings of ECAI'94 Workshop on Agent Theories, Architectures & Languages*, *890*, 1-32. Amsterdam: Springer-Verlag Lecture Notes in Artificial Intelligence.

Wright, I.P. (1997). *Emotional agents*. Unpublished doctoral dissertation, University of Birmingham, Birmingham, UK.

Yeap, W.K., & Jefferies M.E. (1999) Computing a representation of the local environment. *Artificial Intelligence*, *107*(2), 265-301.

Chapter VIII

An Agent-Based Architecture for Virtual Environments for Training

Angélica de Antonio
Universidad Politécnica de Madrid, Spain

Jaime Ramírez
Universidad Politécnica de Madrid, Spain

Gonzalo Méndez
Universidad Politécnica de Madrid, Spain

Abstract

This chapter proposes an architecture for the development of intelligent virtual environments for training (IVETs) which is based on a collection of cooperative software agents. The first level of the architecture is defined as an extension of the classical intelligent tutoring system architecture that adds a new world module. Several software agents are then identified within each module. They communicate among them directly via messages *and indirectly via a common data structure that is used for the collaborative development of plans. Some details are provided about the most remarkable interactions that will be established among agents during the system's execution. The proposed architecture, and its realization in a platform of generic and configurable agents, will facilitate the design and implementation of new IVETs, maximizing the reuse of existing components and the extensibility of the system to add new functionalities.*

Introduction

Computer-based training is a promising application area of three-dimensional virtual environments (VEs). These environments allow the students to navigate through and interact with a virtual representation of a real environment in which they have to learn to carry out a certain task. They are especially useful in situations where the real environment is not available for training, or it is very costly or risky. A good example is training of nuclear power plant operators. A multi-user virtual environment also allows for team training. An intelligent virtual environment for training (IVET) results from the combination of a virtual environment and an intelligent tutoring system (ITS). IVETs are able to supervise the actions of the students and provide tutoring feedback. The intelligent tutoring component of an IVET usually adopts a virtual representation (a pedagogical virtual agent) that inhabits the environment together with the virtual representations of the students (avatars).

The development of three-dimensional virtual environments has a quite short history, dating from the beginning of the '90s. The youth of the field, together with the complexity and variety of the technologies involved, have led to a situation in which neither the software architectures nor the development processes have been standardized yet. Therefore, almost every new system is developed from scratch, in an ad-hoc way, with very particular solutions and monolithic architectures, and in many cases forgetting the principles and techniques of the software engineering discipline (Munro, Surmon, Johnson, Pizzini, & Walker, 1999). Some of the proposed architectures deal only partially with the problem, since they are centered on a specific aspect like the visualization of the VE (Alpdemir & Zobel, 1998; Demyunck, Broeckhove, & Arickx, 1999) or the interaction devices and hardware (Darken, Tonessen, Passarella, & Jones, 1995).

As a result, current VEs lack many of the desirable quality attributes of any software system, such as flexibility, reusability, maintainability, or interoperability.

The size and complexity of VEs will continue to increase in the future, making this situation even worse. Many researchers and developers of VEs are starting to recognize the need of a software engineering approach to the development of VEs (Brown, Encarnaçao, & Schneiderman, 1999; Fencott, 1999; Sánchez, 2001).

In particular, there is a need to define "standard" architectures in order to facilitate the development of individual components by different teams or organizations that may have different skills and knowledge. The development of a new VE will then consist of the selection and adaptation of existing components, and their assembly and integration. In this way, components will be reusable, the system will be flexible to be extended with new components, and the interfaces among components will be clearly defined to facilitate interoperability of possibly very heterogeneous components.

Unfortunately, we are still very far from this ideal state. Given the broad variety and diversity of VEs and their applications, and taking into account that they may require different architectures, we have decided to restrict the scope of our research to a certain type of VEs, namely virtual environments for training (VETs).

The goal of this kind of VEs is to train one or more students in the execution of a certain task. They are especially useful in those situations in which training in the real environment is either impossible or undesirable because it is costly or dangerous. Let's consider as an example training the operators of a nuclear power plant in the execution of maintenance interventions. In the real environment the trainees would be subject to radiation, which is of course unacceptable for their health, and additionally it would be impossible to reproduce some maintenance interventions without interfering with the normal operation of the plant.

In VETs, the supervision of the learning process can be performed by human tutors or by intelligent software tutors, also known as pedagogical agents (in this case we will call it an IVET). Those pedagogical agents, in turn, can be embodied and inhabit the virtual environment together with the students or they can be just a piece of software that interacts with the student via voice, text, or a graphical user interface. Some pedagogical agents have been developed to date, in some cases with quite advanced tutoring capabilities. One of the best known is *Steve*, developed in the Center for Advanced Research in Technology for Education (CARTE) of the Intelligent Systems Division of the University of Southern California (USC) (Rickel & Johnson, 1999, 2000).

Recently, we conducted an experiment (Méndez, Rickel, & de Antonio, 2003) in reusing Steve, in a new virtual world, *HeSPI*, which was developed independently. Steve was carefully designed to be easy to be applied to new domains and virtual worlds. It was originally applied to equipment operation and maintenance training on board a virtual ship. Subsequently, it was significantly extended and applied to leadership training in virtual Bosnia. However, the leadership training application was designed with Steve in mind. In contrast, HeSPI (de Antonio, Ferre, & Ramirez, 2003) was developed independently as a tool for planning maintenance interventions in nuclear power plants. HeSPI's virtual avatars were developed using *Jack*, a human simulation tool distributed by EDS. HeSPI's design was also carefully thought out so that different user interfaces could be easily connected to the system. Two user interfaces were developed, one of them using voice recognition and conventional mouse, keyboard, and monitor, and another one using a 3D mouse and shutter glasses. Thus, HeSPI+Steve looked like a good test bed for evaluating the degree in which nowadays VEs and pedagogical agents can be easily ported and made them interoperate.

All through the integration, we encountered several problems. For example, there were undesired behaviors due to the fact that both Steve and HeSPI performed redundant actions. There were control and synchronization problems. Steve required some information that HeSPI could not provide and vice versa. Many of these difficulties stemmed from the fact that their underlying architectures and their external interfaces were not totally compatible. Our conclusion from this experiment was that there is effectively a need for standard architectures designed to facilitate reusability, extensibility, and interoperability among components.

Component-based standard architectures in the field of educational software are not new. In the last few years, we have witnessed the activity of several standardization organizations all around the world, like the IEEE Learning Technology Standards Committee (LTSC) (www.ltsc.ieee.org), which proposed the Learning Objects Meta-data (LOM) Specification; the IMS Global Learning Consortium Inc. (www.imsglobal.org), creators of the Learning Resource Meta-Data Information Model; ARIADNE (Alliance of Remote Instructional Authoring and Distribution Networks for Europe (www. ariadne-eu.org); or the AICC (Aviation Industry Computer-Based Training Committee, www.aicc.org), authors of the Computer Managed Instruction (CMI) Specifications, Course Structure Format, and CMI Data Model. These standards and specifications were further integrated by the ADL (Advance Distributed Learning) co-Laboratory, created by a U.S. initiative, to give birth to SCORM (Sharable Content Object Reference Model, www.adlnet.org), which is increasingly being recognized as an international standard for e-learning applications.

However, these standards have never taken into account the special characteristics and needs of educational software based on VEs. They are oriented towards Web-based e-learning courses in which the interactivity with the student is restricted to navigating through the materials (Web pages) and answering tests.

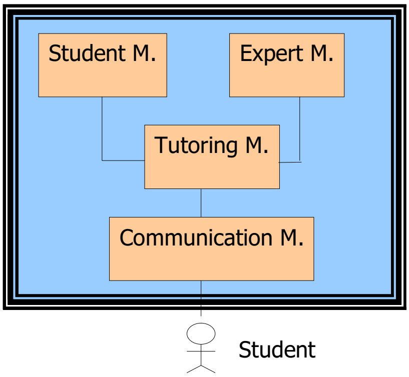
The work that we present in this chapter is a first step towards the goal of defining standard architectures for IVETs.

An Architecture for VETs Based on the Architecture of Intelligent Tutoring Systems

Our approach to the definition of an architecture for VETs is based on the agent paradigm, as opposed to the SCORM and other standards approach that is based on the object-oriented paradigm. The rationale behind this choice is our belief that the design of highly interactive VETs populated by intelligent and autonomous or semi-autonomous entities, in addition to one or more avatars controlled by users, requires higher level software abstractions. Objects and components (CORBA or COM-like components) are passive software entities that are not able to exhibit the kind of pro-activity and reactivity that is required in highly interactive environments. Agents, moreover, are less dependent on other components than objects. An agent that provides a given service can be replaced by any other agent providing the same service, or they can even co-exist, without having to recompile or even to reinitiate the system. New agents can be added dynamically providing new functionalities. Extensibility is one of the most powerful features of agent technology. The way in which agents are designed makes them easier to be reused than objects.

Our work draws from the results obtained in the MAPI project ("Modelo Basado en Agentes Cooperativos para Sistemas Inteligentes de Tutoría con Planificación Instructiva," funded by CICYT from 1996 to 1999) and is currently being funded by the Spanish Ministry of Science and Technology through project MAEVIF (TIC00-1346). In MAPI we designed an architecture based on cooperative agents for the tutoring component of intelligent tutoring systems in which

Figure 1. Architecture of an ITS



communication among agents took place through a shared blackboard structure. Starting from the idea that a VET can be seen as an ITS (an IVET), and the pedagogical agent in an IVET can be seen as an embodiment of the tutoring module of an ITS, our first approach towards extending the MAPI architecture for IVETs was to define an agent for each of the four modules of the generic architecture of an ITS (see Figure 1).

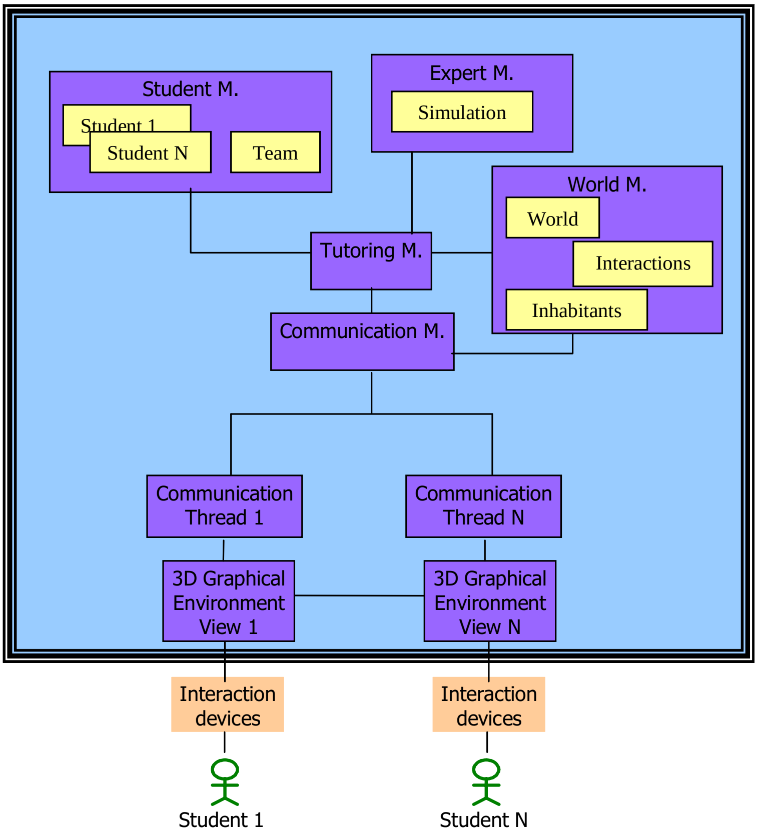
The ITS architecture, however, does not fit well with the requirements of IVETs in several aspects:

- IVETs are usually populated by more than one student, and they are frequently used for team training. An ITS is intended to adapt the teaching and learning process to the needs of every individual student, but they interact with the system one at a time. However, in a multi-student IVET, the systems would have to adapt both to the characteristics of each individual student and to the characteristics of the team.
- The student module should model the knowledge of each individual student but also the collective knowledge of the team.
- The student is not really out of the limits of the ITS, but immersed in it. The student interacts with the IVET by manipulating an avatar within the IVET, possibly using very complex virtual reality devices such as HMDs (head mounted displays), data gloves, or motion tracking systems. Furthermore, each student has a different view of the VE depending on their location within it.
- The communication module in an ITS is usually realized by means of a GUI or a natural language interface that allows the student to communicate with the system. It would be quite intuitive to consider that the 3D graphics model is the communication module of an IVET. However there is a fundamental difference among them. In an IVET some of the learning goals may be directly related to the manipulation and interaction with the 3D environment, while the communication module of a classical ITS is just a means, not an end. For instance, a nuclear power plant operator in an IVET may have to learn that in order to open a valve, he has to walk to the control panel, which is located in the control room, and press a certain button. Therefore, the ITS needs to have explicit knowledge about the 3D VE, its state, and the possibilities of interaction with it.

As a first step we decided to modify and extend the ITS architecture by considering some additional modules (see Figure 2).

First of all, we split the communication module into a set of different views for all the students, plus a particular communication thread for each student and a

Figure 2. Extended ITS architecture for IVETs



centralized communication module to integrate the different communication threads. Then we added a world module, which contains geometrical and semantical information about the 3D graphical representation of the VE and its inhabitants, as well as information about the interaction possibilities. The tutoring module is unique to be able to make decisions that affect all the students as well as tutoring decisions specific to a certain student. The expert module will contain all the necessary data and inference rules to maintain a simulation of the behavior of the system that is represented through the VE (for example, the behavior of a nuclear power plant). The student module, finally, will contain an individual model for each student as well as a model of the team.

An Agent-Based Architecture for IVETs

Taking the extended architecture of the previous section as a reference, the next step was to decide which software agents would be necessary to transform this component-oriented architecture into an agent-oriented architecture. In an agent-oriented architecture, each agent is capable of performing a certain set of tasks, and is capable of communicating with other agents to cooperate with them in the execution of those tasks.

Our agent-based architecture has five principal agents corresponding to the five key modules of the extended ITS architecture:

- A Communication Agent
- A Student Modeling Agent
- A World Agent
- An Expert Agent
- A Tutoring Agent

Each of these principal agents may relate to, communicate with, and delegate some tasks to other subordinate agents, giving rise to a multi-level agent architecture.

In this way, the communication agent will delegate on a set of individual communication agents dedicated to each student. The students can choose among several interface devices for the interaction with the environment, ranging from the simple monitor + mouse + keyboard combination to the most complex and immersive virtual reality combination: head mounted display + motion tracking + data glove + voice recognition. There is a set of device agents to manage the different devices that can be used to interact with the environment and make the system independent of any specific combination of interaction devices. There is also a connection manager agent that is responsible for coordinating the connections of the students.

The student modeling agent is assisted by a:

- *Historic Agent*, responsible for registering the history of interactions among the students and the system.
- *Psychological Agent*, responsible for building a psychological profile of each student, including their learning style, attentiveness, and other personality traits, moods, and emotions that may be interesting for adapting the teaching process.

- *Knowledge Modeling Agent,* responsible for building a model of the student's current knowledge and its evolution.
- *Cognitive Diagnostic Agent,* responsible for trying to determine the causes of the student's mistakes.

The world agent is related to the:

- *Objects and Inhabitants Information Agent,* which has geometrical and semantical knowledge about the objects and the inhabitants of the world. This agent, for instance, will be able to answer questions about the location of the objects or their utility.
- *Interaction Agent,* which has knowledge about the possible interactions with the environment and the effects of those interactions. For instance, it will be able to answer questions like, "What will it happen if I push this button?"
- *Path Planning Agent,* which is capable of finding paths to move along the environment without colliding with objects or walls.

The expert agent, in turn, is related to other agents that are specialists in solving problems related to the subject matter that is being taught to the students. This is one of most variable components in an IVET. Underlying the virtual environment, one or more simulation agents are in charge of simulating the behavior of the system that is represented through the virtual environment (for example, the behavior of the nuclear power plant). In many IVETs the goal of the system is to train students in the execution of procedures. In our prototype for nuclear power plants, for instance, the goal is to teach a team of operators to execute some maintenance procedures. In this case, the expert agent should be able to find the best procedure to solve a given malfunctioning situation, and this is achieved by a planning agent that is able to apply intelligent planning techniques like STRIPS. If the IVET was to be used for teaching Chemistry, for instance, the expert agent should have knowledge about the chemical elements and should be able to plan and simulate reactions.

The tutoring agent, finally, will be assisted by a:

- *Curriculum Agent,* which has knowledge of the curricular structure of the subject matter.
- *Several Tutoring Strategy Agents,* which implement different tutoring strategies.

Figure 3. Agent-based architecture for IVETs

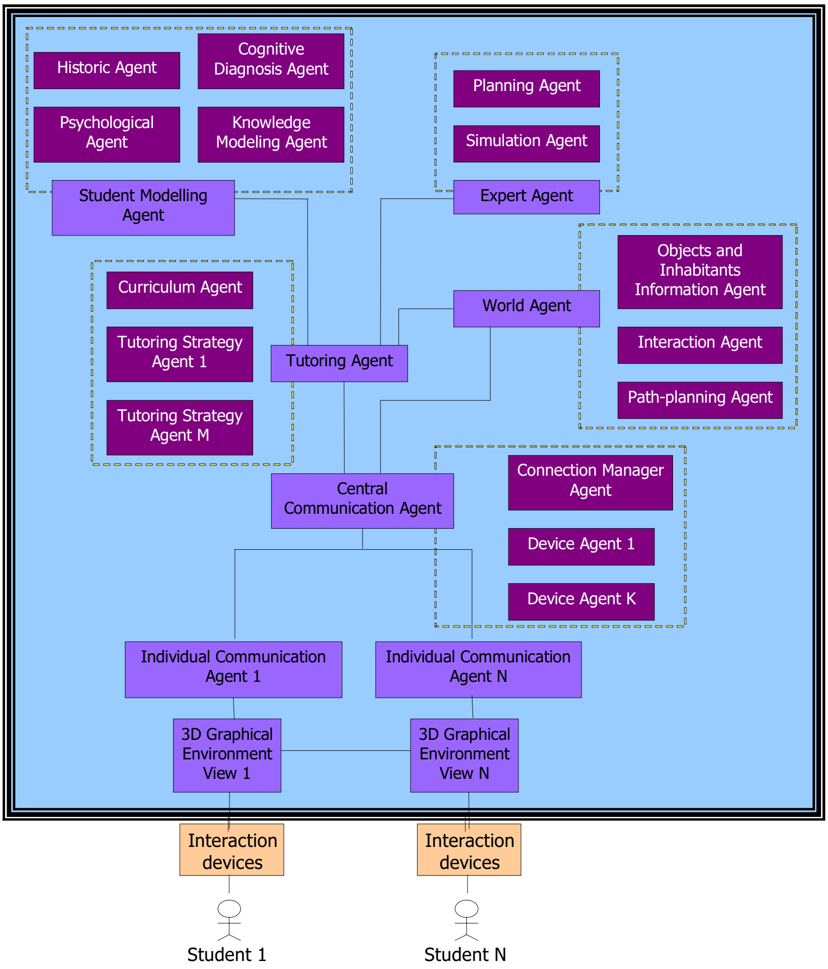


Figure 3 shows how the extended ITS architecture is transformed from a modular point of view to an agent-based architecture.

In the next sections we discuss in more detail some of the more important aspects and functionalities that have to be considered in the design of an IVET, and how we have dealt with them in our agent-based architecture. Throughout these sections more details will be provided about the role played by some of the agents and the interactions that will be established among different agents.

Management of Multiple Views

The fact that multiple users will be simultaneously connected to the system poses interesting challenges to the system architecture. As we mentioned before, each student will be provided with a particular view of the VE. We assume that each student will be represented in the environment by a graphical avatar, and his/her point of view will be located on the avatar's eyes. The interaction of the student with the environment will be performed mainly with his/her hands (pushing buttons, picking up objects, etc.). In our prototypes we have decided that one student will only see the hand/s of his/her avatar, while he/she will see the full body of the other students' avatars.

In order to build and update a given 3D graphical environment view, we need several essential pieces of information:

- The position and orientation of the student within the VE
- The direction of the student's gaze
- The position and gesture of the student's hand/s
- The position of other students
- The actions performed by other students

One possibility for dealing with these information requirements in the system architecture is to have a centralized component that collects this information from every student, builds a common representation of the environment, and then sends to each student's view the updates. This task could be performed by the central communication agent in the architecture of Figure 3. The problem is that the centralized component becomes a bottleneck and has to deal with synchronization and consistency problems. Another possibility is to have all the students' views communicate directly among them. Whenever an important event occurs in one of the views, the relevant information is broadcast to the other views. This option is illustrated by the link among the 3D graphical environment views in Figures 2 and 3, and it is the one that we have chosen to optimize the performance of the system.

Individuality vs. Collectiveness

Having multiple simultaneous users raises a new question that needs to be addressed in the design of an IVET: the degree of individuality versus collectiveness that will be allowed in the tutoring component. In the most general case, we can find a system that can be used both and simultaneously by individual students and by teams. In our prototype for nuclear power plant operators, for instance, there must be a first phase in the training in which each individual student uses the system to acquire or confirm general knowledge about radiation and radioprotection. Then, several activities can be posed that require the intervention of a team of operators. Two issues have to be taken into account in this kind of situation:

- *First, how to coordinate the different students that conform a team, so that one team activity can be initiated.* Different students may be located on different places, they may connect to the IVET at different times, and their learning process may be on different stages. On the other hand, different students may have different profiles and learning goals. For instance, one operator may be learning to operate a crane while another operator may be learning to measure radiation.
- *Second, how the tutoring component is going to supervise the activity of individual students and teams.* Are all of them going to share a tutor or is it going to be only one tutor for all the students? And what is the effect of this decision on the architecture of the system?

In order to solve the first question, the curriculum agent must know, for each collective activity, the number of students involved and the role to be performed by each of them. In turn, the knowledge modeling agent must have knowledge about the learning profiles or roles of each student. Then, when a new student tries to connect to the IVET, the connection manager agent asks for the name or identification of the student and informs the tutoring agent. If the tutoring agent decides that some students are ready to learn a certain activity, it asks the curriculum agent for the number and roles of the participants involved in the activity. Each student must choose one participant with one of his/her roles. Then we can have two possibilities:

- wait for the required number of students with the proper roles to be connected to the system,
- substitute any missing student with a student role agent that is able to play that role.

In the first case, the connection manager agent is endowed with the goal of registering newly connected students for the pending activity, after checking with the tutoring agent if they are ready to learn that activity, and informing the tutoring agent as soon as the required number of students with the proper roles is connected.

This solution has the disadvantage of having to adapt the learning speed of one student to the others', but the advantage of promoting real team learning and social exchange among the students.

The second option may be advisable if the learning speed of the students is very different or if there is a role in one activity for which there is not any student enrolled in the course.

Regarding the issue of having only one or many tutors, we believe that having many embodied tutors moving around the environment, one for each student, may be very disturbing. On the other hand, each student should receive individualized advice. Having only one embodied tutor that has to supervise and talk to many students may imply that the tutor is all the time running from one student to another. We have chosen an intermediate solution. Since each student has a particularized view of the system, it is possible to show something only in one student's view but not in the others'. Then, each student will only see one tutor dedicated to him/her. The tutor will follow the student along the VE, it will observe the student's actions, and it will talk to him. For one student it will look as if the other students were not being supervised.

Problem Solving within the Environment

The main kind of learning activity in an IVET consists of the tutoring agent describing an initial state of the system to the team of learners and asking them to find a way to reach a desired goal. For instance, the tutoring agent may ask the team to stop the reactor, or to change a contaminated filter. Solving the problem will require each of the team members to execute a certain set of actions in an appropriate order. As an example, changing the filter requires a cleaning operator to enter the controlled area and clean the surroundings, a radioprotection operator to measure the radiation level close to the filter at certain points during the change procedure, a couple of mechanical operators to disassemble the filter cartridge and extract the filter, and so on.

A straightforward solution for the expert agent is to have a predefined plan or sequence of actions for each possible problem. The tutoring agent will then have to check whether the students' actions adjust to the plan or not. However, this solution restricts the number of possible problems that the tutoring agent can pose to the student to the ones that have predefined solution plans. A more critical drawback of this solution is that many times different plans may be valid to reach the desired goal (even if they are not equally optimum in terms of time spent or radiation exposure, for instance). Whenever a student executes an action that was not in the predefined solution plan, the system should be able to determine whether it is possible to reach the desired goal from the resulting state or not. The kind of tutoring action to be taken greatly depends on this. The only way to make the system flexible enough to deal with a possibly unlimited number of problems and with unpredicted student actions is to provide the expert agent with the capability of finding the solution for any problem in real time. The expert in solving planning problems will be the planning agent (a planning problem can be defined as finding an optimal sequence of actions to reach a desired goal state from a given initial state).

Initially, the tutoring agent is interested in finding out a plan to reach a certain final state in the VE from the current state. The plan consists of a sequence of actions that the student can perform in the VE. Three kinds of agents will be involved in the planning process: the tutoring agent, the planning agent, and what we call action agents. Each action agent is specialized in a certain set of goals, so that it knows one or more actions that can satisfy each one of goals belonging to this set. As a working hypothesis, we assume there is not more than one action agent that can satisfy a goal (Hypothesis 1). In our system, we have three action agents: the simulation agent, the path-planning agent, and the interaction agent. The interaction among these agents will be carried out by means of a shared blackboard and asynchronous message passing. During the planning process, the planning agent will coordinate action agents.

The path-planning agent can determine whether the avatar that models the student in the VE can walk from a position of the VE to another position of the VE. Hence, this agent will be in charge of satisfying goals of the type *Is_In_Position(X, Y)*, and for that it will use the action $Move\_To((X_i, Y_i), (X_f, Y_f))$. Although the VE is a 3D virtual world, the displacements of the avatar will always be done over a floor or plane. For that reason, we will only use two coordinates to specify the position of the avatar. Besides, as we assume it is always possible to move the mannequin from a position to any one in the VE, the preconditions of the operator $Move\_To((X_i, Y_i), (X_f, Y_f))$ are true.

In addition, the actions of the interaction agent are the basic actions that the avatar can do in the VE except for moving from one position to another, for instance, press a button, pick up an object, insert a card in a card reader, etc. In order to make hierarchical planning possible, some basic actions can be grouped into tasks or higher-level actions. The simulation agent will use the latter type of actions. Let's see an example to clarify the difference between basic actions and tasks. We suppose it is necessary to raise the temperature of a reactor in a nuclear power plant, and to carry out that, an operator must go to the control room and press a certain button. We consider *raise the temperature of the reactor* to be a task, and *go to the control room* and *press button X* to be basic actions.

The planning process is inspired in the STRIPS algorithm (Fikes & Nilsson, 1971). However, our planner accomplishes a breadth search in the state space instead of a depth search; the reason for this is that we are not interested in obtaining any plan, but the best plan. To implement the breadth search, the planning process will maintain a search tree in which each node will include a stack of goals and actions in STRIPS style, a state, and the plan to reach this state (see Figure 4). In some domains, the computational cost of building such a search tree may be too high; therefore we will work with domains in which the size of the state space is manageable. In order to allow the agents to carry out concurrent operations over the search tree, the tree is encapsulated in a blackboard.

The planning process will begin when the tutoring agent introduces in the blackboard an empty parent node, and a child node for each different order of the goals that describe the desired final state and are not in the initial state. In addition, each child node will contain the description of the initial state and an empty plan. Then, the tutoring agent asks the planning agent to begin the planning process. Next, the planning agent notifies the action agents that there are new goals in the blackboard (the tops of the stacks in the leaf nodes of the tree). We call these goals *active goals*. Now, the action agents read all the active goals, and check whether they can satisfy any of them by using one of the actions that they know. It is noteworthy to mention that these read operations can be executed concurrently. If an action agent can satisfy an active goal by means of an action, this agent will carry out the following steps:

1. Add child nodes to the node that comprises the active goal; a different child node will be created for each possible different order of the preconditions of the action. If any child node is already present in the search tree, it must not be added to the tree.
2. For each child node:
   2.1. The ground action, the ground conjunction of the action preconditions and a different order of the ground preconditions are pushed in its stack.
   2.2. Check whether the goal in the top of the stack holds in the state included in the node. If it does, the goal is deleted, and Step 2 is repeated if the top of the stack is a goal. Otherwise, if the goal does not hold, Step 3 is executed.
   2.3. If the top of the stack is a conjunction, check whether the conjunction holds in the state included in the node. If it does, the conjunction is deleted. Otherwise, a *fail node notification* is sent to the planning agent, and this operation ends.

  2.4. If the top of the stack is a ground action, the action is removed from the stack, it is introduced in the plan of the node, and it is applied to the state of the node; next, go to Step 2.2.

3. Notify the planning agent of this operation (*satisfied goal notification*).

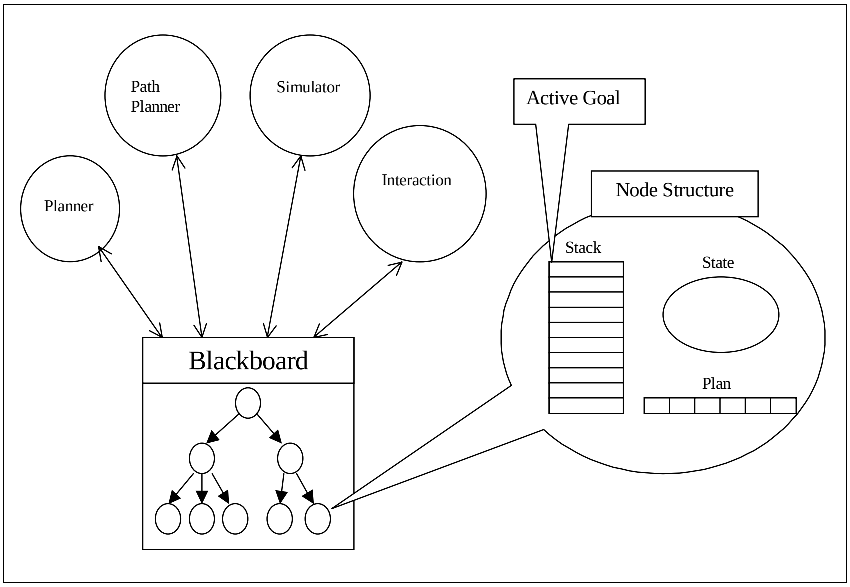
In Step 2.4 we must distinguish a particular case related to the action *Move_To*((X_i, Y_i), (X_f, Y_f)). This action will not be completely grounded in the stack, but only the latter two arguments will be constants. This is due to the fact that this action is used to satisfy a goal with the scheme *Is_In_Position(A, B)*, so X_f/A and Y_f/B. In order to bind the free variables of the action before introducing it in the plan, the plan will be examined to find the last bound action *Move_To* included in the plan. From this bound action, the latter two arguments (constants) will be extracted, and they will be used to bind the free variables. Otherwise, if there is not a bound action *Move_To* in the plan, the initial position of the avatar will be used to bind the free variables.

Thanks to Hypothesis 1, several agents may execute these steps over the blackboard concurrently because different leaf nodes are involved. On the other hand, if an action agent cannot satisfy a certain active goal, the agent will report it to the planning agent. In this way, the planning agent will be able to find out about whether there is a node in the tree whose active goal cannot be satisfied by any agent (*fail node*). In this case, the planning agent will delete the fail node, performing the backtracking; if the deleted node is the last not-empty node, the planning agent will detect the initial problem has no solution. When the planning agent receives a satisfied goal notification, it will tell action agents that the set of active goals has been modified. Then, as soon as possible, the action agents must read the new set of active goals. It could happen that an action agent is using an obsolete set of active goals. However, it is not difficult to see this situation will not be problematic due to Hypothesis 1. Moreover, the planning agent must be able to notice the search tree has been completed. In that situation, the set of active goals is empty; hence, when the action agents try to obtain this set, after receiving a satisfied goal notification, they will obtain an empty set. Then, they have to notify the planning agent of this event, so that this agent will be able to determine the best plan for the initial problem. Finally, the planning agent will report the best plan to the tutoring agent.

According to the explanation presented above, the blackboard must support four operations:

- Initialize the blackboard.
- Obtain the active goals that were not examined by a certain agent yet.
- Satisfy some active goals: for each goal, a ground action must be provided.
- Obtain the best plan.

Figure 4. Planning agents and blackboard



Path Planning

After obtaining the plan, the tutoring agent will have the sequence of actions that each student has to carry out in the VE to solve a problem. However, the tutoring agent will also need to know the best trajectory for each displacement that the avatar must carry out in the VE. In this way, the tutoring agent will be able to check whether the trajectory followed by the student in the VE is acceptable, that is, whether it is close enough to the best trajectory for each movement. The agents in charge of generating the trajectories will be the path-planning agent and the objects and inhabitants information agent.

The objects and inhabitants information agent contains a geometrical model of the VE expressed by means of several graphs. These graphs will have been obtained prior to the planning process from geometrical information related to the VE.

We assume the VE to be divided into several rooms (sub-environments) joined by doors. Then, we use a graph to model the accessibility among the rooms (*environment graph*). Furthermore, we use a different graph for each room to model the accessibility of the different positions (*room graph*). In order to obtain the room graphs, we model each room as a 2D grid in which each free square corresponds to a node of the graph. We decide whether a square is free by using the projection over the floor of all the objects existing in the room. In this way, if a part of the projection of an object is inside a square of the grid, this square will not be free. Besides, an edge between two nodes A and B is added to the graph if the two squares associated to the nodes share a side in the grid. The size of the squares must be adjusted correctly. Otherwise, if the squares are too big, some feasible trajectories may be considered wrong; or if the squares are too small, the computational cost of the search process that will be explained later will increase unnecessarily. In addition, the process to obtain the graphs from the 3D information of the VE can be complete automatically.

The trajectory for each 2D movement included in the plan will be determined after working out the whole plan. Before starting to generate the trajectory for a certain 2D movement, it will be necessary to update the graphs according to the actions to be executed prior to the 2D movement in the plan. For example, if the position of a table is changed, the previous position of the table may be traversed by a trajectory afterwards. As the planning agent does not know the semantics of the actions, it will tell the interaction agent to decide if it is necessary to update the graphs according to each bound action. In turn, the interaction agent will notify the objects and inhabitants information agent of the changes in the position of the objects in the 3D world, so that the objects and inhabitants information agent can modify the graphs.

After the update of the graphs, the path-planning agent must find out whether the movement traverses more than one room. If it does, the path-planning agent will use the environment graph to obtain, if it exists, the sequence of rooms that the avatar must traverse from his initial position to his final position. Next, the trajectory across each room must be obtained. For this, the room graphs will be used. If the movement traverses just one room, the unique graph to use will be the graph associated to this room. In order to work out the path between two nodes in the graphs, the A* algorithm (Hart, Nilsson, & Raphael, 1968) is applied under the assumption that all the edges' weights are one, and using as heuristic function the Euclidean distance. Each time the path-planning agent needs to know the nodes directly connected to a node in a graph, it will ask it to the objects and inhabitants information agent.

The A* algorithm outputs a sequence of nodes that the path-planning agent will translate into a sequence of points or trajectory in the VE with the help of the objects and inhabitants information agent. This trajectory will be saved into an XML file, so that the tutoring agent can employ this information to supervise the movements of the students later on.

Coordination Among Agents During the Supervision Stage

Once the expert solution for a given activity has been calculated, the IVET enters the supervision stage, in which:

- The individual communication agents will observe the behavior of the different team members and will inform the tutoring agent about it through the central communication agent.
- The tutoring agent will compare the real behavior to the expected behavior provided by the expert agent, and it will evaluate the adequacy of each student's behavior.
- The historic agent will register the actions performed by each student.
- The knowledge modeling agent will infer and model the state of knowledge of each student, with the help of the cognitive diagnostic agent in case of errors.
- The psychological agent will use the behavior of the student to infer psychological characteristics.
- The tutoring strategy agent will decide on the next step to be taken, considering the last actions of the student (provided by the historic agent), his/her state of knowledge (provided by the knowledge modeling agent), his/her psychological state (provided by the psychological agent), and the learning objectives and structure of the subject matter (provided by the curriculum agent). The decision might be to wait for new actions of the student, to use the embodied tutor to give a hint, to congratulate the student, to explain why something was wrong, and so forth.

Development of a New IVET

One of the advantages of the proposed architecture is that it has allowed us to build a basic infrastructure of agents that work as a runtime engine. In order to develop a new IVET, it will be the author's responsibility: to select the desired agents among the available ones (for instance selecting the tutoring strategy agent that implements the desired tutoring strategy); to configure the parameters that govern the behavior of those agents (for instance the duration of the session, the number of mistakes that will be allowed before the tutoring agent tells the student the correct answer, etc.); to provide the data specific to the new IVET and subject matter (the geometrical model of the VE, the curriculum, the actions that are possible in the new VE and their effects on the simulation, etc.); and in the worst case to create new agents and register them in the platform.

The requirements for the new IVET under development should be driven by real-world studies (Economou, Mitchell, Pettifer, Cook, & Marsh, 2001), and these requirements will drive, in turn, the design decisions regarding the configuration and adaptation of agents.

As a prototype application of our tool, we have developed a training system for nuclear power plant operators. We had previously developed this system from scratch in 1999, during a one-year period. The re-development using our infrastructure has taken just a few weeks, and the achieved functionality is superior. For instance, the previous implementation was for only one user, the tutor was not embodied, and the communication tutor-student was restricted to correction feedback. The decrease in development time and the increase in functionality suggest that we have successfully achieved our aim. A thorough experimental evaluation of the effectiveness of the solution is out of the scope of this chapter.

Practical Realization

The agent-based architecture for VETs that was described in the previous sections has been implemented with a combination of quite heterogeneous technologies. The agents have been implemented in Java, and the direct communication among them has been realized using the Jade platform with FIPA ACL messages. The 3D VE and avatars have been modeled with 3D Studio Max and imported into OpenGL format with a specific tool developed for that purpose. The visualization of the 3D models, animations, and interactions are managed by a program in C++, making use of the OpenGL graphical library. Microsoft's DirectPlay library has been used for direct communication among the different 3D graphical environment views in order to take into account the movements and actions of the other students for real-time update of each view. Microsoft's DirectInput has been used to manage some interaction devices, namely the mouse, keyboard, and joystick. The head-mounted display and data glove inputs and outputs are managed by specific libraries. Communication among the C++ VE and the Java agents is performed by using a middleware of CORBA objects.

Conclusions

An agent-based architecture is proposed in this chapter for the design of intelligent virtual environments for training. The roots of this architecture are in the generic architecture of an intelligent tutoring system that has been first extended to be applicable to IVETs, and has then been transformed into an agent-based architecture by the identification of the set of generic agents that would be necessary to accomplish the tasks of each module. Some details are provided about the most remarkable interactions that will be established among agents during the system's execution, namely the collaborative planning to find expert solutions to the situations and goals posed to the students, the determination of trajectories for the movements across the VE, and the supervision of the student's behavior. Some peculiar aspects are also discussed, such as the way to manage multiple views in a multi-user environment or the way to balance the individual versus the collective aspects in the system's functioning.

The proposed architecture, and its realization in a platform of generic and configurable agents, will facilitate the design and implementation of new IVETs, maximizing the reuse of existing components and the extensibility of the system to add new functionalities.

References

Alpdemir, M.N., & Zobel, R.N. (1998). A component-based animation framework for DEVS-based simulation environments. *Simulation: Past, Present and Future. Proceedings of the Twelfth European Simulation Multiconference* (ESM'98) (pp. 79-83), Manchester, UK.

de Antonio, A., Ferre, X., & Ramirez, J. (2003). Combining virtual reality with an easy to use and learn interface in a tool for planning and simulating interventions in radiologically controlled areas. *Proceedings of the 10th International Conference on Human-Computer Interaction* (HCI 2003), Crete, Greece.

Brown, J., Encarnaçao, J., & Schneiderman, B. (1999). Human-centered computing, online communities, and virtual environments. *IEEE Computer Graphics and Applications, 19*(6), 70-74.

Darken, R., Tonessen, C., Passarella, & Jones K. (1995). The bridge between developers and virtual environments: A robust virtual environment system architecture. *Proceedings of SPIE–The International Society for Optical Engineering, 2409,* 234-240.

Demyunck, K., Broeckhove, J., & Arickx, F. (1999). Real-time visualization of complex simulations using VE platform software. *Simulation in Industry'99. Proceedings of the 11th European Simulation Symposium* (ESS'99) (pp. 329-333).

Economou, D., Mitchell, W.L., Pettifer, S.R., Cook, J., & Marsh, J. (2001) User centred virtual actor technology. *Proceedings of Virtual Reality, Archaeology and Cultural Heritage* (VAST'2001), Athens, Greece, European Association for Computer Graphics.

Fencott, C. (1999). Towards a design methodology for virtual environments. *Proceedings of the Workshop on User Centered Design and Implementation of Virtual Environments,* University of York, UK.

Fikes, R., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence, 2.*

Hart, P.E., Nilsson, N.J., & Raphael, B. (1968). A formal basis for the heuristic determination of miniman costs paths. *IEEE Transactions Systems Science and Cybernetics, 4*(2).

Méndez, G., Rickel, J., & de Antonio, A. (2003). Steve meets Jack: The integration of an intelligent tutor and a virtual environment with planning capabilities. *Proceedings of the 4th International Workshop on Intelligent Virtual Agents* (IVA'03).

Munro, A., Surmon, D.S., Johnson, M.C., Pizzini, Q.A., & Walker, J.P. (1999). An open architecture for simulation-centered tutors. *Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration. Proceedings of AIED99: Ninth Conference on Artificial Intelligence in Education* (pp. 360-367), Le Mans, France.

Rickel, J., & Johnson, W.L. (1999). Animated agents for procedural training in virtual reality: Perception, cognition and motor control. *Applied Artificial Intelligence, 13,* 343-382.

Rickel, J., & Johnson, W.L. (2000). *Task oriented collaboration with embodied agents in virtual worlds.* MIT Press.

Sánchez, M.I. (2001). *Una aproximación metodológica al desarrollo de entornos virtuales.* PhD Dissertation, Universidad Politécnica de Madrid, Spain.

Section III

Collaborative Virtual Environments and Mixed Reality

Chapter IX

Construction of Collaborative Virtual Environments[1]

Anthony Steed
University College London, UK

Emmanuel Frécon
Swedish Institute of Computer Science, Sweden

Abstract

In this chapter, we give an overview of some of the issues that face programmers and designers when building collaborative virtual environments (CVEs). We do this by highlighting three aspects of CVE system software: the environment model (data structures, behaviour description) that the system provides, the data-sharing mechanism (how the environment model is shared), and the implementation framework (the structure of a typical client or platform in terms of the services it provides to the user). When a CVE system is designed, choices have to be made for each of these aspects, and this then constrains how the designers and programmers go about constructing the CVE worlds themselves. We present

the main body of the overview by using examples that highlight many of the important differences between CVE systems. We will also relate our discussion to the common topics of network topology and awareness management.

Introduction

A collaborative virtual environment (CVE) is a computer generated, three-dimensional space within which a geographically distributed set of users can interact in real-time. Many different types of CVEs can be found in use today, from online computer games to military simulations. The content and behaviour of different CVEs are very wide-ranging because of the different demands of the application. The primary requirement of an online computer game might be rapid response so that the game is fluid and enjoyable. The primary requirement of a military simulation might be verifiability and consistency so that the simulation can be studied for tactical purposes. These different applications thus require very different systems to support the CVE and they use quite different types of programming model and description languages.

In this chapter we look at the structure of *CVE systems* and how the structure constrains and informs the role of the designers and programmers of CVEs. A CVE system is a software suite upon which a number of different CVEs can be built. A particular CVE system, such as a military simulation built around the DIS system (see p. 251), is best suited to support one particular class of CVE. Thus although games can, and have, been built on DIS, it is not the most common CVE system for such applications.

Building a CVE system presents many interesting challenges. A CVE system needs to present audio, video, and potentially haptic data to participants. A CVE system needs to support interactive and reactive capabilities so that the CVE can respond to the participant and present an interesting world. And whilst doing both input and output in real-time, the CVE system also needs to distribute all the data to multiple users at different sites so that they can collaborate.

The types of data involved in describing a CVE and the requirement to be both real-time and distributed means that CVE systems are a field of study in their own right, and they are somewhat distinct from other types of distributed system. For example, CVE systems typically generate high volumes of small data packets that need to be delivered at very low latency with a mixture of reliability requirements. Faced with potentially overwhelming amounts of data to maintain, CVE systems focus on only managing data concerning the local surroundings of each participant. In practice this means reducing the complexity of data within the system, and taking advantage of the limitation of display systems and human perceptual capabilities so that only important, perceivable data are generated.

In this chapter we will look at how typical CVE systems are built. We emphasise commonalities and differences between different CVE systems, and how these impact upon the role of the designers and programmers of the actual CVEs. We will describe the differences between CVE systems by focusing on three aspects of the design of a CVE system: the *environment model* for describing the CVE, the *data-sharing mechanism* to support distribution of CVEs described in that model, and the *implementation framework* of the CVE system. We introduce each of these aspects in turn in the following section. We then give an introduction to some of the main real-world issues to consider when developing a CVE system. The following three sections then discuss environment model, data-sharing mechanism, and implementation framework in more detail. Each of these sections will present two contrasting systems that illustrate some of the main issues. Each section also presents the authors' view of some of the main research challenges in that area. We then devote a section to discussion of scalability, the current challenge for CVEs, and finish by discussing some requirements and prospects for the next generation of CVE systems.

Structure of a CVE System

In this section we elaborate on what we mean by the three aspects of a CVE system: environment model, data-sharing mechanism, and implementation framework. These three aspects can be summarised from a programmer/designer's point of view as: What data structures and programs do I need to write to build my CVE? How does data get shared between the users and what do I need to do to maintain the CVE? What application services are there within the system and how do I use them? Identifying these three aspects will allow us to contrast various CVE systems and isolate conventions that are used, but which may not always be explicit when the systems are described.

Environment Model

By environment model we mean the structures and properties by which the programmers and designers will describe the CVE. This involves a combination of *data elements* such as visual appearance and *behavioural elements* such as the social interactions of a simulated human or a mechanical part's operation. It does not include any of the mechanisms by which the model is shared. It also does not include any specifics of implementation framework of the system. However, as we will see, the distinction between environment model and implementation framework is often not explicit and varies from system to system.

Typically programmers and designers make a distinction between *media assets* such as geometry, texture maps, and audio samples; *properties*, which are non-renderable data that convey higher-level semantics such as weight or temperature; and *behaviours*, which describe changes to media assets and properties such as the change in location of a set of birds in a flocking simulation.

The environment model thus allows a programmer and/or designer to author a specific virtual space. As a simple example, the model will usually allow a set of media assets to be identified and located within a 3D space, and it will allow for the procedural programming code to describe behaviours that indicate how these media assets can vary over time, by, for example, moving and changing volume. An asset is thus usually a static item, generated by a modelling package (such as 3D Studio MAX), and the behaviours, usually written in a language like C++, Java, or Python give those media assets life depending on semantics given by the properties. Ideally the programming code associated with the behaviour of the environment will be largely independent of both the data-sharing mechanism and the implementation framework.

This distinction between asset, property, and behaviour is a gross one, but it is a common one that is made in CVE systems. Thus the construction of a CVE is usually perceived to contain three tasks that require different skill sets. This in turn means that constructing a CVE involves several iterations between modelling, programming, and labelling with semantics. Throughout this chapter we will refer to the people who develop the CVEs as programmers or designers to reflect the spectrum of skills and tasks required.

Data-Sharing Mechanism

Although several users can be supported off a single system, for example in a split screen situation, this does not fit with typical use of CVEs where the users are distributed and each user has significant local computing resources. This requires the environment to be distributed and maintained between different sites. As users alter the environment by interacting with it, those changes are communicated to the other users machines so that every one experiences a consistent environment.

There are two main classes of reason why this is difficult to achieve. The first class deals with communication speed. Network latency means that the different copies of the environment are never exactly the same, and indeed it takes work to stop them diverging. One of the big distinctions between CVE systems is how they deal with resolving ambiguity due to network latency, and how they then force the models to become consistent. The second class is that it simply may not be possible to enforce consistency of the environment with multiple parties. An example is constraint-based interaction between two users. If two users pick up a rigid virtual object at different points and pull in different directions, some part of the environment has to break because the users cannot be physically restrained. For example, either the object has to stretch, or one user has to be forced to drop the object. Even if it is possible to avoid a model violation by enforcing a rule from the behaviours of the world, users might be surprised by the results. A more difficult problem is that the model might not be computable since processing is limited. Collision detection is a good example since the computation required can be quadratic in the number of objects.

We will return to examples and issues in data sharing in the section on data sharing on page 250. At this stage it is worth pointing out that the main difference between systems is in how many of the behaviours of the system can be assumed when writing the data-sharing system. One obvious and common implementation is simply to explicitly share all the data structures that form the environment model. However this can be hard to make consistent since two users might operate on the same object, resulting in two machines attempting to apply the behavioural rules in parallel. In contrast, if you can assume that everyone knows the behavioural rules of the environment, you can use higher-level events and allow everyone to re-interpret a single high-level event as a series of changes to a model. A very simple example might be “Door 18 opens” being communicated and translated by all receivers into “DoorGeometry18 rotates 120^o about the Y axis and DoorSound2 plays for 2 seconds,” with the implicit follow up due to the game behaviours “Zombies178-185 are triggered and start lurching towards the player.” High-level events are used because they save network bandwidth and encapsulate a series of events that should not be separated. However they introduce inflexibility into the system.

Implementation Framework

The final aspect of the CVE system that we will explore is in the implementation framework of the CVE system. By implementation framework we mean the typical process or collection of processes that supports a particular user. Of course, the implementation framework will depend heavily on the model and data-sharing approaches, but even then implementation frameworks vary very widely, and the terminology to describe them varies enormously (toolkit, platform, component frameworks, etc.). The critical issue for us will be to isolate where the locus of implementation of the environment’s behaviour lies. When discussing the modelling of environments, we stressed the separation of behaviour in the model. Implementation frameworks tend to vary on where environment behaviour is described: it could be embedded in the world description and or it could be hard-coded in the system itself. Typically behaviour is actually split: some of the behaviours are fixed for most environments and others are described alongside the environment. The implementation framework can facilitate both types of description. An example of something that is relatively fixed is the mapping from tracking information to location of body parts of an avatar. As a gross characterisation, this would usually be specified by a configuration file that the implementation framework would read when it was started. The code for mapping tracking to position of body parts would likely be written in C or C++ because it would need to be executed in real time at a frequency of around 60Hz. An example of something that changes from world to world is behaviour of autonomous characters. Because it encompasses a lot more wide-ranging behaviour and is perhaps adaptive, this might be written in an interpreted scripting language.

These two examples hint at a core problem. Frequently executed code needs to be highly optimised because CVE systems are real time. However there is a very wide range of potential behaviours, and we might want to dynamically change the behaviour at run-time. Interpreted scripts are useful for dynamically changing behaviour, and with some care they are easy to compose with other scripts. Scripts can be changed whilst the system is running to allow rapid prototyping. Interpreted scripts may not be so efficient as native code, so any scripting relies on a set of services that express static computational facilities that can be called upon during execution.

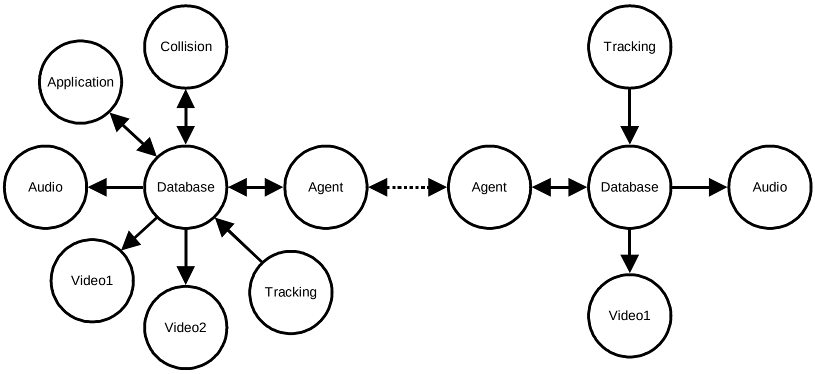
Of course this is, again, a gross characterisation. Application code, that is code that changes from environment to environment, is actually most commonly written in C or C++ and simply linked into the main system. This creates monolithic applications that, although efficient, are difficult to maintain. This is mainly because all of the clients of the CVE will probably need to have exactly the same revision of the code. In the experience of the authors, synchronising such clients is an onerous task. We note that scripting is one way to solve this, because the scripts can be shared just as other parts of the environment model.

If application code changes from environment to environment, what is the implementation framework responsible for? The implementation framework can be considered to be a set of services that do specific jobs. A service that is commonly separated out is visual rendering. This is because you would probably want to run this in parallel on a separate processor whenever possible because it is the main bottleneck with today’s implementation frameworks. However many other such services can be split off. Each service represents a logically independent set of functionality that has its own requirements on performance and update rate. The dVS architecture was an early CVE architecture that illustrates this (Grimsdale, 1991; Division Ltd., 1996). In dVS the system was made up of a set of *actors* (processes), each responsible for a specific set of functionality:

- *Visual actor* renders graphical views of the object database.
- *Audio actor* renders auditory views of the object database.
- *3D tracking actor* manages tracking devices and processes their raw data.
- *Collision actor* processes object movements and generates collide events.
- *Body actor* generates an object that represents the user within the environment.
- *Application actor* contains the application-specific behaviour of the environment.
- *Physics actor* simulates dynamic entities.

Each of these actors accesses a shared database of entities that represent the media assets and properties of the application. These include visual geometry, sounds, physics properties and collision properties. When setting up a multi-user session, some of these actors need to be instantiated more than once, and others are singletons. Figure 1 shows an example configuration of a multi-user CVE session. In the distributed setting, either with multiple machines supporting one user or multiple machines supporting multiple users, special *agent actors* handle communication. Agent actors only communicate necessary information for the actors on the machine that they handle communication for.

Figure 1. Example actor configuration for a multi-user CVE session



The architecture of dVS shows how an implementation framework can be constructed from a set of orthogonal services, with the environment-specific behaviour isolated from the majority of the services. For most CVEs built on dVS, all of the actors except the application actor would remain the same.

Unfortunately most implementation frameworks are not designed with this level of separation between the CVE system, which is general and stable, and the particular CVE, which is very specific and changing rapidly.

Real-World Issues

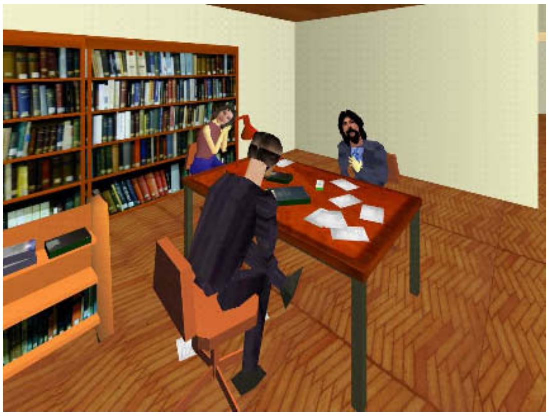
Before describing CVE systems in more detail, it is worth considering some of the real-world issues that affect the design and implementation choices. It has been observed that the design of a CVE is a trade-off between complexity and real-time performance. CVE users and CVE programmers and designers push the complexity of models, but often their expectations cannot be met and the CVE looks artificial or limited. To investigate this trade-off, and to better understand the nature of CVE systems, we consider three real-world issues. Firstly we consider user expectations, that is, the type and quality of the experience that users expect from a CVE. Secondly we turn these user expectations into modelling requirements: the detail and quality of models that have to be supported. Finally we discuss platform limitations: those aspects of modern computing platforms that most seriously impinge on the ability to generate very high-detail and very high-quality CVEs.

User Expectations

The typical user of a CVE will probably be using a non-immersive virtual environment (NIVE), perhaps a games console or a desktop PC. Rarer are users of immersive virtual environments (IVE) using head-mounted displays (HMD) or spatially immersive displays (SIDs, commonly referred to as a CAVEs™). However a lot of the research and development work in CVE systems is driven by the needs of IVE users since CVE systems that support IVEs are often very capable in terms of rendering and network bandwidth.

With both NIVEs and IVEs, the user's first expectation is that the displays (audio, video, etc.) represent a 3D environment within which they can interact. This requires the displays to represent 3D as best they can (through lighting, shading, stereo, occlusion, audio loudness, etc.), but moreover to be consistent in how they do this over time and to present multi-modal cues synchronously. Just as in 2D human-computer interaction, the user of a CVE is going to have to learn

Figure 2. Simulation of a library environment. Users have very strong reactions to the avatars despite the fact that the avatars have very simple appearance and behaviours. Some users will attribute intelligence and deep motives to the avatars.



some of the rules of the environment in order to interact with it. With an NIVE for example, the user has to learn how the controls (mouse, joypad, etc.) control the display. For example, does the joystick move the user through the world or the world around the user? There are then rules that have to be learnt about the environment itself. For example, what is the layout of the building and what objects are potentially dangerous?

Our second expectation concerns representation. We do not advocate photo-realistic or hyper-realistic representations that mimic the real world. Primarily this is because this is still impossible, but even if it were possible, it would not be desirable in all situations. The behaviour of objects is perhaps more complex to describe than visual appearance, and if the objects look real, the users will interpret the affordances of the object as they would in the real world (see Tromp, Steed, & Wilson, 2003, for a discussion of how this can impact CVE design). Of course this is a trade-off; the purpose of a CVE might be to convince the user that something is really reacting to them. In University College London's social phobia work (Freeman et al., 2003), avatars with very simple representations, but complex behaviours, will cause stress for users who have discomfort with speaking in public (see Figure 2).

Modelling Requirements

The visual quality of real-time graphics has improved vastly over the last decade. Three-dimensional environments are now almost ubiquitous in video games, and the detail within them means that most of the effort of building a game now goes on modelling, not programming. The detail in modelling is still a limitation, especially if the world takes on a realistic style. A case in point is the Sony Computer Entertainment Europe game, The Getaway, which involved more than 20 person-years of effort just in modelling the cityscape—110 kilometres of roads in 50 square kilometres of central London (Coates, 2001).

The detail of geometry and appearance are thus one of our biggest limitations when modelling. Scanning geometry from physical models and remote sensing help in some areas, but even these need a lot of input from a modeller. Similarly, realism of behaviour is also difficult. Behaviours are difficult to describe since they must appear consistent with the scene. As an example, consider modelling a human (commonly known as an avatar). The geometry and appearance may be scanned and static shots of virtual characters can easily fool a viewer. However, when such models move the illusion break down or if the user can interact with them, they will quickly see that the avatar has a limited range of expression.

So in both modelling geometry and programming behaviours, we have limits based on the detail required. However as we have alluded to in the previous section, users are quite forgiving of representation of objects and can operate with low detail models as long as they are consistent and understandable. We can also exploit the fact that users have limited perceptual capabilities, so we can reduce detail in the distance or for objects that are not visible.

Finally, we have not touched on modelling of properties of objects. This is mainly because the requirements of platforms vary greatly on their need for properties. Often properties are essential to label objects for the purposes of assigning behaviour to them or having behaviour refer to them. For example even a behaviour as simple as “make the avatar look towards the nearest window” needs some mechanism that labels which piece of geometry are windows. This would usually be done by adding a suitable property to window objects so that behaviours can search for them.

Platform Limitations

In our discussion of issues so far, we have steered away from thorny issues of the power of the machines implementing the CVEs. This is where most programmers and designers spend their time: optimising a particular environment for a particular piece of equipment, being it a Nintendo Gamecube or an SGI Onyx. However the boundaries here are always expanding: more polygons, more sound sources, and better behaviours. As machines get faster the users expect consistency and behaviours from the environments the machine supports, and the models get more complex.

There are some rules of thumb though. It has long been expected that the platform will generate a reasonable frame-rate. These days there is little reason not to aim for 50-60Hz or higher as consumer graphics cards have reached the order of 10^8 polygons per second. Thus today the platform limitations are in the size of the model that can usefully be held in memory and displayed, loading and maintaining these models from disk or from the network, and in the CVE situation, maintaining and animating many hundreds of moving entities. At the time of writing, network and disk bandwidth are probably larger limitations than rendering capability.

Choices for Environment Model

In making a CVE we need to decide what the content of the CVE is, that is, what we are going to have to specify in order to describe the environment. The visual appearance is the first of these. An associated sound might be the second. These are basic constituents of an environment model or scene description language. However when it comes to describing a particular scene, we may want to label or associate meta-data to particular objects to indicate other properties. A very common example is the solidity of an object. An object's visual geometry may be very complex with thousands of polygons. However this geometry may be no use for collision detection because it would be too complex to intersect it with other scene geometry in real time. Thus a "proxy" would be given that is an approximation to the shape object, but of much reduced complexity. This proxy is used for intersection tests on the basis that the user would not be able to tell the difference in the collision response. For example DIVE (see p. 255) allows the user to specify that any geometry is collidable, but it also allows any geometry to be invisible and thus geometry can be used for collision only. Other common practices that a model may allow are declaring types of objects so that they can be aggregated or iterated through. For example this is useful to support behaviours such as "sit in closest chair," where chair is not a first-class type of object in the system, but is a user-defined type that is useful for the particular application.

Hand-in-hand with the choice of the model is the choice of the modelling package. And fundamental to both of these choices is picking a style of model or a convention that will facilitate the particular application or class of applications that the programmer or designer wants to build. The choice boils down to picking a standard modelling language such as the Virtual Reality Modelling Language (VRML, see following section) that is generic and widely supported, or picking a highly customised format that is designed specifically for your application. Many modelling packages support a large subset of VRML at least as a target for exporting models. A highly customised format might facilitate faster modelling for that purpose, but is more difficult to apply to other contexts. Game engines are often of the latter form, and we will use Quake II as an example.

VRML97

The Virtual Reality Modelling Language (VRML, 1997) is a standard that specifies a file format for the description of 3D scenes. It includes geometry and appearance of objects and additional functionality for behaviours through animation, interaction, and simulation. We give a short overview of the structure of VRML as it illustrates several common features of environment models.

A VRML file contains a set of *nodes* that describe the scene. Each node is defined by several *fields*. For example, VRML has a *Cone* node that allows the programmer or designer to describe a cone with four fields, *bottomRadius*, *height*, *side*, and *bottom*. For each field, there is a type. In this case bottomRadius and height are *SFFloat* (floating point numbers), and side and bottom are *SFBool* (binary values). Together these fields indicate the size of the cone and whether or not the bottom is drawn and/or if the side is drawn. In the VRML file itself, you would see ASCII text such as the following:

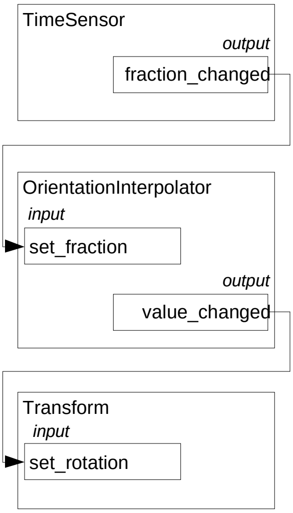
```
Cone {
    height 5.0
    bottom FALSE
}
```

Every field has a default value, and it is legal and preferred not to specify fields if their value is the default. In this case, the default bottomRadius is 1.0 and the default side is TRUE, so this node will appear in the scene as an open cone. There are such nodes for different shapes, and nodes that specify material and textures of objects. There are nodes for specifying groups and rigid transformations of objects, and nodes for other visual scene properties such as fog and a background. All CVE systems have mostly equivalent functionality for describing geometry and appearance. VRML97 also specifies sound properties which is less common in such scene description languages.

VRML97 also allows behaviours to be specified. It does this through a data flow scheme. A graph structure that links fields in different nodes together is defined using ROUTE statement. Data flowing through the graph specifies new values for fields, and these new values are calculated in every frame to animate the

Figure 3. Animating a rotating box using VRML animation nodes

```
DEF TRANS Transform {
  children [USE ONE_BOX]
}
DEF TIMER TimeSensor {
  loop TRUE
  cycleInterval 2.0
}
DEF ROTATOR OrientationInterpolator {
  key [ 0, 0.5, 1 ]
  keyValue [ 0 1 0 1, 0 1 0 3.141, 0 1 0 6.281]
}
ROUTE TIME.fraction_changed TO
  ROTATOR.set_fraction
ROUTE ROTATOR.value_changed TO
  TRANS.rotation
```



scene. There are nodes that specify time generators, and there are nodes that take time values and generate animations of other types of values. The following example should illustrate how this allows behaviours to be described.

In Figure 3, we see an example of a very simple behaviour that specifies a rotation. The node *Transform* wraps a piece of geometry, in this case a box. All transform nodes have a field called *set_rotation*, to which new values can be passed in order to set the rotation component of the transform. The *OrientationInterpolator* node takes a time value between 0 and 1, and generates an orientation. It does this by using the fields *key* and *keyValue*, which indicate how time (*key*) maps to orientation (*keyValue*). If the time is between the keys, then the orientation is calculated as an interpolation between the two closest keyValues. For example if the time is 0.25, the orientation is halfway between (0 1 0 0) and (0 1 0 3.141) which is (0 1 0 1.5705). Orientations are specified in axis-angle format. That is, the first three values specify the axis and the fourth the angle rotation about that axis. The effect of this OrientationInterpolator is thus to rotate once when the time varies between 0 and 1. The OrientationInterpolator is driven by a *TimeSensor* node that generates time values in the range 0-2. Every frame the TimeSensor takes the current time and finds that value module 2. The TimeSensor generates a value in the range 0-2 in each frame. The OrientationInterpolator does not know about time values above 1, so it simply uses the closest value. The complete effect then is a box that rotates around in one second, waits one second, then repeats. If more complex effects are required, a data-processing script can be written in Java or JavaScript and placed into the data-flow graph.

VRML97 itself is a description of a single scene; it is not designed to support CVEs. However there have been many extensions of VRML97 that support networked environments. Living Worlds was an effort to standardise multi-user extensions to VRML97 (LivingWorlds, 1998). LivingSpace (Wray & Hawkes, 1998) was an implementation of this based on three layers. The lowest level is a generic notification system. Above this notification system, generic support for state sharing is provided by an event interface. Finally, the top layer consists of the support for zones, a region of the space according to the Living Worlds proposal. We will explore some of these concepts in later sections.

Quake

The Quake series of games from Id Software is notable because the series changed the public perception of what was possible with 3D and online games. It is also notable because it was easy to modify the games and a large community of "mod" developers grew up around the series.

We will use Quake II as an exemplar of a typical game engine because the complete source code for the game has been released under the GNU Public Licence and the interested reader can compare it to the other systems in this chapter. Game engines have been widely discussed in the CVE community because they are very successful at generating relatively complex worlds (Capps, McDowell, & Zyda, 2001). Under the hood there is not much to distinguish game engines from other CVEs. Typically game engines are more constrained in their behaviour, but more effort is expended on generating models (c.f. the example of SCEE's The Getaway earlier). Indeed because the game data is often a very large component of the effort, the game engine and the production pipeline for the environment are built in parallel. Thus game engines often dictate specific tools or their own tools for model building. The models are thus highly customised for the engine that will run them, and a particular combination of production pipeline and game engine might be extremely inflexible and not lend themselves well to other types of game.

The full Quake II environment models would be very long, but we will note some features that make a contrast with VRML97. Firstly there is a clear distinction between behaviours and media assets. The game behaviours are described in a dynamically loadable library compiled from C code. The media assets are very highly customised, and indeed they are generated by very specific tools. Id Software wrote their own production tools when producing the game. After the game was released, they made these tools publicly available. These tools seeded the mod community, which has since extended these tools. The bulk of the geometry in a Quake II world comprises architectural models. These are described in a format that revolves around a binary-space partition (BSP) tree (Fuchs, Kedem, & Naylor, 1980). The leaves of the BSP tree contain pointers to clusters of faces that define the actual surfaces of the world. The BSP tree does two main jobs beyond being a geometry representation: it provides a basis for pre-computed visibility sets (Airey, Rohlf, & Brooks, 1990; Teller & Sequin, 1991), and it provides an efficient data structure for collision detection. Generation of BSP trees is time consuming and error prone, and this is one of the reasons why the game requires its own tools. The tools for modelling the worlds are very limited compared to other, more general modelling tools, and they impose a number of constraints so that BSP trees can be generated successfully. An example constraint is that the vertices of the model can only lie on integer values in a range of –8000 to +8000 in each direction. Some properties are written into the BSP file to enable game behaviours. Some objects will be labelled as starting points, lifts, and so on.

The second type of asset involves the models of the players' characters. These are described in a separate data structure. A character is made up of a sequence of 3D meshes. Each mesh represents a pose, and typically there are fewer than 200 poses. When a character is animated, its representation is limited to being an interpolation between two such poses.

Overall then, Quake II, although very successful as a game, is difficult to re-purpose as a general system because it has highly customised data structures.

Research Issues

Environment models and the scene description languages that go with them are still a matter for research. VRML's role will be superseded in the near future by X3D (X3D Working Group, 2003), but X3D is not a fundamental change from VRML97 in the way scenes are modelled. Several extensions have been proposed that facilitate a particular class of modelling problem. For example, GeoVRML is an extension to VRML97 that makes it easier to model very large terrains (Reddy & Iverson, 2002). Game engines continue to evolve to match user expectations and the increasing power of platforms, but a big challenge for games developers is trying to support a range of platforms, especially now that mobile games are starting to generate more significant sales. The research issues today are thus in generating canonical environment models that can be compiled or simplified to run-times that satisfy a variety of different computing platforms.

Another current challenge is supporting physical dynamics simulation as a service in the implementation framework. This is done in limited demonstrations today, but the work of companies such as Havok has shown how a variety of dynamics models can be integrated into environments. There is also useful work to be done on the related problem of taking visual models for geometry and deriving useful collision surfaces. Indeed, currently there may be two completely different representations of the same world, including a polygonal geometric model for the renderer and a BSP model solely for the collision detection (Shaw, 2003).

Many scene description languages have limited facilities for describing novel types of content, or properties, beyond that which they were originally designed for. It is difficult to add novel data to Quake II models because the environment model was highly customised to support the types of assets the game required when it shipped. VRML does allow novel data to be described using a mechanism know as prototyping. However even then novel data is difficult to support throughout a production pipeline because modelling tools obviously cannot edit content for which they do not know the semantics. Thus a modelling tool either needs to be aware of the types of novel content even if it can't support the content or it must try to preserve content that it detects is novel. Unfortunately the latter is hard to do and is not commonly done in modellers, so we are back in the realm of customised production pipelines for specific models. This is one reason why modelling tends to be a pipeline process and not an integrated process.

Finally we note that in certain fields there are more specialised environment models that are either native to one package but widely supported by other packages for the purposes of inter-operability, or are industry-defined standards. An example package-native environment model is CATIA, which is customised for product design and thus focuses on descriptions of shapes such as curves, surfaces, and solids, rather than polygons (Dassault Systemes, 2003). An example industry standard is IFC2x Edition 2, which is customised for describing architectural building components (IAI-International, 2003).

Choices for Data-Sharing Mechanism

Earlier in this chapter we made the distinction between data-sharing mechanisms that assume there are shared structures and data-sharing mechanisms that assume that high-level events can be used to synchronise clients. Now, we can better characterise the distinction because we have covered environment models and how particular applications can be described in those models. If an application is well defined and more closed in nature, it will tend to use events that describe a high-level state of the environment. Earlier we used the example of a game that used a high-level event of a door opening that triggered several consequences. High-level events are useful because maintenance of state between several machines is easier because a smaller number of events are required. Indeed if the events are standardised, then several different CVE systems with completely different implementation frameworks might interact. We will describe DIS, which is an example of such a mechanism.

In contrast, if a CVE system is more open in nature, it may not be possible to define a set of such high-level events, and a more general data-sharing mechanism will be required. The simplest type of general shared data is distributed shared memory. In this approach the programmer simply declares data structures for their model in shared memory, and this is transparently synchronised between hosts. This approach is used by the DIVERSE system (Kelso et al., 2003). However, CVE systems usually have a well-defined environment model, and it is more common for the model to be shared between hosts without directly copying the actual in-memory data structures. As an example of this type of system, we will describe DIV, Distributed Inventor, which is based on Open Inventor.

DIS

For over two decades, the defence community, led by the U.S. Department of Defence (DoD), have invested in a number of research programmes on distributed simulation. Many of defence simulations rely on the DIS (Distributed Interactive Simulation) standard (IEEE, 1993) that emerged as a result of these efforts. The core of DIS is a suite of network protocols that allows interconnection between heterogeneous collections of simulators. Unlike many systems, DIS places no constraints on the software architecture of the simulator. DIS-compliant simulators are very diverse, ranging from immersive virtual reality systems through to intelligent agent systems.

A simulation is comprised of a number of objects. Each simulator node introduces objects into the environment and is subsequently responsible for those objects. Other simulator nodes maintain local copies of those objects. Each simulator node broadcasts regular events for each object it is responsible for. These events must completely describe any changes in state. Simulators receiving these events must themselves decide how to apply the state change to their local copy. Because there is no constraint of the representation of an object on a simulator, it is very important that the events that specify change in state follow the standard packet format provided by the PDU (Protocal Data Unit).

All PDUs incorporate an identifier for the entity concerned, the responsible for the entity, an application identifier, the type of the entity and the type of the PDU. The remainder of the PDU depends on the type. For example, PDUs describing vehicles will contain information such as velocity and orientation, whereas PDUs concerning humans will contain information such as stance or gait. There are a large number of different PDU types because they have to encompass all state changes for the variety of different entities that might be encountered in a military simulation.

If a client can interpret PDUs, they can participate in a DIS simulation. However, for general simulations DIS has some drawbacks. Network messages are sent using UDP. Consequently, they can get lost or be received in the wrong order. This is not necessarily a problem for entities such as airborne vehicles that move continuously. In this case the expectation is that a missing or mis-ordered event will soon be superseded by a more recent event. However, if an object changes state only infrequently, there is a possibly of an inconsistency. Inconsistencies might also arise because there is no centralised time-management thus concurrent events with side effects on overlapping entities might be resolved differently at different simulators. The event-based natured of DIS also means that it is cumbersome to introduce large amounts of state, such as weather conditions, in to a simulation.

A further set of problems revolves around the fact that simulators just broadcast events. This means that every other simulator receives these events and must handle them. With only moderate numbers of objects and a fixed update rate, this would quickly lead to congestion. To combat this DIS relies heavily on a technique called dead reckoning. Many objects have state that can be extrapolated into the future. For vehicles this might include velocity and acceleration. Every simulator can run perform this extrapolation. The simulator responsible for an object keeps a copy of the object state when it last sent an event, and extrapolates this to estimate the state that the other simulators maintain. When the actual state diverges significantly from the extrapolated state, the responsible simulator sends another event. This significantly reduces the number of events that need to be sent.

As a distributed system, DIS concentrates on the sending of minimal state changes to maintain a common state among heterogeneous simulators. DIS itself is low level, that is, it is close to the network. It has no notion of a shared database or event ordering, so there is no guarantee of consistency between simulators. Building a CVE around DIS requires significant software infrastructure. However the main principle of DIS, determining a small number of events that efficiently describe change in state, is a key principle that underlies most CVE toolkits.

DIV

Distributed inventor (DIV) will be our example of the shared-scene graph approach (Hesina, Schmalstieg, Fuhrmann, & Purgathofer, 1999). Scene graphs are a common abstraction for graphics. Open Inventor (Strauss & Carey, 1992) is one such abstraction and is particularly suited to describing interactive 3D scenes. It is strongly related to VRML, in that VRML1.0, the precursor to VRML97, was derived from the file format for serialising scene graphs built in Open Inventor. Open Inventor supports loading of scene graphs from a file, construction of scene graphs within code, or a combination of both. Like most scene graphs, nodes in the graphs represented geometric primitives, appearances, transforms, and interactive elements.

What DIV does is to share this graph between multiple clients. When a node in one instance of a Open Inventor graph is changed, it generates a local callback with an event that encapsulates the change made, be it a structural change in the scene graph or a local change in the properties of the node such as a change in a colour or a vertex location. These events are then distributed amongst all sites using message passing. Table 1 shows the events that are used to maintain a scene graph. These are typical of all shared-scene graph toolkits as they reflect the general types of edits that can be done on graphs.

One limitation of this approach is that packaging up certain behaviours can take a lot of callbacks. Imagine a flock animation where the position of every bird has to change each frame and the shape of the wings has to change on most frames. Although the behaviour is quite simple, many dozens of small changes need to be communicated across the network every frame. This is exactly the type of situation in which a higher-level event might be more useful. For example, it might not matter for the application that the position of the wings of each bird be synchronised between each site. If it didn't matter, then the wings could be

Table 1. Events used to share DIV scene graphs (adapted from Hesina et al., 1999)

Message	Parameters
Update field	node ID, field ID, value
Create node	node type, parent node name, child index
Delete node	node name
Create sub-graph	file name or URL, parent node name, child index
Set node name	path to node, new node name

animated separately at each user's machine, independent of other users. The whole flock could be controlled by a single message per frame which controlled all of the positions of the birds.

DIV is a very flexible system. Many different CVEs can be built on it because it uses a very general data structure. The trade-off is that many events can be generated because the system cannot aggregate a set of small changes into a higher-level event.

Research Issues

In this section we have discussed data-sharing mechanisms for CVEs systems. It is worth stressing that we have covered just a couple of mechanisms that characterise dozens of systems. Data sharing is the core topic of networked virtual environments; more examples can be found in Singhal and Zyda (1999). One issue we have not addressed is how consistency between multiple sites is maintained when each site has the ability to apply the behaviours inside the environment. A variety of methods exist, many involving nominating or negotiating ownership of objects so that at most one user has control over a particular object at any one time.

The speed of updates and volume of data distinguish CVEs from other types of distributed application. The volume of data issue can be tackled by exploiting the user expectations as discussed in the section on real-world issues on page 241 and moving to a partially shared model approach. This approach is very simple: It observes that we only really need to replicate the data or receive events for objects that are close to us or applicable to our current task.

Finally another shortcoming for most types of current data-sharing mechanisms is that they can be difficult to extend to cope with types of data. In DIS, if a new type of entity is required, it either has to be shoehorned into an existing PDU, or a new PDU has to be defined. In shared scene graph approaches, new data types have to fit into the pre-defined data structures. EQUIP is a recent system that attempts to overcome this by allowing dynamic type extension of the running system through dynamic loading of implementations (Greenhalgh, 2002). EQUIP embeds a lot of the experience of the MASSIVE series of systems that are well known for their approach to scalability (see section on awareness management, page 262).

Choices for Implementation Framework

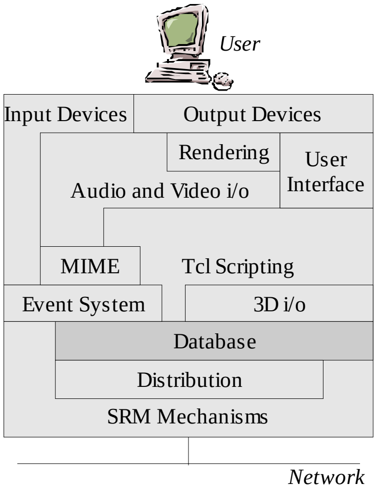
The implementation framework is where there is the most variety amongst CVE systems. The environment model and data-sharing mechanism place constraints on the implementation framework, but even then there are a lot of options. A critical option is the programming languages supported. As we mentioned in the section on environment models, an implementation framework might support an interpreted scripting language for rapid prototyping and behaviour description. Interpreted scripts can be written and stored with or in the scene description. This is in contrast to an implementation framework where applications are written as standalone executables. We will describe DIVE and CAVElib to exemplify the difference here.

DIVE

DIVE is a long-established system for CVE research prototyping (Carlsson & Hagsand, 1993; Frécon & Stenius, 1998; Frécon, Smith, Steed, Stenius, & Stahl, 2001). DIVE defines an environment model that uses a scene graph. Its data-sharing mechanism uses replication of that scene graph.

An overview of the structure of the system is given in Figure 4. At the conceptual and programming level, DIVE is based on a hierarchical database of objects, termed *entities*. Applications operate solely on the scene-graph abstraction and

Figure 4. The different modules that compose the DIVE system, together with their interfaces



do not communicate directly with one another. This technique allows a clean separation between application and network interfaces. Thus, programming will not differ when writing single-user applications or multi-user applications running over the Internet. This model has proven to be successful; Dive has changed its inter-process communication package three times since the first version in 1991, and existing applications did not require any redesign.

While the hierarchical database model is inherited from traditional scene graphs, as used in the computer graphics community, the Dive database is semantically richer. For example, it contains structures for storing information about other users, or non-geometric data specific to a particular application. In Dive, the database is partially replicated at all participating nodes using a top-down approach, i.e., mechanisms are offered to control the replication of sub-branches of a given entity. It is worth noting that the conceptual model is very similar to that of the Spline system (Waters et al., 1997), even though both systems have emerged from distinct efforts and have developed separately.

In Dive, an event system realizes the operations and modifications that occur within the database. Consequently, all operations on entities such as material modifications or transformations will generate events to which applications can react. Additionally, there are spontaneous and user-driven events such as collision between objects or user interaction with input devices. An interesting feature of the event system is its support of high-level application-specific events, enabling applications to define their content and utilization. This enables several processes composing of the same application (or a set of applications) to exchange any kind of information using their own protocol.

Most events occurring within the system will generate network updates that completely describe them. Other connected peers that hold a replica of the concerned entities will be able to apply the described modification unambiguously. Network messages are propagated using the multicast mechanisms that are built in the system. Dive uses a variation of SRM (scalable reliable multicast (Floyd, 1997)) to control the transmission of updates and ensure the consistency of the database at all connected peers. The SRM approach requires the transport layer to be able to ask the application (in this case Dive as a whole) to regenerate updates if necessary. Update regeneration is necessary when gaps are discovered in the sequence numbers that are associated with every entity in the database. Gaps imply that network messages must have been lost along the path from a sender to one of its receivers. In addition it is possible to access any document using more common network protocols (HTTP and FTP), and to integrate these documents within the environment by recognizing their media types (such as VRML, HTML, etc.).

In any application, the content of the database must be initialized. Dive uses a module that manages several three-dimensional formats and translates them into the internal data structures that best represent their content. Usually only one peer will load and parse a particular file, and the resulting entity hierarchy will be distributed to other connected peers through a series of (multicast) updates that describe the resulting entities.

DIVE has an embedded scripting language, Tcl, which provides an interface to most of the services of the system. Scripts register an interest in, and are triggered by, events that occur within the system. They will usually react by modifying the state of the shared database. Moreover, these modifications can lead to other events, which will possibly trigger additional scripts. A series of commands allow the logic of the scripts to gather information from the database and decide on the correct sequence of actions.

The primary display module is the graphical renderer. Traditionally, the rendering module traverses the database hierarchy and draws the scene from the viewpoint of the user. DIVE also has integrated audio and video facilities. Audio and video streams between participants are distributed using *unreliable* multicast communication. Audio streams are spatialised so as to build a *soundscape*, where the perceived output of an audio source is a function of the distance to the source, the inter-aural distance, and the direction of the source. The audio module supports mono-, stereo-, or quadri-phony audio rendering through speakers or headphones connected to the workstation. Input can be taken from microphones or from audio sample files referenced by a URL. Similarly, the video module takes its input from cameras connected to the workstations or video files referenced by URLs. Video streams can either be presented to remote users in separate windows or onto textures within the rendered environment.

The services described previously are independent of any DIVE application. Many different DIVE applications exist that use these services directly. The DIVE run-time environment consists of a set of communicating processes, running on nodes distributed within both local and wide-area networks. The processes, representing either human users or autonomous applications, have access to a number of databases, which they update concurrently. As described earlier, each database contains a number of abstract descriptions of graphical objects that, together, constitute a virtual world. A typical DIVE application will, upon connection to a virtual world, introduce a set of objects to the environment that will serve as its user-interface, and start listening to events and react accordingly. One essential application of the system is the 3D browser, *Vishnu*, which is a standard application that gives its user a presence within the environment. It introduces a new entity called an *actor* to the shared environment, which is the virtual representation of the real user. Vishnu renders a visual and aural space, and provides users with an interface that allows them to explore and interact with this space. Vishnu is a default high-level user client.

Users may also be presented with a two-dimensional interface that offers access to rendering, collaboration, and editing facilities. The interface itself is written using the same scripting language as offered by the world database. Consequently, CVE applications can dynamically query and modify the appearance of the 2D interface. For example, the London Traveler application (Steed, Frécon, Avatare Nöu, Pemberton, & Smith, 1999) exploits this feature by adding an application-specific menu to the regular interface of the DIVE browser.

Finally, a MIME (Multimedia Internet Mail Extensions) module is provided to better integrate with external resources. It automatically interprets external URLs. For example, an audio stream will be forwarded onto the audio module where it will be mixed into the final soundscape.

DIVE thus specifies a range of different services at different levels. Programmers can use the core libraries that provide core services and extended modules that provide user services. However, what tends to distinguish DIVE from other systems is that there is a default application, Vishnu, which integrates many of these services. Vishnu can load a world description file that, because it allows embedded Tcl scripts, is powerful enough to describe a very wide range of environments. Vishnu, the application itself, changes rarely, so there is a high degree of interoperability between instantiations at different users' sites.

CAVElib™

The CAVE™ library (CAVElib™) is designed to support spatially immersive display systems such as the CAVE™. As an implementation framework it mostly deals with display configuration and input devices, though it provides a well-defined structure for applications and some networking support. It does not impose an environment model, though it does provide an interface to OpenGL Performer, which can be used to store a scene-graph for rendering and interaction. The programmers thus define their own data structures. The data-sharing mechanism is very low level; we will discuss this later.

The core of the CAVElib is a set of services that isolate users from having to know the precise devices and displays connected to the system. For example, the user does not deal with actual tracking input devices, but abstract devices labelled by a logical name such as head. This and other abstractions, such as button labelling, are configured in an external file. CAVElib also isolates programmers from the display configuration. The programmers need to make a choice of OpenGL or SGI Performer-based graphics API. With the OpenGL library, the programmer needs to synchronise all data across the multiple rendering threads. With SGI Performer, Performer itself takes care of the multiple rendering processes. In both situations the programmer's application is a separate process that runs in lock step with the renderers.

CAVElib includes some basic CVE networking support. If configured to do so, an instance of a CAVElib application will multicast its interaction device data on the local network at a periodic rate. CAVElib automatically listens for such messages from other applications and builds a data structure that represents the other users. From this, the application can build a CVE by drawing avatars, enabling shared interaction, and so on. This is enough for very basic CVEs, but for more complex behaviour, it is necessary to be able to communicate more application data to the other clients. CAVElib provides a mechanism to do this: Any client can send data to the network, which will generate a callback at the other sites. This callback will be presented with a binary data stream and will be expected to unpack this and interpret this. This leaves the CVE programmer with a lot of work to do: deciding the data formats, standardising this amongst application, and then maintaining all the clients to the same code revision.

CAVElib networking thus provides low-level aspects of both shared memory and event-based data sharing. To the programmer, the array of the position and inputs of the other user can be considered to be a shared data structure. The data callbacks have to be considered to be event based, because the events themselves are transitory and do not explicitly represent shared data. This networking approach assumes that all the clients can interpret the data received from the network. This implies trusting the senders, and assuming that the data can reliably be matched and can be unpacked into a known type. Any type of system can be built on top of this mechanism, which is very close to low-level networking libraries.

Although CAVElib is quite powerful, it must described as a thin library in comparison to DIVE, in that it provides very little support for user services. CAVElib has flexibility, but at the cost of application programmers having to do a lot of work. The only way to share code between CVEs is to share C++ classes. Thus environments behaviour is embedded into the application and is immutable at run-time.

Research Issues

The developers of DIVE have made strenuous efforts to standardise a single application binary, and move semantics and behaviours into the world description. This enables a number of interesting run-time properties: A DIVE user can join an online session without having the same version of the client, and having none of the environment description. The former is enabled because of the abstraction of using a distributed scene graph. The latter is enabled by making all of the environment description available within the database through the use of interpreted scripts and meta-data properties. DIVE's environment model is thus quite sophisticated in comparison to many other systems.

Although we have contrasted Dive and CAVElib, they are not really in competition. With Dive the focus has been on the high-level user services, whereas with CAVElib the focus has been on broad support for device support, rendering, operating system abstraction, and display re-configuration. Indeed versions of Dive that support immersive displays have been built on top of CAVElib, though they haven't used the network services (Steed, Frécon, & Mortensen, 2001).

This is the trend in CVE research at the moment: building higher-level user services to set up collaborative sessions, improving interpersonal communication, and interfacing to other awareness and group collaboration tools. It is unlikely that an abstract architecture for CVE services will emerge soon, unless a single implementation framework proves itself to be much more capable than all the others.

Scalability

In our discussion so far, we have ignored the two questions that occupy most of the research literature in the field: network topology and awareness management. We have avoided these questions because their resolution shouldn't really impinge on the user, programmer, or designer experience. In an ideal world, the programmer and developer would be concerned with the implementation framework, data-sharing mechanism, and environment model, and be isolated from issues of scalability. However, this is far from being the case. Because of platform limitations, the programmer or designer will often have to address scalability issues head on, and sometimes the user will have to reduce their expectations because of these limitations.

Network topology is important because it trades off latency and bandwidth for simple mechanisms for consistency. Awareness management is important because in any reasonably sized CVE, it is impossible for a single user's machine to maintain the complete, up-to-date state of the CVE.

Network Topology

Client-server systems make one process, the server, responsible for maintaining the environment model. In this type of system, clients send object updates to the server, and the server relays these updates to the other clients. This is a common approach that is used in the three systems we mentioned already: LivingWorlds, Quake II, and EQUIP. The advantage of this type of system is that it gracefully solves the problems of consistency by letting the server decide upon the sequence of actions. Another advantage of a server solution is the possible gain in bandwidth at the client side. The server is able to take a number of decisions upon which object updates should be communicated, at which pace, within which vicinity. All these decisions can be made in concert with the clients and their known available bandwidth access. Consequently, client-server solutions are often used for community-oriented systems, which target consumer computers with modem connections.

The client-server approach has a number of drawbacks. The main one is the introduction of additional communication delays. Indeed, before any decision has to be taken at the client side, the client has to ensure that it will be allowed to perform the action. Furthermore, the server is responsible for dispatching object updates to all interested participants. Therefore, network packets will travel twice: once from the source client to the server, and a second time from the server to the destination clients. On a congested Internet, this travel time can be measured in hundreds of milliseconds, if not in seconds. Server architectures also face the problem of scale. As the number of clients grows, the server's processing and network capability will become a bottleneck. A solution, as employed in a number of systems, is the multiplication of servers in various ways (for example, by virtual geographical position, by actual geographical position, etc.). This solution has a financial cost that might not be sustainable within all contexts.

Finally, through the introduction of a central point, a server-centric solution introduces possible long-lived failures. As soon as one or several servers stop working, either for hardware or software reasons, part of the virtual environment will also stop working and stop living.

In the **peer-to-peer** model, all participants' processes will communicate directly with a restricted and well-chosen set of other participants. Examples of systems using this model are MASSIVE-1 (Greenhalgh & Benford, 1995) and DIVE.

As opposed to the client-server model above, this model has the advantage of reducing network delays by removal of the need to relay updates via a server. Since real-time interaction is one of the key facilities of CVEs, peer-to-peer systems are generally preferred for systems tuned for highly interactive environments. This solution has the advantage of not putting the burden of scale on any specific central point within the network. Instead, used in conjunction with partitioning techniques, each client will only have to communicate with a restricted set of its peers. As consumer hardware is gaining in both communication and processing power, peer-to-peer systems are gaining importance.

However, there are a number of drawbacks to the peer-to-peer approach. For example, maintaining consistency of behaviour becomes a more complex problem—it involves arbitration between peers.

Additionally, filtering facilities such as those offered by a central server are harder to achieve. One solution would be for each pair of clients to actually negotiate how this communication should occur, but such a solution requires some additional processing power at the sending client, which is not always compatible with the number of other tasks that must be performed in real time. Finally, a pure peer-to-peer approach is the one that actually puts the greatest burden on the network since packets have to be duplicated as many times as there are destination peers. To relieve this situation, multicast has been proposed and is in use in a large number of systems (see p. 263).

Awareness Management

Awareness management schemes exploit human perceptual and cognitive limitations by only transporting the data that is likely to be relevant to the user. Generally they focus on either ignoring data that is out of sight or out of earshot, or on reducing the fidelity of data that is far away. For example, audio packets from distant participants can be discarded since audio spatialisation will render them inaudible or position updates from an entity behind a door can be discarded under the condition that door is closed and blocks the view. As participants move around the world, their interest, and thus the relevance of different objects will change. Thus awareness management schemes need to be flexible and dynamic. The best exemplars of awareness management schemes are NPSNET, SPLINE and MASSIVE-2.

- NPSNET (Macedonia, Zyda, Pratt, Barham, & Zeswitz, 1994) divides the environment into a regular array of hexagonal cells. Each participant sends position updates to the single cell they are contained within and receives updates from all cells within a fixed radius. This scheme works well if the participants in the simulation are relatively uniformly distributed, such as in the battle simulations for which NPSNET was designed. However, if the participants are clustered and are all mutually aware of each other, they might still be over-whelmed with data.
- SPLINE (Sterns & Yerazunis, 1997) in order to avoid the problem of regularly-sized cells, SPLINE divides the environment into *locales* of variable size and shape. Each locale contains portals that express its adjacency to other cells. Each participant sends position updates to the locale they are currently in and receives information from their current locale and its neighbours. The locale mechanism allows very flexible worlds to be described, but the configuration is still static, so congregations of participants in a single locale may still cause overload.

- MASSIVE-2 (Benford & Greenhalgh, 1997) extends the locale mechanism by dividing the environment into regions whose boundaries can provide different degrees of permeability for different media. For example, a wall between two regions may block all visual information but only attenuate audio information. Furthermore regions can also provide aggregate representations of their contents and regions can also be mobile. One example that Benford & Greenhalgh use is of a region that surrounds a crowd of participants and moves with them. From a distance, the crowd-region presents a simplified representation of the participants with it, with less frequent position updates and pre-spatialised audio.

All of these systems use distance or occlusion to tackle the issues of scale and to reduce the effects of movement and interaction. They work by partitioning the world into distinct regions and allocating separate sets of system and network resources to each region. Consequently, only a reduced number of participants will share each set of resources.

Other Approaches

In virtual environments, packets sent by participants have to reach a number of destinations. These destinations will be those participants that the system decides are interested in the packets. Partitioning techniques of all sorts are used to avoid simply broadcasting all events. There is, however, an alternative choice, which is the one of multicast (Macedonia, Zyda, Pratt, Brutzman, & Barham, 1995). Multicast is a networking facility that allows an IP address to indicate a group of receivers. Each packet sent to a multicast IP address will be received by all members of the group. The sender sends exactly one packet and the network itself deals with duplication of the packets as required so that each member of the group receives the packet as if it had been sent directly. Duplication will usually happen within the routers themselves, allowing for hardware acceleration and a faster delivery of the packet. This is in contrast to the peer-to-peer approach where each client sends each packet multiple times, each with a different address.

However, multicast has a number of drawbacks. Until the début of IPv6, the number of available multicast groups was restricted. Therefore, schemes where each active object would be associated with a separate multicast group and where remote participants would join these groups as needed have been impractical. Such schemes are also impaired by the fact that joining and leaving operations require of network and computing resources both at the client side and within the routers in the Internet. This problem applies to all multicast solutions.

Furthermore, the spreading of multicast on the Internet has been slow: Network operators are reluctant to offer multicast to their customers, computer hardware only supports a handful of multicast groups in network cards, and operating systems have been slow to incorporate multicast capabilities. Finally, multicast packet delivery is based on UDP and is, thus, unreliable.

Multicast has a number of advantages over standard unicast communications. For example, in unicast communications based on the client-server model, packets have to travel all the way from the network hardware, through the operating system up to the application server before a decision can be made whether they should be forwarded to another participant or not. For the forwarding to happen, packets have to travel all the way back from the application, through the operating system, down to the network hardware. These travel times under stressed situations can account for a large part of the delays introduced. These travel times are also of importance in pure peer-to-peer unicast approaches where packets are already duplicated at the clients. This is especially true since such clients have to perform a number of other computing-intensive operations such as the rendering of the graphical 3D scene or the mixing of audio packets coming from the remote participants. On the other hand, uninteresting multicast packets can already be discarded at the hardware level or at the low-level software level.

To alleviate the slow spreading of multicast and its difficulty to reach consumers, a number of systems rely on mixed architectures. An example is the Spline system (Sterns & Yerazunis, 1997). In Spline, servers are placed on a trusted network to glue together true clients and other multicast-capable peers. Packets coming from the clients will be multiplexed at the application level to all necessary clients of the servers and also sent to the multicast groups. Symmetrically, multicast packets incoming at the server will be forwarded as necessary to the clients. Additional computing is performed at the servers in order to minimise the bandwidth used.

Future CVEs

User expectations of CVEs have expanded rapidly in the last few years. In our opinion the main challenges are now in modelling detailed environments and scaling up to large numbers of users without sacrificing flexibility. The success of multi-player online games is showing that it is possible to make persistent online environments, but currently these environments are limited in their scope. This scope is being pushed by current games such Second Life (http://secondlife.com/) and There (http://www.there.com/), which offer distributed environments amongst moderate numbers of players with a wide range of user customisation and user authoring.

In the near future the main tension in the design of CVE systems will be between control of content, which is facilitated by central servers with relatively fixed game behaviours, and open systems that users use to build their own environments. If a truly open system is built, then games authors will no longer be writing the system, but will be focussing instead on describing the behaviours and content of the world and relying on a standard and widely used set of services.

This open system is probably far from being specified, and there are still significant challenges to be met. We still do not have mature modelling tools for describing CVEs. A particular CVE system tends to use only one data-sharing mechanism, which limits scalability and flexibility. And finally, implementation frameworks are wide ranging, and there is no universal agreement what behaviours should be provided by services within the system and what behaviours should be embedded into the description of the specific CVE that is being built.

References

Airey, E.J.M., Rohlf, J.H., & Brooks Jr., F.P. (1990). Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics, Proceedings of ACM Symposium on Interactive 3D Graphics, 24*(2), 41-50.

Benford, S., & Greenhalgh, C. (1997). Introducing third party objects into the spatial model of interaction. In J.A. Hughes, W. Prinz, T. Rodden, & K. Schmidt (Eds.), *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work* (ECSCW'97) (pp. 189-204). Kluwer Academic Publishers.

Capps, M., McDowell, P., & Zyda, M. (2001). A future for entertainment-defense research collaboration. *IEEE Computer Graphics & Applications,* (January/February), 37-43.

Carlsson, C., & Hagsand, O. (1993). DIVE—a platform for multi-user virtual environments. *Computers & Graphics, 17*(6), 663-669.

Coates, S. (2001). London wasn't built in a day: Content acquisition for levels in The Getaway. Retrieved December 17, 2003 from: *www.gamasutra.com/features/20010321/coates_01.htm*

Dassault Systemes. (2003). CATIA. Retrieved December 17, 2003, from *www.3ds.com/en/brands/catia_ipf.asp*

Division Ltd. (1996). *dVS for UNIX workstations.* Developer Guide, Revision 3.0.

Floyd, S., Jacobson, V., Liu, C., McCanne, S., & Zhang, L. (1997). A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking, 5*(6), 784-803.

Frécon, E., Smith, G., Steed, A., Stenius, M., & Stahl, O. (2001). An overview of the COVEN platform. *Presence: Teleoperators and Virtual Environments, 10*(1), 109-127.

Frécon, E., & Stenius, M. (1998). DIVE: A scalable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal, 5*(3), 91-100.

Freeman, D., Slater, M., Bebbington, P., Garety, P.A., Kuipers, E., Fowler, D., Met, A., Read, C., Jordan, J., & Vinayagamoorthy, V. (2003). Can virtual reality be used to investigate persecutory ideation? *The Journal of Nervous and Mental Disease.*

Fuchs, H., Kedem, Z.M., & Naylor, B.F. (1980). On visible surface generation by a priori tree structures. *Computer Graphics, Proceedings of ACM SIGGRAPH 80, 14*(3), 124-133.

Greenhalgh, C. (2002). EQUIP: A software platform for distributed interactive systems. Retrieved December 17, 2003, from *www.crg.cs.nott.ac.uk/~cmg/Equator/Downloads/docs/equip-tech.pdf*

Greenhalgh, C.M., & Benford, S.D. (1995). MASSIVE: A virtual reality system for tele-conferencing. *Transactions on Computer Human Interfaces (TOCHI), 2*(3), 239-261.

Grimsdale, G. (1991). dVS—distributed virtual environment system. *Proceedings of Computer Graphics 1991 Conference,* London.

Hesina, G., Schmalstieg, D., Fuhrmann, A., & Purgathofer. W. (1999). Distributed open inventor: A practical approach to distributed 3D graphics. In D. Brutzman, H. Ko, & M. Slater (Eds.), *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 74-81).

IAI-International. (2003). International Alliance for Interoperability, Industry Foundation Classes, IFC2x Edition 2, May. Retrieved December 17, 2003 from: *www.iai-international.org/iai_international/Technical_Documents/iai_documents.html*

IEEE. (1993). IEEE standard for information technology—protocols for distributed simulation applications: Entity information and interaction. *IEEE Standard 1278-1993.* New York: IEEE Computer Society.

Kelso, J., Steven G., Satterfield, L.E. Arsenault, P., Ketchan, M., & Kriz, R.D. (2003). DIVERSE: A framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations. *Presence: Teleoperators and Virtual Environments, 12*(1), 19-36.

LivingWorlds. (1998). LivingWorlds: Making VRML 97 applications interpersonal and interoperable. Retrieved December 17, 2003 from: *www.web3d.org/WorkingGroups/living-worlds/*

Macedonia, M.R., Zyda, M.J., Pratt, D.R., Barham, P.T., & Zeswitz, S. (1994). NPSNET: A network software architecture for large scale virtual environments. *Presence: Teleoperators and Virtual Environments, 3*(4), 265-287.

Macedonia, M.R., Zyda, M.J., Pratt, D.R., Brutzman, D.P., & Barham, P.T. (1995). Exploiting reality with multicast groups: A network architecture for large scale virtual environments. *Proceedings of the 1995 IEEE Virtual Reality Annual Symposium* (pp. 2-10).

Reddy, M., & Iverson, L. (2002). GeoVRML 1.1 Specification. Retrieved December 17, 2003 from: *www.geovrml.org/1.1/doc/*

Shaw, I. (2003). Personal communication, September.

Singhal, S., & Zyda, M. (1999). *Networked virtual environments: Design and implementation.* Addison-Wesley.

Steed, A., Frécon, E., Avatare Nöu, A., Pemberton, D., & Smith, G. (1999). The London travel demonstrator. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology 1999* (pp. 50-57), University College, London, December 20-22. ACM Press.

Steed, A., Mortensen, J., & Frécon E. (2001). Spelunking: Experiences using the DIVE system on CAVE-like platforms. In B. Frohlicj, J. Deisinger, & H.J. Bullinger (Eds.), *Immersive Projection Technologies and Virtual Environments 2001* (pp. 153-164). Vienna: Springer-Verlag.

Strauss, P., & Carey, R. (1992). An object-oriented 3D graphics toolkit. *Proceedings of SIGGRAPH'92* (pp. 341-349).

Teller, S.J., & Sequin, C.H. (1991). Visibility preprocessing for interactive walkthroughs. *Computer Graphics, Proceedings of SIGGRAPH 91, 25*(4), 61-90.

Tromp, J., Steed, A., & Wilson, J. (2003). Systematic usability evaluation and design issues for collaborative virtual environments. *Presence: Teleoperators and Virtual Environments, 10*(3), 241-267.

Waters, R.C., Anderson, D.B., Barrus, J.W., Brogan, D.C., Casey, M.S., McKeown S.G., Nitta, T., Sterns, I.B., & Yerazunis, W.S. (1997). Diamond Park and Spline: Social virtual reality with 3D animation, spoken interaction and runtime extendability. *Presence: Teleoperators and Virtual Environments*, *6*(4), 461-481.

Wray, M.J., & Hawkes, R. (1998). LivingSpace: Distributed virtual environments and VRML: An event-based architecture. *Computer Networks and ISDN Systems*, *30*, 43-51.

X3D Working Group. (2003). Extensible 3D (X3D™ graphics, X3D Working Group. Retrieved December 17, 2003 from: *www.web3d.org/x3d.html*

Endnotes

[1] CAVE is a registered trademark of the University of Illinois' Board of Trustees. CAVELib is a registered trademark of the University of Illinois' Board of Trustees.

Chapter X

Toward a User-Centred Method for Studying CVEs for Learning

Daphne Economou
University of the Aegean, Greece

Steve Pettifer
University of Manchester, UK

Abstract

This chapter addresses one of the challenges the collaborative virtual environments (CVEs) research community faces which is the lack of a systematic approach to study social interaction in CVEs, determine requirements for CVE systems design, and inform the CVE systems design. It does this by presenting a method for studying multi-user systems in an educational context. The method has been developed as part of the Senet project, which is investigating the use of virtual actors in CVEs for learning. Groupware prototypes are studied in order to identify requirements and design factors for CVEs. The method adopts a rigorous approach for *organizing experimental settings, collecting and analysing data, and informing CVE systems design. The analysis part of the method shares many of the Interaction Analysis foci and expands on it by providing a grid-based method of transforming rich qualitative data in a quantitative form. The outcome of this analysis is used for the derivation of design guidelines that can inform the construction of CVEs for learning. The method is described by a third phase of work in the Senet project.*

Introduction

Collaborative virtual environments (CVEs) aim to provide effective means of using computers as tools for distributed communication and may be used for diverse tasks such as people working together, communication, education, or entertainment. Real-world collaborative work involves a considerable and complex information exchange (Hutchins, 1990; Heath & Luff, 1991, 1996; Hutchins & Klausen, 1996; Suchman, 1996; Bellotti & Rogers, 1997; Harper, 1997), and all such systems rely on the provision of functionalities and metaphors that emulate human-human interaction in order to facilitate computer-mediated communication, and support interaction and collaboration.

Informing the design of CVEs demands an understanding of the social interaction these types of environments afford. However, an immense problem faced by the CVE research community is the lack of a systematic user-centred methodology for studying social interaction in CVEs to inform the design of CVE systems (Durlach & Mavors, 1994; Steed & Tromp, 1998; D2.9, 1999; Kaur Deol, Steed, Hand, Istance, & Tromp, 2000a, 200b; Benford, Greenhalgh, Rodden, & Pycock, 2001). CVEs systems' development so far has been driven by the challenge of providing novel solutions to technological concerns. There is an existing body of work looking at user needs, but this is primarily from the perspective of usability (Kaur, 1998; Stanney, Mourant, & Kennedy, 1988). What is needed is to broaden that perspective to recognise the situated and social nature of the processes in collaboration.

This chapter describes a systematic method that aims to identify the design factors involved in the construction of CVEs for learning, to understand the respective properties that these are formed from, and to inform CVE systems design. The method has been developed as part of the Senet project, which studies the role of virtual actors in CVEs for learning (Economou, Mitchell, & Boyle, 2000). An exploratory phased approach is adopted where robust prototypes are constructed and studied in order to determine requirements and design factors for CVEs. The study is based on a real-world situation to determine real requirements for CVE technology. Despite the exploratory nature of the work, a rigorous method is adopted for organizing experimental settings, collecting and analysing data, and informing CVE systems design. Data collection occurs by video-recording the users and user activities in the CVE, keeping record of textual communication, and note taking. The analysis part of the method is partly based on Interaction Analysis, and results in a mixture of quantitative and qualitative findings. The method deals with transforming rich qualitative data in a quantitative form that is used to draw design guidelines for CVEs for learning. Design guidelines help build up a substantial body of knowledge in a particular context (Newman & Lamming, 1995) and provide a means of communicating system requirements to clear advice for implementations to software engineers. The application of the method involves the use of the Deva CVE system (Pettifer & West, 1999) to create a multi-user board game for use in museum education.

Background and Motivation

> *"Although it is important not to try to simply replicate what we think of as 'reality', when designing systems to support collaborative work we can learn a great deal from observations of people working and collaborating together in conventional settings."* (Snowdon, Churchill, & Munro, 2001, p. 8)

The above statement is supported by a number of researchers (Moran & Anderson, 1990; Heath & Luff, 1991; Engeström & Middleton, 1996; Bowers & Martin, 1999). The rationale underpinning the development of rich collaborative CVEs is the desire to develop an arena where the interactive experience can be supported satisfactorily. However, several problems in terms of studying and analysing social interaction in CVEs have been identified:

- the vast amount of factors involved in the construction of CVEs and virtual actors,
- the current immaturity of the technology,
- the prototypical nature of current applications, and
- lack of rigorous research methodologies for studying and informing CVE design.

Kaur (1997) has identified 46 design properties to be considered when designing VEs for usability. This number of factors increases dramatically when considering CVEs that require the consideration of: interaction, communication, collaboration between users and the CVE, users and objects contained in the CVE, and population. This complicates the isolation of the design decisions responsible for the overall effectiveness of the environment and the inter-play between various factors.

The current immaturity of the technology does not allow the full potential of the CVEs to be exploited. This means that many of the applications developed so far have been of a prototypical nature. There are two issues in respect to the prototypical nature of applications: (1) it is often not feasible to create different conditions for experiments within the time and effort available, thus the process of studying specific phenomena is constrained; and (2) defects in the prototypical functionality of the application might cause difficulties in conducting studies with real users (Steed & Tromp, 1998). The technology is not mature enough to afford the activities that such complicated environments require.

Due to the 'newness' of VR and CVE technologies, currently there is no systematic research approach for studying and informing VR (Durlach & Mavors, 1994; Kaur Deol et al., 2000a, 2000b) and CVE (Steed & Tromp, 1998; Tromp, 1999) system design. It will be some time before VR systems are built using systematic methodologies to model and verify the system design (Mills & Noyes, 1999). Insights to VR and CVE systems are coming mainly from the fields of human-computer interaction (HCI) and computer-supported cooperative work (CSCW). However, there is a call for a systematic method for studying human interaction, managing a large amount of disparate data, and producing results that directly inform the CVE system design. The outcome of a workshop on usability evaluation for VEs emphasised the need for extracting generalisable and re-usable results from user studies (Kaur Deol et al., 2000b). This is a challenge, as one of the problems in CVE research is the lack of a formal method for studying and evaluating CVEs.

Towards a Methodology for Studying CVES for Learning

The overall research methodology followed in this project has been presented elsewhere (Economou et al., 2000). The two following sections briefly discuss some important points concerning the methodology addressing the above problems.

Three novel aspects characterise this methodology:

- It uses a 'real-world' application to determine requirements for CVEs.

- It follows an exploratory iterative approach of breaking the problem into a series of manageable phases of increased sophistication, which provides the means of managing the complexity, allows the results of each phase to inform subsequent phases, and allows requirements to be progressively identified and evaluated.
- It follows rigorous steps to study social aspects of CVEs, organise experimental settings, collect and analyse the data, produce design guidelines for the use of virtual actors in CVEs for learning, and inform the development of underlying virtual actor technology.

Real-World Application

Problems arising in a real-world situation can determine the success or the failure of the system (Gunton, 1993). In order to study an authentic learning activity, the research was based around the work of Manchester Museum's Education Service (Mitchell, 1999).

This service caters for school visits to the museum aimed at Key Stage Level 2 (9-11 years old). It provides access to a wide range of museum artefacts relevant to subjects in the National Curriculum for education. One particular strength of the museum, with a major part in the Education Service's teaching, is its collection of everyday life ancient Egyptian artefacts from the town of Kahun. The artefact chosen as the basis of the learning activity in this research is Senet—a board game for two players. Players take turns to throw a die. The object of the game is to "bear off" your 10 pieces first. Through the activity and a collaborative process, the children get familiar with the artefact and learn by using it how it was played.

Developing a CVE based on Senet provides a good testbed for various CVE properties. It allows object manipulation (the board, die, and pieces), individual operations, as well as operations in pairs or as larger groups. In terms of collaboration it allows cooperation (to learn the game) as well as competition (to win the game). The game situation allows a range of teaching styles from traditional instructional methods (e.g., explaining the rules in advance) to constructivist methods (learning by playing). Current educational thought recognises the need for sociocultural methods that emphasise the social roles of teachers and learners (Soloway et al., 1996). A more practical impetus for collaborative learning has come from two main sources in the UK. The National Curriculum for education places great emphasis on such learning. In addition, the UK Government has proposed a National Grid for Learning (NGfL) (DfEE97, 1997; DfEE98, 1998).

The game supports the needs of experimentation in various ways. It is a fairly well- structured task (the players have to follow certain steps to learn the rules and play the game). The length of the required time to play matches well *** the length of time the children could take part in a task before becoming restless (30-45 minutes). Players' knowledge assessment can occur in a fairly unobtrusive manner (e.g., by observing if they follow the rules).

A Phased Approach

The research approach is of an exploratory nature. The studies being carried out are not evaluations, but observations of what is going on. There is a great need for more exploratory study of novel learning applications. Roussos et al. (1999) call for the building of novel learning applications and carrying out informal evaluations of them in use.

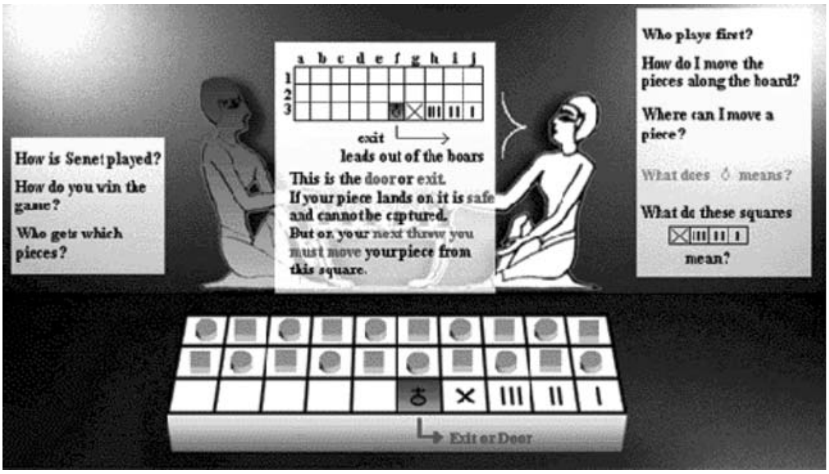
In the Senet project a 'low-tech prototyping' approach was adopted. The project was divided into three phases. In the first two phases of the project, more mature technologies (single display groupware and conventional groupware) were used. This was in order to study social interaction factors in isolation, and construct robust prototype collaborative learning applications. These prototypes were then observed in use in order to identify the types of interactivity and social communication that would need to be supported in a complete CVE.

In the first phase, a prototype application was developed that took the form of a single display groupware (Stewart, Bederson, & Druin, 1999; Bullock et al., 2001). Users see the Senet board and pieces, and can also access the rules of the game (Figure 1). Users sit next to each other and view the application on a single, shared display. The interactions between them were external to the computer. The prototype was constructed using 2D multimedia tools. This helped to simplify issues surrounding the navigation and object manipulation. The purpose of this phase was to gather what goes on in such a 'real-world' game playing situation:

- the types of interactions that occur between the users and the game environment,
- the communication between users (content and modes),
- the roles that the users adopt in a game playing situation, and
- controls over the communication and the game playing activity.

The study also aimed to identify usability issues surrounding the prototype in order to inform the design of environments developed in subsequent phases.

Figure 1. Single display groupware



How is Senet played?
How do you win the game?
Who gets which pieces?

This is the door or exit.
If your piece lands on it is safe and cannot be captured.
But on your next throw you must move your piece from this square.

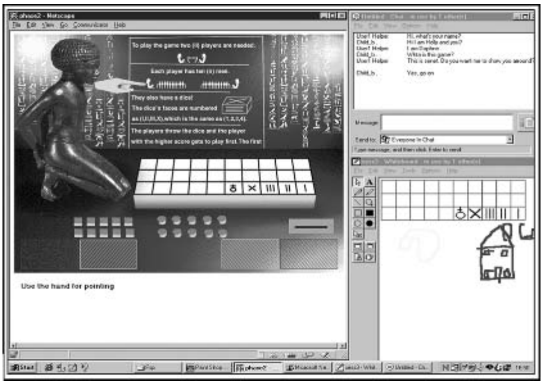
Who plays first?
How do I move the pieces along the board?
Where can I move a piece?
What does ☥ means?
What do these squares ☒|III|II|I mean?

The second-phase prototypes developed took the form of conventional groupware systems. Participants were remotely located so interactions between them were internal to the computer. The prototypes were developed using 2D multimedia tools coupled with groupware technology typical of that used in education (NetMeeting). The prototypes introduced the concept of population to the CVE as the users were represented with virtual actors. The purpose of this phase was to explore issues surrounding the interaction being internal to the environment and the effects on the behaviour of participants:

- *Appearance:* The users' representation via their virtual actors.
- *Awareness:* What the virtual actor can perceive about the VE and the situation.
- *Interaction:* With objects in the environment and the environment itself.
- *Turn-Taking:* To communicate, to interact with objects or other users.
- *Communication Content:* The actual communicative exchanges.
- *Communication Modes:* The ways in which the virtual actor's communication can be presented (e.g., text, speech).
- *Pedagogy:* The roles and tactics users adopt for delivering various topics.

Three prototypes were developed to satisfy the purpose of this phase. In the first prototype, a child was playing with an expert (a researcher who adopted the role of the teacher). The rules of the game where provided in the Senet environment (as part of the decoration on the wall). The users could see each other via the same virtual actor. The communication was text based, via the tools NetMeeting provided (see Figure 2(a)). The second prototype followed the same scenario, with the difference being that the source of the rules was the expert, and the users were represented with their own virtual actor. The third prototype was similar to the second one, with the difference being that two children were playing with each other and the expert took the role of the mediator. In the last two prototypes, the users communicated by typing text in chat boxes associated with their own actor or using a hand for pointing (see Figure 2(b)).

Figure 2(a). Conventional groupware prototype—dialogue external to the game environment



To play the game two (II) players are needed.

Each player has ten (X) men.

They also have a dice!

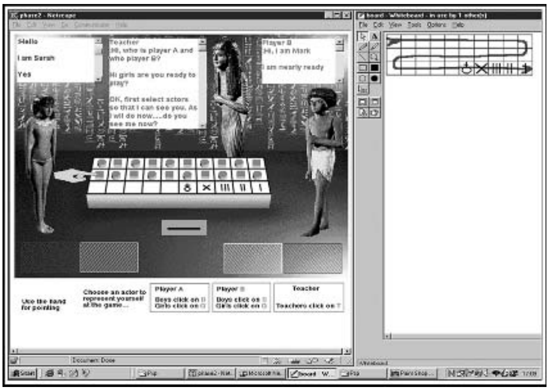
The dice's faces are numbered as (I,II,III,X), which is the same as (1,2,3,4).

The players throw the dice and the player with the higher score gets to play first. The first

Use the hand for pointing

User1 Helper: Hi, what's your name?
Child_b: Hi I am Holly and you?
User1 Helper: I am Explorer
Child_b: What is the game?
User1 Helper: This is senet. Do you want me to show you around?
Child_b: Yes, go on

Figure 2(b). Conventional groupware—dialogue internal to the game environment



phase2 - Netscape
board - Whiteboard - in use by 1 other(s)
File Edit View Tools Options Help
Document: Done
Start

Hello

I am Sarah

Yes

Teacher
Hi, who is player A and who player B?

Hi girls are you ready to play?

OK, first select actors so that I can see you. As I will do now....do you see me now?

Player B
Hi, I am Mark

I am nearly ready

Use the hand for pointing

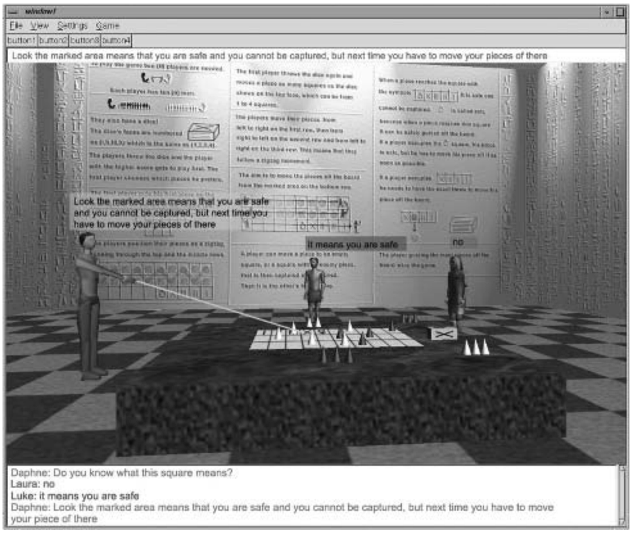
Choose an actor to represent yourself at the game...

Player A
Boys click on [illegible]
Girls click on [illegible]

Player B
Boys click on [illegible]
Girls click on [illegible]

Teacher
Teachers click on T

Figure 3. Senet prototype in Deva


window1
File View Settings Game
button1 button2 button3 button4
Look the marked area means that you are safe and you cannot be captured, but next time you have to move your pieces of there
It means you are safe
no

Look the marked area means that you are safe and you cannot be captured, but next time you have to move your pieces of there

Daphne: Do you know what this square means?
Laura: no
Luke: it means you are safe
Daphne: Look the marked area means that you are safe and you cannot be captured, but next time you have to move your piece of there

In the third phase, an application has been constructed using the Deva 3D CVE technology according to preliminary design guidelines identified in previous phases (see Figure 3). The study was evaluative as well as exploratory. All the interactions were internal to the CVE. Two children were playing against each other and the expert took the role of the mediator. The users were represented with their own virtual actor, which could: (1) walk, indicating the user's position in the space; (2) point (in which case there is a "laser pointer" from their hand); and (3) select and move objects (by positioning the actor's hand to touch the object, or by pointing in case the object is not within the actor's reach). Communication was 2D text ('text bubble') above the 'speaking' actor's head, or a 'transcript' window outside the game environment, which kept a history of the dialogue. When an actor who was out of viewpoint spoke, warning text appeared at the left or right edges of the screen depending on the speaker's location relative to the listener, indicating who was talking (e.g., "*the user's name* is talking"). The user's dialogue appeared in a transcript window when the return key was pressed.

The phased approach provides several benefits such as managing complexity by dealing with a manageable set of factors in each phase (e.g., 2D/3D and population) and allowing the results of each phase to inform subsequent phases. Thus, requirements can be progressively identified. The use of more robust technologies allows the essential features of the situation (interactivity and social communication) to be studied with real users in a way not possible with more immature and inaccessible CVE technology.

The studies in the first two phases aimed at deriving a rich set of qualitative information. From this, a set of requirements has been identified and then used to inform the design of the third-phase application. The work in the first two phases has been seen to be of a more exploratory nature, more like formative evaluation in contrast to the work in the third phase, which involved evaluation of a more summative nature.

A Rigorous Method for Studying Social Interaction in CVEs

Despite the exploratory nature of the work, the primary purpose of the method to be adopted is a form of requirements gathering that follows rigorous steps to enable the identification of design factors in a way that can directly inform CVE systems design. Candidate methods such as *conversation analysis* (Atkinson & Heritage, 1984; Boden & Zimmerman, 1991; Silverman, 1997) and *discourse analysis* (Coulthard, Montgomery, & Brazil, 1981) are narrowly focussed on issues surrounding the dialogue itself. Intimate and subjective study of human activities and interaction requires a permanent record of naturally occurring events (e.g., field notes, video, audio) (Luff, Hindmarch, & Heath, 2000). Ethnographic approaches contribute to understanding the production of social actions and activities, and recognise the activities of others. However, coupled with video, it results in a vast amount of rich qualitative data. The complexity of dealing with video data has been recognised by a growing number of researchers (including Silverman, 2000). It is not only unmanageable, but the moment-to-moment detailed analysis is notoriously time consuming (Allen, 1989; Neal, 1989). The information is interrelated, and it is difficult to be separated and rationalised. Viller and Sommerville (1999) argue that it is difficult to draw design principles and other abstract lessons from a technique that is concerned with detail of a particular situation. Thus, it is difficult to make generalisations about design factors related to CVEs. The analysis needs to be practised by a group of analysts to overcome subjectivity.

One method for which video technology is essential is ‘Interaction Analysis’ (Jordan & Henderson, 1995). This method has its roots in the social sciences, and sees knowledge and action as fundamentally social in origin, organisation, and use. It studies human activities such as talk, non-verbal interaction, and the use of artefacts and technologies. It is primarily defined by its ‘analytic foci’ or ways into a videotape. Such foci include: structure of events, temporal organisation of activity, turn-taking, trouble and repair, and spatial organisation of activity. Important to Interaction Analysis is the data analysis by a group of analysts which goes some way to countering subjectivity of analysis. However, group-based analysis is not always possible (as in the case of the Senet project) because of resource limitations.

The proposed solution that addresses these problems is the creation of an analytic grid that can be used to generate numerical values from the qualitative data (this is discussed in detail below). For example, if the factor to be studied is physical activities that the virtual actors should be eligible of to improve communication and interaction issues, then the quantitative information subtracted out of the qualitative data should indicate in which circumstances and for what purpose certain physical activities have been used. In this form the data is much more manageable and can be linked forward more reliably to the design factors that are developed.

The analytic foci and orientation adopted in the method used to study the Senet project, outlined next, is based and adds on the Interaction Analysis foci. The method follows rigorous steps for organizing experimental settings, collecting and analysing data, and provides the means of managing large amounts of disparate data (videotapes, field notes, text files). It consists of seven main steps, which are carried out sequentially:

- data collection
- transcription
- chunking of the transcription
- creation of a grid
- application of the grid
- analysis at the session level
- derivation of design guidelines

The seven-step method has been applied to the second phase of the study and derived preliminary set of design guidelines (Economou, Mitchell, Pettifer, Cook, & Marsh, 2001) that directed the development of the third phase prototype CVE. The method has subsequently been applied to the third phase of the project to evaluate the effectiveness of the implemented preliminary design guidelines and to investigate new factors arising in a 3D CVEs for learning. The method can be repeated according to the analytic categories to be studied.

The section below outlines the processes involved in each step of the seven-step method in the third phase of the Senet project.

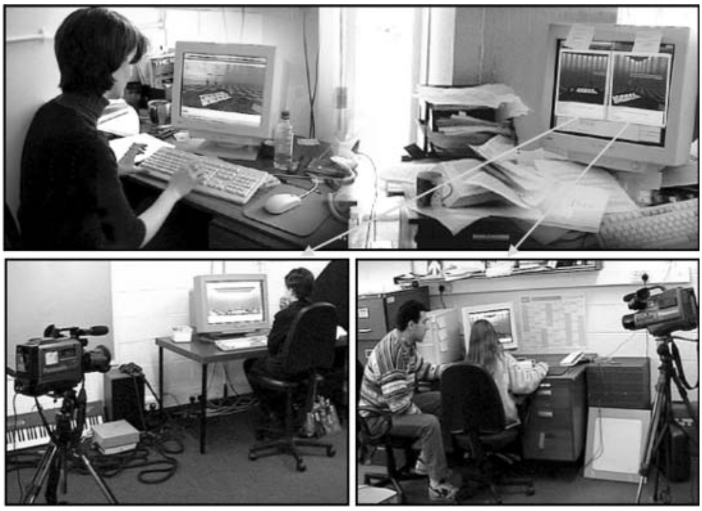
The Method Applied

Data Collection

The third phase of the study was conducted in the laboratories of the Advanced Interfaces Group at the University of Manchester. Twelve (six pairs) of 12-year-old children participated in the studies. Three rooms were used: One contained a researcher playing the role of the 'expert' (E), and the other two rooms each contained a 'child' actively participating in the activity (AC), accompanied by a second researcher, a 'helper' (H) providing technical support (Figure 4). The third-phase prototype (Deva prototype) has been used. The children were introduced to the use of the Deva tools (e.g., the mouse controls and communication tools), and afterwards they were asked to carry out various tasks such as: read the rules and set up the board, learn how to play the game, and play the game.

The children were videotaped individually. The video cameras were set to capture the users' interactions with the artefacts and other users in the CVE. Screenshots of one of the child's screens, providing a detailed record of interactions between users, were also videotaped. A transcription of the users' textual communication saved in a file provided a permanent record of the user's dialogue. The transcription provided a record of the sequential organisation of the user's turns to talk and the exact time of the exchange. For capturing the expert's activities, the 'think-aloud' method was used (Monk, Wright, Haber, & Daven-

Figure 4. The physical set up for the third-phase study



port, 1993) and tape recorded. It is one of the few methods of getting a record of the user's mental activity. The way it works is that the users think aloud about their activities in terms of mental reasoning (e.g., the expert described aloud her actions, decision making, and observations while playing with the children). This allowed the close study of problems such as the expert's lack of awareness of the child's exact situation. Questionnaires filled out by the children before the session obtained background information about the children.

Each session lasted approximately 45 minutes. Each session was followed up by an interview with the children about their experiences, which lasted approximately 10 minutes and was tape-recorded.

Transcription

The transcription step involves creating one account of a session by combining data that captures interactions taking place internally (e.g., users' dialogue in the

Table 1. An illustration of a part of an example session transcription

	The helper approached the AC.
Communication external to the system, between the H and the AC	*"H": "Have a sit. These machines are connected up and we are going to learn how to play this game. 1* Someone is going to help you 2* . Not me 3* .It will be someone else on the other machine."*
	1* The H pointed the screen
	2* The H pointed the room next door
	3* The H pointed himself and smiled looking at the AC.
	The AC nodded positively. The H was smiling at her.
	"H": "Type a greeting in that box and press return."
	The H showed the Deva message box that AC had to use to type messages.
	The AC came closer to the screen.
Simultaneous external interactions	*"H": "Have you used computers before?"*
	The H was looking the AC.
	"AC": "Yes"
	The AC started typing.
	The H was looking the screen while the AC was typing the message.
Communication internal to the system	**"02/03/99 11:27:07","AC:","hello"**
	"H": "All right."
	The H moved away.
	"H": "Wait they will come back to you."
	Pause the AC was waiting for the E's response.
	"H": "So are you and Cathy from the same class?" *"AC": "Yes"*
	The AC turned and looked the H, then turned back to her machine.
	"02/03/99 11:30:18","E":,"hello, what's your name?"
	"H": "She is coming back to you."
	The H talked to the AC from distance.
	"AC": "Aha!"
	The AC was surprised, she took the mouse and tried to reply

AC = child actively participating in the activity; E = expert—a knowledgeable user playing the role of a teacher; H = helper providing technical support

CVE, transcription of screen activities) and externally to the prototype (e.g., dialogues between people in the real world—the child and the helper; nonverbal interaction). The videotapes in the current study were synchronised, based on noticeable events on all videotapes. Transcriptions provide a more manageable way of handling the data (see Table 1).

Chunking of the Transcription

Dividing the transcription into a series of ethnographic chunks provides a more manageable set of units for analysis. According to Interaction Analysis, one such chunk is the *event*, which is a stretch of interaction that coheres in some manner that is meaningful to the participants. Events can be named and constitute recognisable, culturally significant tokens in social communication. Jordan and Henderson (1995) refer to tutoring sessions, bedtime stories, as examples of recognisable events. Events in turn can be sub-divided into a set of *segments* (e.g., in a meal event segments such as 'setting up the table' or 'serving the coffee' can be identified).

To identify ethnographic chunks it is necessary to draw on cultural knowledge or local experts. For this particular study four ethnographic chunks were identified:

- session
- stage
- segment
- turn

The *session* is equivalent to a game-playing event. A session consists of a series of *stages*. The stages characterise the changes in topic (e.g., introduction of the system tools, explanation of the game, playing the game). The structure of a session depends on the prototype used for the study and the teaching strategy adopted based on how the situation unfolds. For example, Table 2 shows a typical session structure that consisted of seven stages.

Table 2. An example session stage structure

1-2.	H explained the system tools to the AC
3.	E directed the AC to go and read the rules and gather around the board when they were ready to start
4.	AC dealt with the first part of the rules first, decided who plays first with which pieces and set up the board
5.	AC dealt with the second part of the rules that deals with the way pieces can be moved
6.	AC play the game
7.	the session closed

AC = child actively participating in the activity; E = expert—a knowledgeable user playing the role of a teacher; H = helper providing technical support.

Stages are sub-divided into segments. *Segments* are differentiated by a change in the pedagogical tactics employed. The current study recognises pedagogical tactics as the methods that a teacher employs for delivering a topic. These are elements of learning theories the teachers use during the tutoring process (e.g., the teachers use the blackboard, they play a video, or involve children in hands-on activities), which when they coalesce form a pedagogical approach (e.g., instructional learning, cognitive apprenticeship). A pedagogical tactic is defined by a sequence of actions, which in the current system of transcription form a segment. For example, in the first stage of the above example (where the helper explains the system tools to the child), two segments can be identified (Table 3). In the first segment the helper explains to the child how communication takes place with the expert, located close to the child. In the second segment the helper intervenes to cover the delay.

The last ethnographic chunk unit is the *turn*. The boundary of a turn is marked by the other participant taking control. There are two kinds of turns: *internal* (a turn using one of the prototype’s tools, e.g., typing in a chat box or moving an

Table 3. Two segments of the first stage of the example session

Stage	Segments	Turns	Transcriptions
1	**a**	**1ex**	The helper approached the AC. *“H”: “Have a sit. These machines are connected up and we are going to learn how to play this game. 1* Someone is going to help you 2* . Not me 3* .It will be someone else on the other machine.”* 1* The H pointed the screen 2* The H pointed the room next door 3* The H pointed himself and smiled looking at the AC.
		2ex	The AC nodded positively. The H was smiling at her.
		3ex	*“H”: “Type a greeting in that box and press return.”* The H showed the Deva message box that AC had to use to type messages. The AC came closer to the screen. *“H”: “Have you used computers before?”* The H was looking the AC.
		4ex	*“AC”: “Yes”* The AC started typing. The H was looking the screen while the AC was typing the message.
		1in	`"02/03/99 11:27:07","AC:","hello"`
		5ex	*“H”: “All right.”* The H moved away.
	b	**1ex**	*“H”:“Wait they will come back to you.”* Pause the AC was waiting for the E’s response. *“H”:“So are you and Cathy from the same class?”*
		2ex	*“AC”: “Yes”* The AC turned and looked the H, then turned back to her machine.
		1in	`"02/03/99 11:30:18","E":,"hello, what’s your name?"`
		3ex	*“H”:“She is coming back to you.”*
			The H talked to the AC from distance.
		4ex	*“AC”: “Aha!”* The AC was surprised, she took the mouse and tried to reply

H = provides technical support; AC = child actively participating in the activity object) and *external* (between participants external to the prototype). In the above example of two segments, two internal and nine external turns can be identified (explained in detail in Table 3).

Depending on the analytic category (factor) the sessions are analysed for, it can be decided which ethnographic chunk to consider (the formation of analytic categories are discussed in the section, “Creation of a Grid”). For example, if the focus of the study is to examine what marked the users turn, then the sessions are studied down to a turn level. When broader issues are under study, like pedagogical methods adopted in different sessions, analysing the sessions in units as small, as turns would be unreasonable. This is because a pedagogical method can be identified, considering a set of actions that involve many exchanges of turn between active participants. In such cases, whole segments and even stages are studied.

To aid referring to an ethnographic chunk a coding system that captures the stage, the segment and the turn has been developed. This works from left to right. For example, **1_a_1in** or simply **1a1in** stands for: the **1**, for first stage, the **a**, for the first segment, and the **1in**, for the first internal turn. External turns are coded as **ex**.

Creation of a Grid

The moment-to-moment detailed analysis of each ethnographic chunk leads to a vast amount of rich qualitative data, which comes into an unmanageable form. Repetitions of actions are difficult to associate with certain events and follow throughout a whole session. This makes generalisations about design factors difficult. In addition, the interpretation of users’ behaviours is subjective and depends on individual analysts’ experience.

To address these problems, the development of a ‘grid’ is proposed which aims to generate countable values out of the rich qualitative data. The grid is formed of analytic categories (Table 4). To identify the analytic categories to appear in a grid, it is necessary to have a clear idea about the factors to be studied. This requires a hypothesis at the beginning of the study. When the study has an exploratory nature, like the current research, having a hypothesis at the beginning of the study is unadvised (Glaser & Strauss, 1967). The impetus for the analytic categories to form the grid is provided by:

- a framework of design factors based on literature (Benford, Bowers, Fahèn, Greenhalgh, & Snowdon, 1995; Capin, Pandizc, Chauvineau, Thalmann, & Thalmann, 1996);
- findings from an exploratory study that aimed to gather the key interactional affordances of a situation that a CVE system design should support—this

Table 4. A basic form of the grid

	Analytic category 1			Analytic category 2						
Turn index	Type 1	Type 2	Type …n	Type 1	Type 2	Type …n	Location		Description	Participants
							Internal interaction	External interaction		

is the role of the first and second phase of the method outlined earlier in section, "A Phased Approach."

The analytic categories identified to study in the third phase of this research using the Deva prototype were:

- turn taking
- physical activity
- communication activity
- external intervention
- pedagogy

Each analytic category is formed of different analytic themes. The analytic themes change according to the situations to be studied, which might be different along different phases of a long-scale study. For example, the themes studied in physical activity analytic category in the second phase of the study were: *head movement*—where the participants are looking, nodding, facial expressions; *position of the body*; *movement of the rest of the body excluding the head*—walking, pointing, moving objects, gesturing. Physical activity in the third phase of the study studied virtual actors' head movement—direction of gaze identified by the users' viewpoints; position of the virtual actors' body; movement of the virtual actors' body excluding the head—walking, pointing, moving objects.

Four columns are common to all the analytic categories:

- *Turn Index*, which identifies an exact part of a transcription down to a turn level (e.g., 1b5in).
- *Location*, which indicates the relative actions internal versus external to the prototype.
- *Description*, which provides a summary of the turn content.
- *Participants*, which indicates responsible users for a certain activity described in the grid.

Application of the Grid

To illustrate the use of the grid, the same example session transcription is analysed (one of the third-phase sessions using the prototype CVE based on the Deva system; see Figure 3) for each of the analytic categories. The example setting was explained earlier.

Turn Taking

The turn taking analytic category examines the sessions down to a turn level to find what marked users' turns including:

- textual boundaries (e.g., text appearing in text bubbles, or the transcript window);
- internal visual cues the system provided (e.g., virtual actors' features, prompt that somebody is talking).

Using the example session transcription, **180** out of the **402** users' turns have been marked by textual turn boundaries, and **222** by visual ones. Of the **180** textual boundaries: **168** were based on **text bubbles**, indicating the users' activity status, talking or finishing their utterance, and thus awaiting a reply; and **11** on the **transcript window**. Of the **222** visual boundaries: **47** were virtual actors walking in the CVE; **46,** the virtual actors' position in the CVE and virtual actors' viewpoint; **82,** the virtual actors interacting with objects and pointing; and **47,** the virtual actors talking (text appearing in text bubbles).

The descriptive part of the grid contributes to a better understanding of the type of internal visual boundaries that marked users' turns, and showed that the above actions indicated: the users' *on-going activities* and the *user responsible for certain actions*; *the users' focus of attention*; *the users' process of activity* (the beginning and completion of actions); *the users' intention of action* and *offering of a turn*; as well as *users encountering difficulties*.

Physical Activity

The physical activity analytic category examines the sessions down to a turn level and is concerned with:

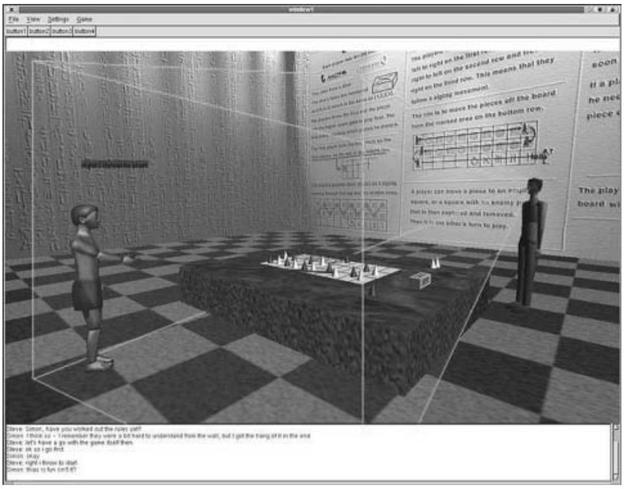
- virtual actors' head movement, where looking was captured via the monitors displaying the users' viewpoints (see Figure 4), and the direction the virtual actors were facing (see Figure 5);

- position of the virtual actors' bodies;
- movement of the virtual actors' bodies, excluding the head: virtual actors walking, pointing, and moving objects in the CVE.

In the session being examined, **270** occurrences of physical movements were recorded: **75** were **head movements**: virtual actors' orientation; virtual actors' direction of gaze (identifying where the participants were looking based on their viewpoint); **89** of the **270** occurrences of physical movements were the virtual actors' position of the body in the CVE, while **106** were related to the movement of the rest of their body excluding the head of which: **43** were virtual actors walking; **5,** virtual actors pointing; and **58,** virtual actors moving objects.

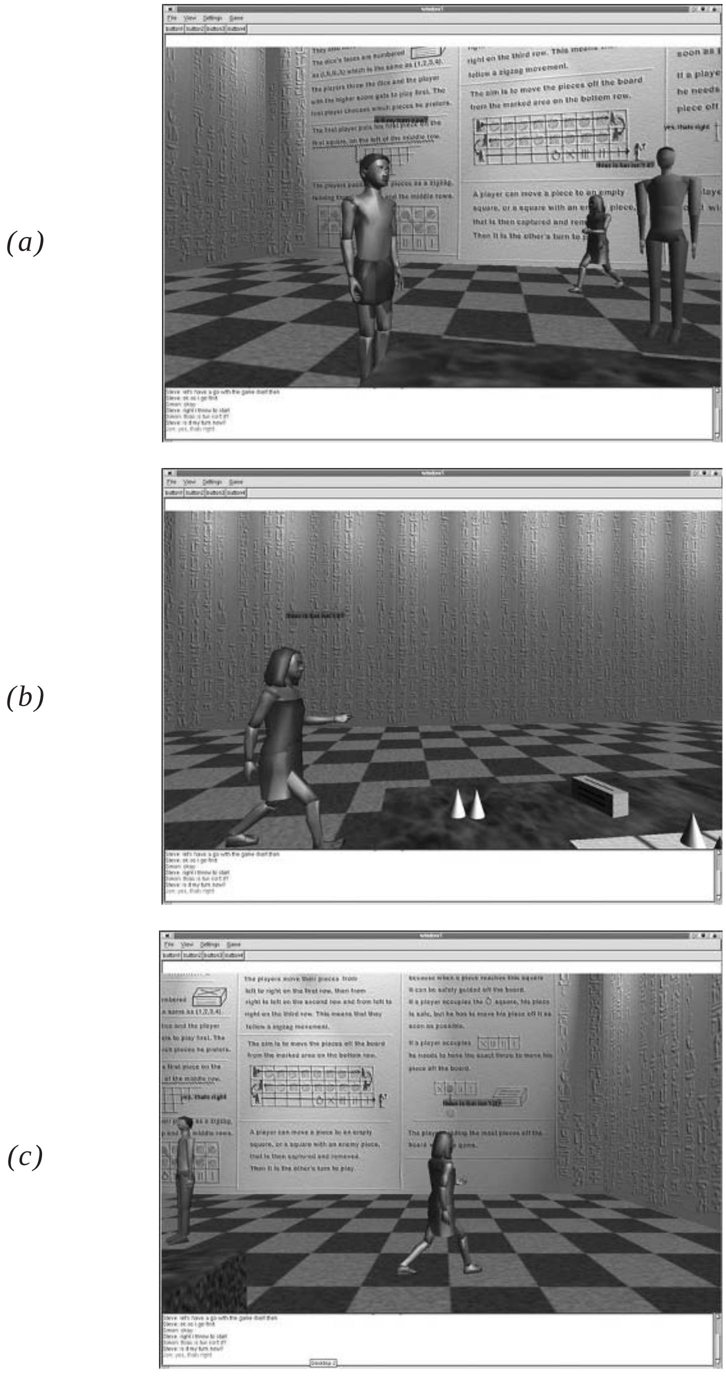
The grid provided a numerical record of actions and allowed different types of analytic categories to be studied comparatively. This process allows association between certain types of analytic categories to emerge. Association between virtual actors' looking, orientation, and walking activities indicated the users' *focus of attention* on the speaker, or a user performing an action (see Figure 5). It also indicated the users' *process of activity and intention to perform a certain action or claim a turn*. Virtual actors' walking and looking towards a particular direction or object indicated the users' process of reaching their target

Figure 5. The users' position, orientation, and distance from other objects and virtual actors in the CVE indicates the users' focus of attention (In this figure the expert (the adult figure) observes the children (the child figure) moving a piece. The differently shaded area approximates the expert's viewpoint.)



Steve: Simon, have you worked out the rules yet?
Simon: I think so - I remember they were a bit hard to understand from the wall, but I got the hang of it in the end
Steve: let's have a go with the game itself then
Steve: ok so i go first
Simon: okay
Steve: right i throw to start
Simon: this is fun isn't it?

Figure 6. (a) Shows that the girl child was in the process of going to read the rules, (b) shows the girl child on the way back to the board, and (c) shows that the girl child reached the board where she was intended to make a move playing the game



(see Figure 6) and their intention to perform an action using that object (e.g., children were approaching the board to make a move, or the wall to read the rules). In after-session discussion, children characteristically mentioned, *"We could tell what the other users were about to do by the position of their actor."* Virtual actors standing across from one another, to see and be visible to others, was a way of attracting attention to claim a turn. However, such visual cues were not explicit as children mentioned, *"She was getting around the board, so I wasn't sure if she wanted to make a move."* Also the virtual actors' action of turning and looking at other virtual actors after the completion of their turn indicated their *anticipation of response (offering a turn)*, as well as offering the user to continue. Meanwhile, virtual actors bumping into obstacles in the CVE indicated users *encountering difficulties* with navigation.

Although pointing was a frequent action external to the system communication, it was restricted to limited cases in the CVE. After-session discussion with the children revealed that this was due to lack of reactive visual clues related to their action. One child commented, *"If at least I could see my hand or something…it would be easier."* Instead children used moving objects to demonstrate something, so there was a spontaneous visual clue to their action (e.g., a piece being moved). In the limited cases pointing has been observed, it has been used mainly as a means of response (e.g., *"How did you move?… like that"*), or to direct other users to do something (e.g., *"Correct your move by moving your piece there…"*).

Figure 6(a) shows that the girl child was in the process of going to read the rules, Figure 6(b) shows the girl child on the way back to the board, and Figure 6(c) shows that the girl child reached the board where she was expected to make a move playing the game.

Moving objects involved actions such as rolling the dice, moving pieces to play the game, complementing an explanation with a demonstration, and correcting a move (e.g., the expert moved pieces on children's behalf to correct and support them; in extreme cases the expert even had to reset the board and restart the game). Apart from common cases where the players were moving their own pieces, there were cases where players moved pieces on behalf of their co-player to help them, corrected their move, and in some cases to cheat. The system did not provide any tools to prevent this from happening.

The virtual actors' action moving objects indicated:

- *the user in control*, by the laser from the virtual actor's hand to the object being moved;
- *the process of activity*, user's action moving pieces indicated his or her involvement in a certain activity like setting up the board, taking turn to play, demonstrating something, or correcting others;

- *the offering of a turn*, the completion of a move indicated the completion of the user's turn and thus indicated availability for turn taking; and
- *user comprehension*, the efficiency of children's moves were clear indicators of their understanding of the rules.

Communication Activity

Communicative activity analytic category examined the sessions down to a turn level and evaluated the efficiency of the communication tools supported by the third-phase prototype, such as: textual communication, text bubbles, and the transcript window; and multi-modal acts such as pointing, demonstrations, and user actions as a means of response.

The application of the grid to the example session for the communicative activity analytic category showed that multi-modal communication was very important. **Thirty-nine** out of **107** communication activities occurrences were textual based; **68** were multi-modal acts, of which **9** were pointing, **5** were demonstrations, and **54** were user actions as a means of response.

Presenting the user's communication as a text bubble above their virtual actor was beneficial in terms of: *focusing the users' viewpoint*, as a user had to turn to see someone else's text bubble, and thus follow the speaker's action; and *making the speaker explicit*, by relating the bubble to the speaking virtual actor.

The users relied on the transcript window to follow earlier communication, and in cases they could not see the speaking virtual actor. Text appearing in text bubbles and visual cues as small as text appearing in the transcript window played an important role in marking turn boundaries (this has already been demonstrated above).

One of the problems regarding the text bubbles was when the virtual actors were positioned close to each other, one bubble box obscured another. In such cases the users resorted to using the transcript window. Another problem was that the text remained in the text bubble until the next time the user typed something, which was confusing because it appeared as if the speaker was currently referring to something that was referred to earlier. This indicated the need of a means of making the bubble box disappear after a period of time.

The rules on the wall provided an important educational resource, as well as an important means of communicating information in the CVE. Based on information analysing this exemplar session for physical activity, **8** times children walked towards the wall to read the rules either after being directed by the expert or on their own initiative.

External Intervention

The external intervention analytic category is examined at the session or segment level and is concerned with issues that may cause a complete breakdown in a session. It investigated why the expert's support wasn't sufficient and additional external help (by the helper) was required to recover from or prevent an activity breakdown. The grid-based analysis helped in studying the cues that signified external intervention, the reason for the intervention, who adopted the role of the intervener, and what was the intervener's action (e.g., to recover from breakdown or prevent serious breakdown).

Twenty-six cases of external intervention were recorded in the example session. **Eight** were to **recover** and **18** to **prevent** activity breakdown. External interventions in the third phase have been minimised compared to the second phase (14% external interventions in the third phase versus 33% in the second phase). This means that the Deva Senet prototype system satisfactorily supported most of the user requirements.

The main cues that signified the necessity for external interventions were children encountering difficulties, missing ongoing activities being involved in an activity of less importance, interrupting each other (e.g., to prevent a child monopolising the activity and not allowing the other child to take a turn), requesting support, and requiring encouragement.

Pedagogy

The last analytic category of the grid is pedagogy. It studies simultaneously:

- who adopted the teacher's role: expert, co-player, helper;
- topic being covered;
- pedagogical tactics employed;
- change in tactic; and
- reason for the tactic being adopted.

A pedagogic style can only be followed across an entire segment, stage, or even session level. This is due to the fact that a pedagogical tactic can only be identified throughout several turns. The focus of the analysis was to evaluate the efficiency of the educational resources and the tools for practical management provided by the Deva prototype. This was achieved by identifying the pedagogical styles the system tools supported according to the contextual information to be delivered and the competence of the participants.

Following the expert's activity throughout a whole session, it appears that initially the expert provided a *model* for the children to follow (e.g., providing demonstrations, or moving on behalf of the children). Then the expert *coached* the children by involving them in the game playing activity, or asking them to comment on their co-player's move. As the children became familiar with the game, the expert *gradually removed support*. The children's teacher (who observed the activity) emphasised that the tactic the expert adopted in questioning the children was essential for encouraging the children's engagement and aid to the problem-solving nature of the activity. One of the most frequent tactics the expert used to correct the children was demonstrating possible moves they could have done, then asking them to correct their move. This tactic encouraged *articulation* and *reflection,* as the children started developing their own strategies in playing the game.

Analysis at the Session Level

This step is about investigating what the numerical values that derive from the grid at the turn or segment level reveal for the way activities develop throughout a whole session. The analyst needs to examine how and why the participant's behaviour changes over the whole session. The study of the flow of the segments in the whole session is based on the timeline (the turn index) provided by the grid. The numerical values and the grid descriptions provide a concise narrative of the session and allow the examination of its overall structure. This also allows different analytic categories to be compared. For example, comparing the physical activity and communication activity analytic categories might reveal cases in which pointing has been used to complement speech (also see the section, "Physical Activity," above).

The final stage is to compare different sessions' findings against each other in order to identify patterns of activities. This process derives *key points*, which are then translated to design guidelines. For example, a key point that derived by studying physical activity was *'the need of control over the session and individual users'*. This was based on the observation that the Deva prototype did not provide any controls to prevent users from rolling the dice more than once at a time, moving more than a piece at a time, or moving their co-players' pieces, highlighting the expert's inability to monitor and manage individuals and the situation. It also highlighted the need to provide users with certain roles with various controls over object manipulation, the situation, and other users.

The analysis at a session level might identify problems with the grid itself, in terms of the analytic categories the grid deals with, or ones that need to be added to the grid and investigated further.

Derivation of Design Guidelines

The final stage of the method translates key points deriving for each analytic category of the grid into design guidelines. The findings from all sessions and for all the analytic categories of the grid are considered.

Design guidelines need to be precise. Providing guidelines with extra information and examples reduces the chances of the guideline being too vague or conflicting (Reisner, 1987). The method follows a model of reporting design guidelines for usability in CVEs which is determined by four parts:

- *Design Guideline (DG)*, which reports the DG that needs to be incorporated.
- *Motivation*, which argues the importance of the DG based on the phases' results.
- *Benefit*, which discusses how the application of a DG addresses the issues that drew the creation of the DG itself (depending on context, it is possible that some DGs may have a negative force in the CVE; this can be addressed with the evaluation of the DGs, which may address the need for the derivation of other DGs to overcome such problems).
- *Examples*, one or two of the practical implementation of the DG.

This method is based on Kaur's method of reporting design guidelines for usability in VEs (Kaur, 1998).

Forty DGs were derived from the application of the grid based analysis to the third phase of this study. These are outlined in the Table 5. The context of their use is learning environments. They are related to the following aspects of CVEs for learning:

- *Environment*, which address issues related to general tools CVEs for learning should provide.
- *Objects*, which address issues regarding the objects' features contained in CVEs for learning.
- *Virtual Actors*, which address issues regarding the virtual actors' features in CVEs for learning.
- *Virtual Actor Behaviour*, which address issues related to the behaviours virtual actors with different roles in CVEs for learning should incorporate.

Table 5. List of design guidelines based on the third-phase user studies

Design guidelines (DG)	
Environment	
DG1:	users need to have simultaneous control over certain types of activities but not on others
DG2:	a history of the communication activity needs to be recorded
DG3:	a history of the physical activity needs to be recorded and replayable
DG4:	a permanent information resource related to the educational activity should be available in the CVE and should always be visible to all the users
DG5:	the permanent information resource should incorporate multimedia display techniques
DG6:	audio communication should be supported
Objects	
DG7:	key objects in the CVE should incorporate intelligence
Virtual actors	
DG8:	the virtual actors should be aesthetically pleasing
DG9:	the virtual actor should provide the user with a unique representation
DG10:	a virtual actor should convey the user’s role in the CVE (e.g. child, expert)
DG11:	a virtual actor should convey the user’s viewpoint
DG12:	a virtual actor should reveal the user’s actionpoint
DG13:	a tool should be provided for users to lock onto the active virtual actor and follow it automatically
DG14:	users need to be provided with real-time cues about their own actions
DG15:	a virtual actor should be easily associated with its communication
DG16:	text bubbles should not overlap
DG17:	a text bubble should remain for a short period of time after the end of an utterance
DG18:	a virtual actor should convey the user’s communication as it is being composed
DG19:	a virtual actor should reveal the sequence of the dialogue exchange
DG20:	a virtual actor should convey explicitly the user’s process of activity and state-of-mind
DG21:	a virtual actor should convey the user’s intention to take a turn
DG22:	a virtual actor should convey the user’s offering of a turn
DG23:	an active participant needs to be identified even when their virtual actor is out of other users’ viewpoints
DG24:	users’ viewpoints should be easily directed to see an active participant even when they are out of other users’ viewpoints
DG25:	the speaker needs to be identified even when their virtual actor is out of other users’ viewpoints
DG26:	a user’s viewpoint should be easily directed to see the speaker even when the speaker is out of other users’ viewpoints
DG27:	a virtual actor should convey the user’s intention to take a turn even when not being in other users’ viewpoints
DG28:	the virtual actor should convey the user’s offering of a turn even when being out of other users’ viewpoints
DG29:	private communication should be supported
DG30:	private channels of interaction should be supported
DG31:	a virtual actor should show when the user is involved in private communication and whether or not others could join in
DG32:	a virtual actor should show when the user is involved in private interaction and whether or not others could join in
Virtual actors’ behaviour	
Student virtual actor	
DG33:	a student virtual actor should have a basic customisable behaviour
Teacher virtual actors	
DG34:	the teacher should be in control of the children’s behaviour
DG35:	the teacher should have control over an individual user’s viewpoint
DG36:	the teacher should be able to take control of objects in the CVE
DG37:	the teacher should be in control of the communication tools
DG38:	the teacher should be aware of and have control over private communication between children
DG39:	the teacher should be aware of and have control over private interactions between children
DG40:	the teacher should have an episodic memory of children’s mistakes

Virtual actor behaviour includes behaviours for two categories of users:

- *The Student*, the naïve users, who did not know the rules of the game and were new to the experience of participating in a CVE application.
- *The Teacher*, the knowledgeable users in the CVE, who did not play, but knew the rules of the game, was aware of the process to be followed, and whose duty was to assist the children and provide guidance and support.

Future Trends

There are various directions for further work that span the following areas:

- growth and generalisation of the design guidelines
- development of tools for designers
- automation of the seven-step grid-based method

Growth and Generalisation of the Design Guidelines

A natural direction of the research would be the further development of the currently developed design guidelines by increasing the CVE population. Another direction is the generalisation of the design guidelines to a wider area of applications. This could be achieved by applying the design guidelines to application domains outside education and evaluating their usefulness based on user studies. This process could also aid the further growth of the current list of design guidelines.

Develop Tools for Designers

Another natural step for further development is:

- the provision of a tool for presenting design guidelines to CVE designers in an easily accessible and comprehensive way; and
- the development of a system that provides tools that facilitate implementation of the proposed design guidelines.

The use of hypertext or multimedia technology could be used for the development of a design guidelines presentation tool for CVE designers. The use of a hypertext tool for providing guidance to VE designers (Kaur, 1998) received a good usability score from the VE designers and had a positive impact on the design process.

Achieving the second requires the development of CVE systems that provide the underlying technology that allows the implementation of the recommended design guidelines. They should also provide an interface easy to use by CVE designers or artists—people who do not necessarily need to have engineering knowledge to easily implement the recommended design guidelines.

Automation of the Seven-Step Grid-Based Method

Another important extension of the research reported in this chapter is the automation of the seven-step grid-based method for conducting video analysis and automatically generating task models, behavioural patterns, and statistical information from rich qualitative data.

This could be achieved by providing analysts with a non-linear tool that allows:

- the representation of a list of analytic categories and analytic themes (events, tasks, and actions) involved in each analytic category and a display of their relationships—this work could be an extension of the work of Luckin et al. (1998), who developed a tool for tracking interactivity in multimedia environments;
- direct annotation of the videotape (this skips the tedious and time-consuming transcription process and allows direct chunking of the session) and creation of link between the observed actions and analytic categories or tasks; and/or
- graphical representation of actions against a timeline and their association with appropriate tasks.

Conclusions

This chapter proposed a method that provides rigour in the study of social interaction in CVEs, to determine requirements for CVE systems design and inform the CVE systems design. The method provides a means of managing a large amount of disparate data (two videotapes, audio, notes, text files). The proposed grid-based analysis provides a means of obtaining a more concise and objective measure of the moment-to-moment details occurring in a session. The chunking, indexing, and use of the timeline provides means of looking at the overall structure and flows within a session. The outcome of this analysis is a set of design guidelines that can inform the construction of CVEs.

There are some drawbacks with the method. Firstly, it is not as exhaustive or does not generate as much rich, qualitative informative as, for example, ethnographic techniques. However, the primary purpose of the method is to gather requirements within certain restrictions of time and resources. Secondly, the grid is more suitable for analysing at the turn level rather than for broader issues such as pedagogy, where a whole segment needs to be analysed to reflect the use of different pedagogic tactics in a stage. Further development is needed to the method to address this issue. Thirdly, the factors to be studied must be reasonably clear in order to derive analytic categories for the grid. This means that in the early stages of research, some exploratory studies are needed (as was the case in the first-phase studies in the Senet project).

The method developed shares many of the analytic foci, which defines Interaction Analysis. It builds on this, by means of the grid, to provide a more efficient and rigorous requirements gathering technique. Its application to the third phase of the work showed how the method can be repeated and extended to form an evaluation method for CVEs.

Acknowledgments

Thanks to the State Scholarships Foundation of Greece for funding Daphne Economou's PhD, the MMU Manchester Multimedia Centre for use of their facilities, and Claremont Road Primary School and Knutsford High School for their cooperation.

References

Allen, C. (1989). The use of video in organizational studies. *ACM SIHCHI Bulletin: Special Edition on Video as a Research and Design Tool, 21*(2), 115-117.

Atkinson, J.M., & Heritage, J. (eds.). (1984). *Structures of social action: Studies in conversation analysis*. Cambridge: University Press.

Bellotti, V., & Rogers, Y. (1997). From Web press to Web pressure: Multimedia representations and multimedia publishing. *Proceedings of the ACM Conference on Human Factors in Computing (CHI'97)* (pp. 279-286). New York: ACM.

Benford, S., Bowers, J., Fahèn, L., Greenhalgh, C., & Snowdon, D. (1995). User embodiment in collaborative virtual environments. *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems* (pp. 242-249). New York: ACM.

Benford, S., Greenhalgh, C., Rodden, T., & Pycock, J. (2001) Collaborative virtual environments. *Communications of the ACM*, *44*(7), 79-85.

Boden, D., & Zimmerman, D.H. (1991). *Talk and social structure: Studies in ethnomethodology and conversation analysis*. Cambridge: Polity.

Bowers, J., & Martin, D. (1999). Informing collaborative information visualisation through an ethnography of ambulance control. *Proceedings of the 6th European Conference on Computer Supported Cooperative Work (ECSCW'99)* (pp. 311-330). Kluwer Academic Publishers.

Bullock, A., Simsarian, K.T., Stenius, M., Hansoon, P., Wallberg, A., Åkesson, K., Frécon, A., Ståhl, O., Nord, B., & Fahlén, L.E. (2001). Designing interactive collaborative environments. In E.F. Churchill, D.N. Snowdon, & A.J. Munro (Eds.), *Collaborative virtual environments, digital places and spaces for interaction* (pp. 179-201). Springer-Verlag.

Capin, T.K., Pandizc, I.S., Chauvineau, E., Thalmann, N.M., & Thalmann, D. (1996). *Modeling and animation of virtual humans*. Public Report Number D4.1, AC040-GEN-MIR-DS-P-041.b0.

Coulthard, M., Montgomery. M., & Brazil, D. (1981). Developing a description of spoken discourse. In M. Coulthard & M. Montgomery (Eds.), *Studies in discourse analysis* (pp. 1-50). London: Routledge.

DfEE. (1997). *Connecting the learning society: The government's consultation paper on the national grid for learning*. UK: Department of Education and Employment.

DfEE. (1998). *Open for learning, open for business: The government's national grid for learning challenge*. UK: Department of Education and Employment.

Durlach, N., & Mavors, A.S. (1994). *Virtual reality: Scientific and technological challenges*. Washington, DC: National Academy Press.

Economou, D., Mitchell, W.L., & Boyle, T. (2000). Requirements elicitation for virtual actors in collaborative learning environments. *Computers & Education, 34*(3-4), 225-239.

Economou, D., Mitchell, W.L., Pettifer, S.R., Cook, J., & Marsh, J. (2001). User-centred virtual actor technology. *Proceedings of Virtual Reality, Archaeology and Cultural Heritage (VAST'2001)* (pp. 323-332). New York: ACM.

Engeström, Y., & Middleton, D. (eds.). (1996). *Cognition and communication at work*. Cambridge, UK: CUP.

Glaser, B.G., & Strauss, A. (1967). *The discovery of grounded theory: Strategies for qualitative research*. Chicago: Aldine.

Gunton, T. (Ed.). (1993) *Information systems practice: The complete guide*. Manchester: NCC Blackwell.

Harper, R. (1997). Gatherers of information: The mission process at the International Monetary Fund. *Proceedings of the 5th European Conference on Computer-Supported Cooperative Work (ECSCW'97)* (pp. 361-376). Kluwer Academic Publishers.

Heath, C., & Luff, P. (1991). Collaborative activity and technological design: Task coordination in London Underground control room. *Proceedings of the 2nd European Conference on Computer Supported Cooperative Work (ECSCW'91)* (pp. 65-80). Kluwer Academic Publishers.

Heath, C. & Luff, P. (1996). Convergent Activities: line control and passenger information on the London Underground. In Y. Engeström & D. Middleton (Eds.), *Cognition and communication at work* (pp. 96-129). New York: Cambridge University Press.

Hutchins, E.L. (1990). The technology of team navigation. In R.E. Kraut, J. Galegher, & C. Egido (Eds.), *Intellectual teamwork: The social and technological foundation of cooperative work* (pp. 191-221). Hillsdale, NJ: Lawrence Erlbaum.

Hutchins, E., & Klausen, T. (1996). Distributed cognition in an airline cockpit. In D. Middleton & Y. Engeström (Eds.), *Communication and cognition at work* (pp. 15-34). Cambridge: Cambridge University Press.

Jordan, B., & Henderson, A. (1995). Interaction Analysis: Foundations and practice. *The Journal of Learning Sciences, 4*(1), 39-103.

Kaur, K. (1997). Designing virtual environments for usability. *Proceedings of the Human-Computer Interaction (INTERACT'97)* (pp. 636-639). UK: Chapman & Hall.

Kaur, K. (1998). *Designing virtual environments for usability*. PhD thesis, Centre for HCI Design, City University, London.

Kaur Deol, K.K., Steed, A., Hand, C., Istance, H., & Tromp, J. (2000a). Usability evaluation for virtual environments: Methods, results and future directions (part 1). *Interfaces, 43,* 4-8.

Kaur Deol, K.K., Steed, A., Hand, C., Istance, H., & Tromp, J. (2000b) .Usability evaluation for virtual environments: Methods, results and future directions (part 2). *Interfaces, 44,* 4-7.

Luckin, R., Plowman, L., Gjedde, L., Laurillard, D., Stratfold, M., & Taylor, J. (1998) An evaluator's toolkit for tracking interactivity and learning. In M. Oliver (Ed.), *Innovation in the evaluation of learning technology* (pp. 42-64). London: University of North London.

Luff, P., Hindmarch, J., & Heath, C. (eds.). (2000). *Workplace studies, recovering work practice and informing system design*. Cambridge Press.

Mills, S., & Noyes, J. (1999) Virtual reality: An overview of user-related design issues. *Interacting with Computers*, *11*(4), 375-386.

Mitchell W.L. (1999) Moving the museum onto the Internet: The use of virtual environments in education about ancient Egypt. In J.A. Vince & R.A. Earnshaw (Eds.), *Virtual worlds on the Internet* (pp. 263-278). IEEE Computer Society Press.

Monk, A., Wright, P., Haber, J., & Davenport, L. (1993). *Improving your human computer interface: A practical approach*. Hemel Hempstead: Prentice-Hall International.

Moran, T.P., & Anderson, R.J. (1990). The workaday world as a paradigm for CSCW design. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work CSCW'90* (pp. 381-393). New York: ACM Press.

Neal, L. (1989). The use of video in empirical research. *ACM SIGCHI Bulletin: Special Edition on Video as a Research and Design Tool, 21*(2), 100-101.

Newman, W.M., & Lamming, M.G. (1995). *Interactive system design.* Addison-Wesley.

Pettifer, S., & West, A. (1999). *Deva: An operating environment for future large-scale virtual reality.* Department of Computer Science, University of Manchester, UK.

Reisner, P. (1987). Discussion: HCI, what is it and what research is needed? In J.M. Carroll (Ed.), *Interfacing thought: Cognitive aspects of human-computer interaction* (pp. 337-352). Cambridge, MA: MIT Press.

Roussos, M., Johnson, A., Moher, T., Leigh, J., Vasilakis, C., & Barnes, C. (1999). Learning and building together in an immersive virtual world. *Presence: Teleoperators and Virtual Environments*, *8*(3), 247-263.

Silverman, D. (Ed.). (1997). *Qualitative research: Theory, method and practice*. London: Sage.

Silverman, D. (Ed.). (2000). *Doing qualitative research: A practical handbook*. London: Sage.

Snowdon, D., Churchill, E.F., & Munro, A.J. (2001). Collaborative virtual environments: Digital spaces and places for CSCW: An introduction. In E.F. Churchill, D.N. Snowdon, & A.J. Munro (Eds.), *Collaborative virtual environments, digital places and spaces for interaction* (pp. 3-17). Springer-Verlag.

Soloway, E., Jackson, S.L., Klein, J., Quintana, C., Reed, J., Spitulnik, J., Stratford, S.J., Studer, S., Eng, J., & Scala, N. (1996). Learning theory in practice: Case studies of learner-centered design. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96)* (pp. 189-196). New York: ACM.

Stanney, K.M., Mourant, R.R., & Kennedy, R.S. (1998). Human factors issues in virtual environments: A review of the literature. *Presence, Teleoperators and Virtual Environments*, *7*(3), 327-351.

Steed, A., & Tromp, J. (1998). Experiences with the evaluation of CVE applications. *Proceedings of CVE'98* (pp. 123-130).

Stewart, J., Bederson, B.B., & Druin, A. (1999). Single display groupware: A model for cooperative collaboration. *Proceedings of the ACM Conference on Human Factors in Computing Systems: The CHI is the Limit (CHI'99)* (pp. 286-293). New York: ACM.

Suchman, L. (1996). Constituting shared workspaces. In Y. Engeström, & D. Middleton (Eds.), *Cognition and communication at work* (pp. 35-60). Cambridge: Cambridge University Press.

Tromp, J. (Ed.). (1999). *Usability design for CVEs*. Public Report Number D2.9, A040-NOT-CRG-DS-P-029-02.

Viller, S., & Sommerville, I. (1999) Social analysis in the requirements engineering process: From ethnography to method. *Proceedings of the International Symposium on Requirements Engineering (RE'99)* (pp. 6-13). IEEE Computer Society Press.

Chapter XI

A Component-Oriented Approach for Mixed Reality Applications

Michael Haller
Upper Austria University of Applied Sciences, Austria

Abstract

This chapter introduces a component-oriented approach for developing mixed reality (MR) applications. After a short definition of mixed reality, we present two possible solutions for a component-oriented framework. Both solutions have been implemented in two different MR projects (SAVE and AMIRE). The first project, SAVE, is a safety training system for virtual environments, whereas the goal of the AMIRE project is to develop different authoring tools for mixed reality applications. A component-oriented solution allows developers to implement better designed MR applications, and it fosters the reusability of existing MR software solutions (often called MR gems). Finally, it supports the implementation of adequate visual authoring tools that help end users to develop their own MR applications with no programming skills.

Introduction

The often underestimated complexity of mixed reality (MR) applications necessitates efficient application design. Rapid prototyping of mixed reality applications is mostly impossible, because of two reasons: Firstly, most of the existing frameworks are in many cases too complex to be extended, and secondly, it needs a lot of software development skills and interface programming knowledge to develop well-designed MR software. In this chapter we want to present a component-oriented approach for developing MR applications. The goal of this approach is to support developers during their development of new MR applications. Having a component-oriented framework makes the programming life easy, because the developers do not have to reinvent everything from scratch. Based on this approach corresponding authoring tools will support end users to develop their own MR applications without having programming skills. After a short overview in the taxonomy of mixed reality and virtual environments, we present the requirements for such applications. We then describe related work in this field and present a general component-oriented approach, followed by description of two showcases, where our approach has successfully been implemented. Both applications are based on the component-oriented approach and result in a generic and flexible system. Finally, we describe future trends, including the implementation of nice MR authoring tools for end users with no programming skills. We conclude this chapter with a short summary.

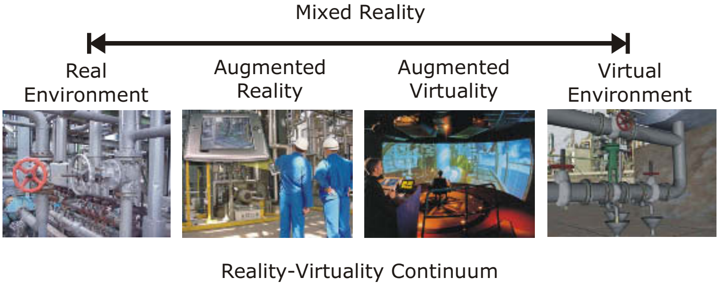
Background

In the following section we will give a short overview of mixed reality describing the requirements for the setup of a MR application and presenting the related work in this field.

From the Virtual Environment to the Real Environment

Mixed reality (MR) is a particular superset of augmented reality (AR) technology that involves the merging of real and virtual worlds somewhere along the reality-virtuality continuum, which connects completely real to completely virtual environments. The terminology was first introduced by Milgram and Kishino (1994) and is depicted in Figure 1. MR technology has been exploited in the

Figure 1. From the real environment to the virtual environment



medical, military, and entertainment fields (Azuma, 1997); more and more new fields such as industry and training are becoming interested in its possibilities. The use of mixed reality enhances users' perceptions and the interaction with the real world (Azuma et al., 2001). Virtual objects display information that the users cannot directly detect with their senses. In addition, this information conveyed by the virtual objects helps a user perform real-world tasks.

Virtual environments and virtual reality-based applications can become very complex. Even more complex than present VR systems are the VR tools for modeling these environments (Bimber, Fröhlich, Schmalstieg, & Encarnação, 2001). Often, VR applications and the corresponding authoring tools are not easily extendible, and the authors of VR environments require a lot of programming knowledge for realizing the desired virtual scene (Milgram & Kishino, 1994). On the one hand, developers should have programming skills; on the other hand, they should have an exact technical knowledge about the composition of the scene. Due to the different hardware devices (input and output devices), the framework has to be extendable, open for new devices, and finally it should be easy to use for programmers. Figure 2 depicts the most important components of a virtual environment: The user is the central part of the system, followed by the input and output devices that present the environment to the user and finally the simulation itself that renders the virtual environment.

Normally, virtual reality systems work on powerful graphics machines (Figure 3). The visual representation is mostly displayed to the user through an HMD device that allows an immersive feeling, combined with a positional tracking device that reports the user's head and body position to the system. With this input, the simulation can generate a first-person view. Especially in VR simulation, external trainers often supervise the training session. In this case, an external application often communicates with the simulation (e.g., via TCP/IP or UDP).

Figure 2. The most important components of a virtual environment application

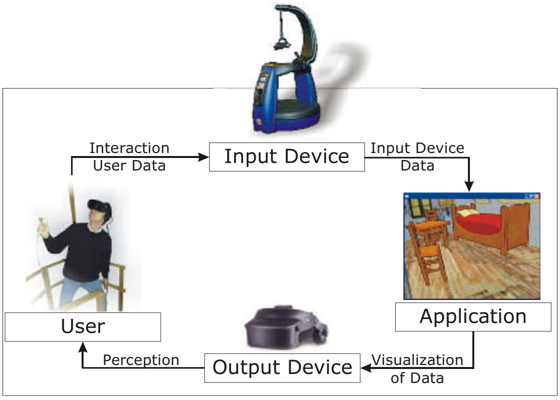
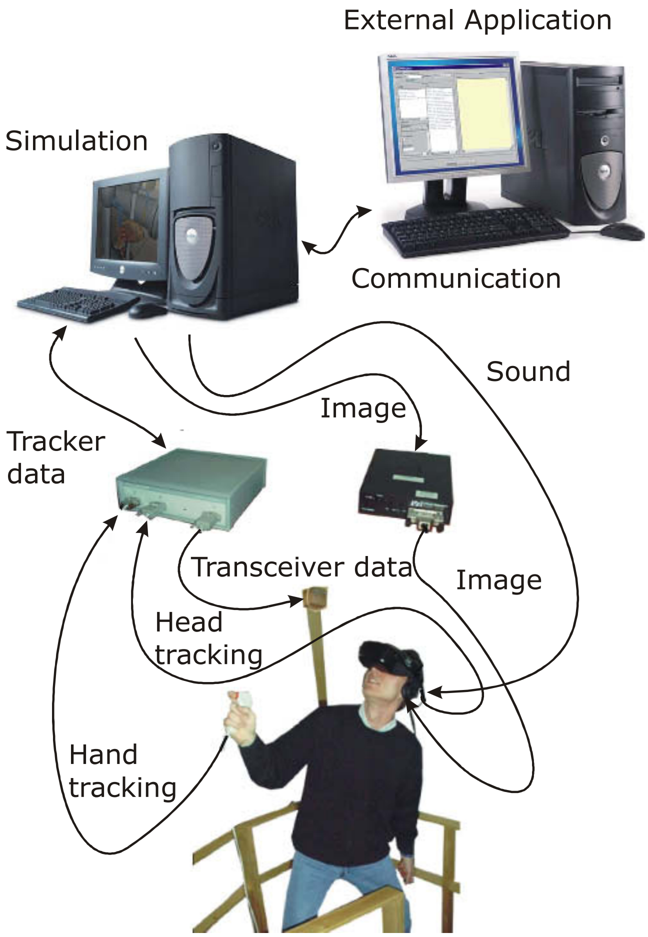


Figure 3. A typical topology of a virtual reality application



Requirements for a Mixed Reality Component Architecture

The component-oriented approach for the implementation of mixed reality applications is characterized mainly by the following facts:

- It should allow third-party development.
- The framework should hide programming code.
- Components can be implemented by using other components.
- In most cases, the component-oriented approach offers authoring tools that allow for an easy development of new components.

As mentioned in Dachselt (2001), we have to distinguish between technical and authoring requirements. From the technical point of view, the component-oriented approach should provide:

- **Portability:** Independently from an underlying renderer, each component should be removable, and by doing so, it should not influence the system in a negative way.
- **Distribution:** Components should not only work on one system. A distributed application should also be supported.
- **Adaptation:** Each component can be modified by the user's preferences. This should be possible with minimal effort.
- **Performance:** It is one of the key factors for a mixed reality application to work in real time. Long delays between user inputs and the system output cause discomfort or simulator sickness, and it negatively affects the user's feeling of presence.

From the authoring point of view, there are the following additional requirements:

- A **clear interface** for the components—This includes the data and the component definition.
- Well-written **documentation** and **description** of the components.
- Support of adequate **authoring tools** that allow a rapid prototyping for authoring users who do not have programming backgrounds.
- **Easy configuration** of the properties for a component.
- **Persistence** of components.

- Loading of **additional components** without a modification of the underlying framework.

Dörner and Grimm (2001) define the following component features:

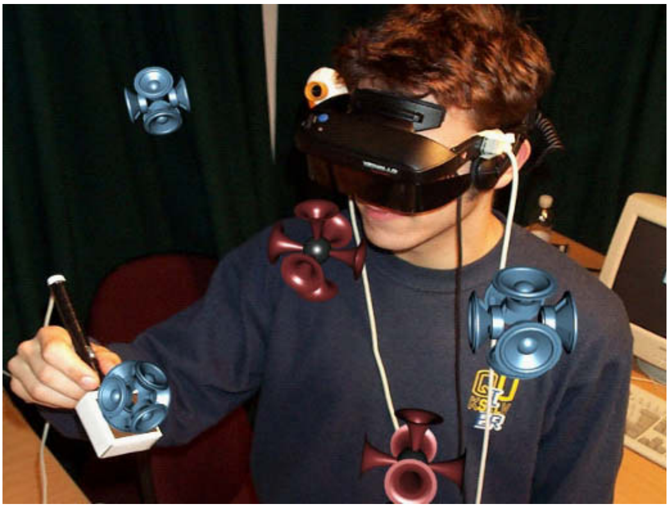
- **Customization:** Each component needs to be able to be modified by an author. Not only programmers should be able to customize components, but also authors and experts by using authoring tools.
- **Persistence:** Components and their state should be stored in a way that they can be reloaded. This should even include the distribution of components over a network.
- **Reflection:** Functionality that allows information to be retrieved from a component, such as the events that are sent to another component.
- **Event-based communication:** Components have to communicate with others—in most cases, this is an event-driven communication in which e-messages are sent from a so-called event source to all entities that are registered to listen to the particular events. The communication messages are mostly forwarded by so-called slots, or pins.

Related Mixed Reality Frameworks

Many European projects mainly focus on the development of MR applications for a special domain (e.g., technical maintenance), such as ARVIKA (Friedrich, 2000), Studierstube (Schmalstieg, Fuhrmann, Szalavari, & Gervautz, 1996), and DWARF (Bauer et al., 2001). Unfortunately, only MR experts are able to develop MR/AR applications. Prototyping of MR-based applications becomes a very difficult task, because most research institutes have to develop these applications starting from scratch.

With the use of the ARToolKit library (Kato, Billinghurst, Blanding, & May, 1999), the development of MR/AR applications became easier and more popular. Figure 4 shows an AR example based on ARToolKit, in which the users can place 3D sound sources into the real world (Haller, Dobler, & Stampfl, 2002; Dobler, Haller, & Stampfl, 2002). The ARToolKit library is free for non-commercial projects and easy to integrate into an OpenGL environment. In addition, it offers the possibility to recognize objects without the usage of high-end tracking systems. Most current MR applications are isolated, and each institute implements its own framework. The situation is similar to the first steps in game development, where everyone was programming his/her own game-engine. The development of mixed reality software is like game development: In

Figure 4. In the ASR (augmented sound reality), the user is able to place virtual 3D sound sources into the real world



general it needs to be very flexible, because things change and new requirements and features need to be added to stay competitive. A source that can easily be adapted to change is a must and worthy goal.

Most current AR/MR applications are based on a self-made framework or/and at least on a scene graph library (e.g., Open Inventor, OpenGL Performer, Open Scene graph, or OpenSG). Of course, most of the AR/MR frameworks like Studierstube (Schmalstieg et al., 1996), DWARF (Bauer et al., 2001), and Tinmith-evo5 (Piekarski & Thomas, 2003) are object-oriented frameworks, but not all of them have a component-oriented approach or there is a lack of corresponding visual authoring tools (e.g., www.studierstube.org or www.augmentedreality.de). Scripting support (see www.studierstube.org/april/ and http://www.cc.gatech.edu/projects/ael/projects/dart.html) would help experienced end users, but it seems to be too complex for end users with absolutely no programming skills.

Component-Oriented Design: Overview

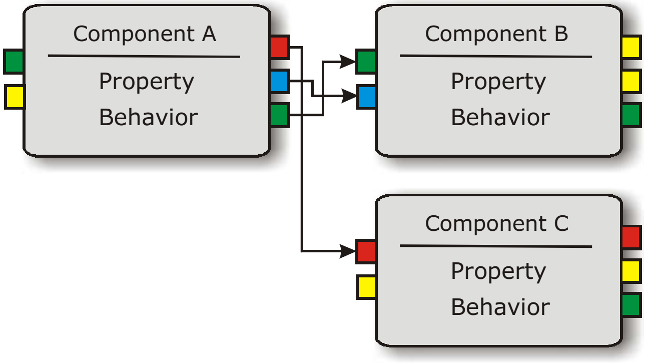
The following section presents a component-oriented design for developing MR applications, describing the main features of the design approach. Moreover, it depicts the component-oriented design workflow, including a short overview of the different rules.

The Component-Oriented Design for an MR Application

Geiger, Reimann, and Rosenbach (2000) define a component as a separated entity with a specific size. It is characterized by dependencies, and the framework permits a dynamic loading of components. Components can be either large or small, but they have to be of a clear structure. In our sense, a component-oriented MR system should be comparable to a set of different small LEGO™ components, which can be connected. In our view, a visible component has its geometry and a property. Moreover, it is characterized by a behavior. Components can be very simple, but they can also be composed of other simple components. They are often called *composed components* or *compound components*. Components hide their internal operations, and programmers do not need to understand the internal complexity.

Each component is composed of in- and out-slots, which can either send data to another component or receive data from the previous component. Consequently, out-slots can be connected to in-slots. This concept is illustrated in Figure 5. We

Figure 5. Components can be connected with in-slots and out-slots (These slots have to be of the same type.)

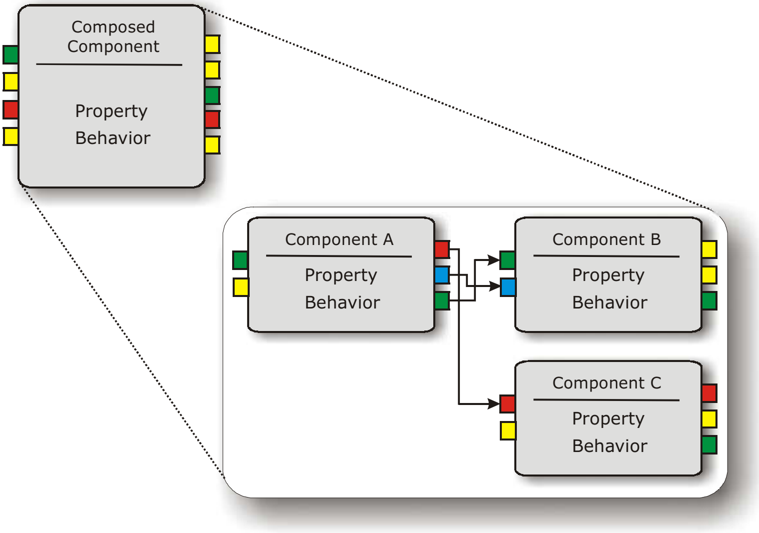


use typing for these slots to specify the receiver and the emitter data. By comparing the type of in- and out-slots, we can decide which slots are compatible for connection. Using slots for inter-object communication is not new in a component-oriented approach. Our communication concept is based on prototypes for components. This means the component developer registers the component via a component manager.

By routing events from out-slots to in-slots of another node, customized functions and dependencies can be implemented, e.g., if a switch has been switched on, a lamp shines, and so forth. In the SAVE-system, all the components are described by prototype nodes based on VRML 97 with their in-slots, out-slots, and parameters, which allow for a closer description of the object (Haller, 2001; Haller et al., 2002).

Of course, components can be composed of simpler components with fewer properties. In Figure 6 we have an example in which not all slots of the components A, B, and C are routed to the outside of the composed component—the composed component has fewer slots. Once a component has generated an event, the event is propagated from the out-slot along any route to other nodes. Event notifications are propagated from sources to listeners by the corresponding method invocations on the target listener objects. Data enters via the first component, passes along to the second component, and so on. The architecture

Figure 6. Different components can be grouped into composed components (This approach is important for authoring components. Otherwise, the visualization can be too disorganized and unmanageable.)



has an event-driven design, and components only start information processing after receiving an event. After processing, the components can generate new events, and the processing can be done in parallel. The system is based on the producer/consumer concept. No component (consumer) starts sending messages and processing information without having received a message from other nodes (producer). The only components that generate messages are called *active components* (e.g., timer, clock). They are triggered by system calls, but they do not start sending messages themselves.

There is no doubt that appropriate authoring tools would greatly assist users. Therefore, visual-based authoring tools are closely connected to the component-oriented approach. Normally, end users get an authoring tool with a set of predefined small components. These components can be modified and end users can tweak their properties. If the authors are more experienced, they can build their own components by modifying the WRL/XML files; indeed, programmers can add more features by programming new components. The structure of the component is described in a WRL/XML file that allows end users with no programming skills to tweak their components according to their requirements.

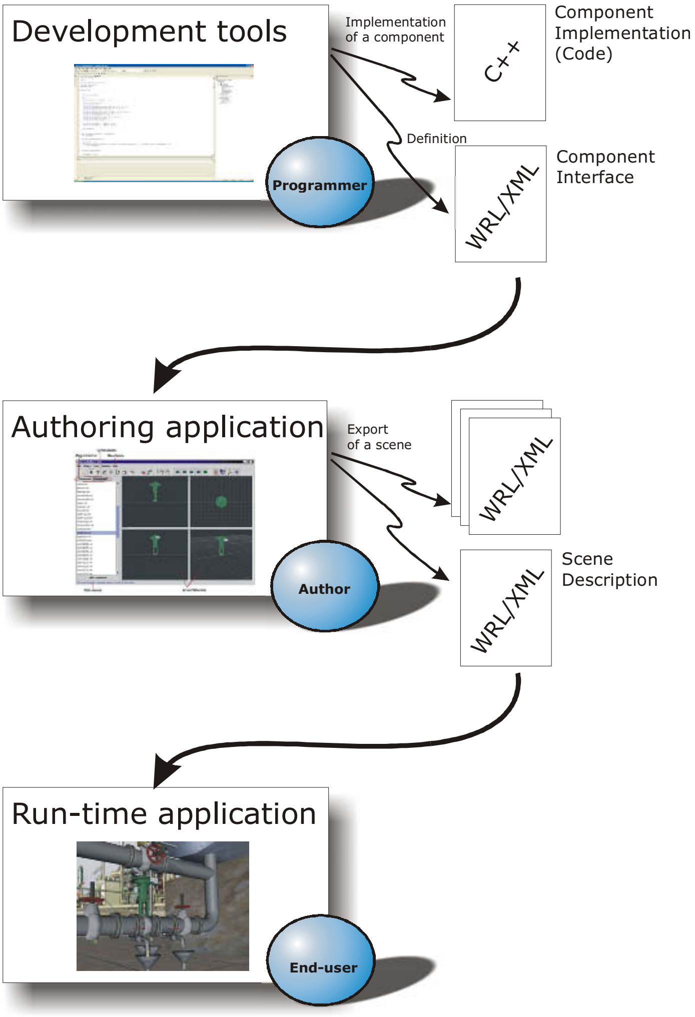
The Component-Oriented Workflow

The component-oriented workflow is depicted in Figure 7, which shows the three different categories of persons.

The **programmers** are responsible for the creation, implementation, and definition of components. In this case the programmers are using a conventional programming environment. The component interface can be described in an XML-form. In SAVE we used so-called PROTO for the interface description. In fact, these PROTOs have been based on the VRML prototypes (www.web3d.org/x3d/spec/vrml/vrml97/). XML has become more popular as of late because of two reasons: Firstly, there are a number of open-source XML parsers that help developers to parse the XML files. Secondly, the XML structure is flexible, extendable, and easy to use.

The **authors** are using so-called authoring tools for the implementation of the application. The underlying framework is responsible for the creation of components and the connection between them. Normally, authors are not programming experts—they are experts in their field and normally they know what the end users want to have. But in the normal case, they don’t know how to implement due to their lack of programming skills. The results of the author are the scenarios. They include the graphical elements, the objects of a mixed reality scenario, the property settings, the behavior of the components, and finally the logical connection between the components.

Figure 7. The component-oriented workflow



Finally, the **end users** are the persons that are involved in the virtual environment. They are not concerned with programming, modeling, and authoring.

Component-Oriented Design: Two Showcases

The following two showcases (SAVE and AMIRE) are based on the component-oriented framework. First, we present the SAVE system, followed by a closer description of its component-oriented architecture. Second, we demonstrate the AMIRE project, including a presentation of the component-oriented approach.

SAVE (Safety Virtual Environment)

In 1997, we started implementing a VR-based training simulation for an oil refinery. The first prototypes of SAVE were quite simple and showed only a small part of the refinery (www.faw.uni-linz.ac.at/save). In 1999, we had to add new training scenarios and new features and functionality (cf., Figures 8 and 9). What we wanted to have was a framework in which anyone could develop a VR-based application with a high degree of complexity. Moreover, we wanted to have an application design with high reusability, for which parts could be reused in different VR training scenarios. We knew that effective reuse requires more than just reusable code and libraries. Our vision was a transition from library-based reuse to kit-based reuse, and we wanted to move away from the traditional development of VR environments, for which the users require programming skills. The component-oriented approach with the included authoring tools show how modeling rather than programming can be used to realize virtual environments.

Figure 8. SAVE is a VR-based training program for oil refinery employees

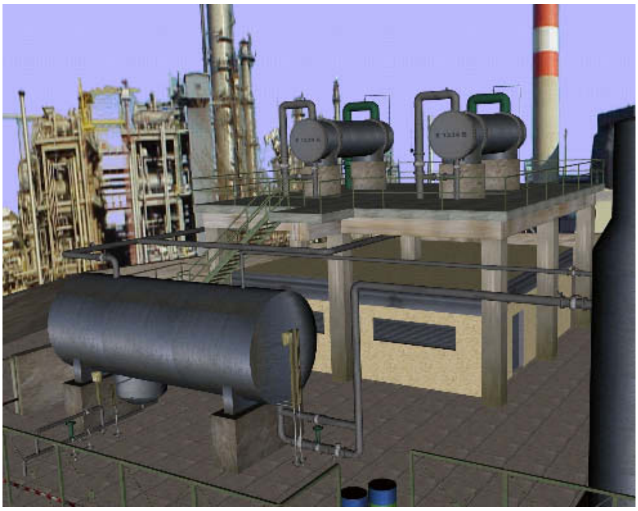
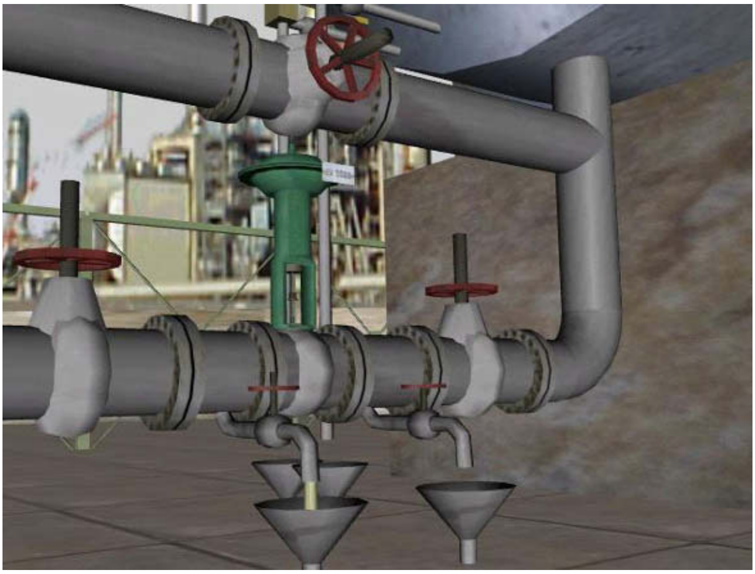


Figure 9. A typical scenario that has to be built-in for the virtual plant simulation (In fact, the dependencies between the components and the animation of the components itself are not so complex.)



The Component-Oriented Design of SAVE

In SAVE, we wanted to have a component-oriented design that is not comparable to the commercial component models such as JavaBeans or COM. Our system is comparable to a set of different small LEGO™ components, which can be connected. In our view, a visible component has its geometry and its properties. Moreover, it is characterized by a behavior. Our goal was to offer a repository with different simple and even complex objects. All the components are described by prototype nodes based on VRML 97, describing their in-slots, out-slots, and parameters, which provide a more detailed description of the component (cf., Figure 10).

Figure 11 depicts two components, A and B, and the interface described in VRML 97. Notice that the figure only describes the interface.

Figure 10. An example of a simple component interface described in VRML 97

```
PROTO VRExample [

  eventIn  SFBool inputValue

  eventOut SFBool outputValue

  field MFNode children[]

] {}
```

Figure 11. The interface of two components with a Boolean slot

```
PROTO A [

  field ...

  eventIn SFBool outSlot

] {}


PROTO B [

  field ...

  eventOut SFBool inSlot

] {}
```

The concrete components with the corresponding data have to be described separately (cf., Figure 12).

The programmer's view is depicted in Figure 13, where the two components that are connected via slots are implemented. In our example, a **Boolean** value will be sent through the communication network. The message-mechanism is implemented in the base class of **A** and **B**, namely **VRNode**, in which the base methods for the communication are implemented. The **main()**-function creates two components. Next, the components **aComp** and **bComp** and their slots (in this case the **Boolean** slot) are connected together by calling the **AddListener()**

Figure 12. Two components with the routing statement

```
DEF a A {
  field ...
  ....
  }

DEF b B {
  field ...
  ....
  }

ROUTE a.outSlot TO b.inSlot
```

method. In our example, the program calls the **Set()** method of the component **aComp** that results a change of the corresponding **Boolean** value of the out-slot. Next, all listeners are notified by calling the **NotifiyListener()** method. Consequently, component **bComp** gets the corresponding **Boolean** value in the **ValueChanged()** method, and finally it prints the value on screen.

To be independent of the component **aComp**, an underlying graphic library, we encapsulate direct calls to the actual graphics library in certain graphical classes. Instead of creating the scene graph using library specific calls, often a so-called meta scene graph will be built.

Figure 14 shows the three graphs of a simple example. In this case, a clickableButton consists of two separate geometries: one of the up-state (not pushed) and one for the down-state (pushed). If the user clicks the button, it remains in down-state until the user clicks it again.

The leftmost graph structure presents the VRML 97 data structure generated by the SAVE parser. Note the thick black lines connecting the nodes. These lines represent parent-children relationships. The graph structure in the middle of Figure 14 constitutes the meta scene graph. Again, the thick black lines represent

Figure 13. The programmer's view of the SAVE component-oriented approach

```
/* The base class VRNode implements all methods for the
 * registration and removal of the slots
 */
class A : public VRNode {
  public:
    int outSlotIdx;

    A() {
      outSlotIdx = AddOutput(&typeid(VRBool),"outSlot");
    }

    void Set(bool b) {
      SetBoolOutputValue(outSlotIdx, b);  // Create 'VRValue'
      NotifyListeners(outSlotIdx);        // Notify listeners about
    }                                     // new value
};

class B : public VRNode {
  public:
    int inSlotIdx;
    B() {
      inSlotIdx = AddInput(&typeid(VRBool), "inSlot");
    }
```

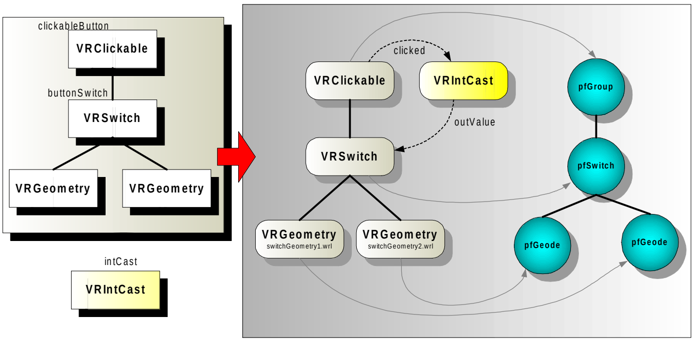
the parent-children relationships. This structure is derived from the VRML 97 scene graph. The right data structure is the scene graph of the graphics library which contains group-, transformation- and geometry-nodes. Note that the component communication network is not part of this scene graph.

Figure 13. The programmer's view of the SAVE component-oriented approach (continued)

```
// ValueChanged is called whenever the out-slot emits a value to
    // the in-slot.
    void ValueChanged(int in,
                      int inSlotHandle,
                      const VRValue* value) {

       // Which slot?
       if (inSlot == in) {
         VRBool* v = (VRBool*)value;  // Get the Boolean value
         cout << v->GetBoolValue() << "\n";
       }
    }
};

int main(int argc, char** argv) {
    A* aComp = new A();
    B* bComp = new B();

    // Connect component aComp with component bComp
    aComp->AddListener("outSlot", bComp, "inSlot");

    // Change the value of aComp. Consequently change the value of
    // component bComp
    aComp->Set(false);

    return 0;
}
```

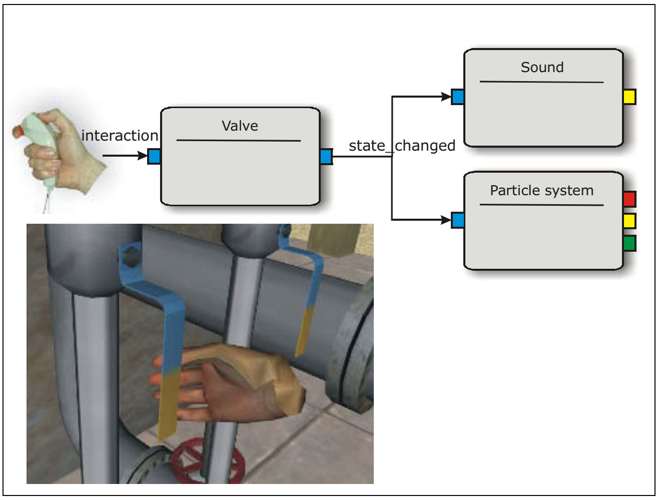
Building the meta scene graph does not require any knowledge of the underlying graphics library. The graphical classes can easily be replaced by classes that encapsulate calls to an alternative graphics library, and no further changes in the source are required to add new graphical objects in the virtual environment. The mapping of the VRML source to the actual scene graph is performed in two

Figure 14. From the description to the scene graph



```
ROUTE clickableButton.clicked TO intCast.inState
ROUTE intCast.outValue TO buttonSwitch.setChoice
```

Figure 15. The user interacts with the virtual valve that changes its state and forwards its state-changed messages to the sound and a particle system component



steps. First, our parser identifies all VRML nodes and creates a corresponding VRML node representation. Second, the VRML node is traversed recursively and each node is converted to its corresponding meta scene graph node representation. The result, in fact, is a meta scene graph. Finally, after a second traversal, the component communication network is established.

Figure 15 depicts an example of components that are connected together. The user holds a joystick with an integrated tracking receiver, which tracks both the position and the orientation of the user's hand. Accordingly, the virtual hand can be moved. While focusing the valve and after clicking to the interaction button of the joystick, the message will be propagated through the communication network (Figure 15). In our example the valve sends its message to a sound component that plays a sound file; the other listener is a particle system that simulates the outcoming water.

AMIRE (Authoring Mixed Reality)

In the AMIRE project the main goal is to provide mixed reality to other professionals than programmers and to facilitate efficient creation and modification of mixed reality applications (www.amire.net). With the mixed reality authoring tool, people with lesser programming skills should be able to develop MR applications cost effectively with fewer resources and in reduced time. The AMIRE project is an EU-funded project. The AMIRE authoring tool is based on two main principles: the use of user-centered design approach and open source code. Different authors have been involved all along the development process,

Figure 16. The main goal of AMIRE is an authoring tool based on a component-oriented approach to achieve MR-based applications (One of these applications is a refinery training program, where employees get more detailed information using a Tablet-PC.)

and they have affected the requirements as well as the user interface design. The tool is developed as open source and offered to all mixed reality developers. The project aims at more widespread use of mixed reality in different applications domains. The AMIRE team wants to promote the use of mixed reality in new application fields by more heterogeneous developers. With the high reuse of the MR content and easily maintainable, structured component libraries, the development times decrease and rapid prototyping of MR applications is possible.

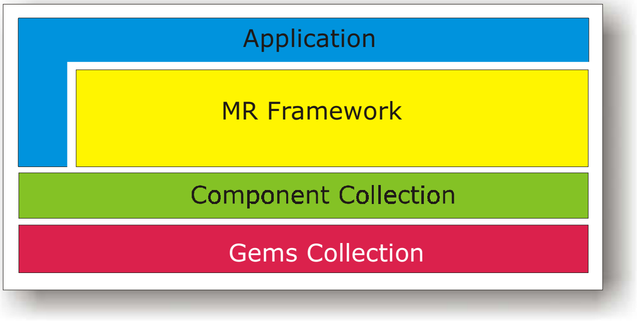
The Component-Oriented Design of AMIRE

AMIRE wants to adopt existing solutions and provide efficient means to encapsulate different solutions (called gems) in a uniform way (called components). Similar to existing gem collections (e.g., game programming gems), AMIRE offers an MR gem collection containing efficient solutions to individual mixed reality problems. A gem could be an object recognition library, a library for the graphical user interface, a tracking library, or simply a 3D model loader. The gem collection is integrated into the AMIRE framework. Typically, MR gems can be reused in many different applications. For example, a “work path animation” gem that visualizes the workflow of a special machine in an oil refinery can be reused to explain painting techniques of a famous painting in a museum.

The MR gems in turn can be used to build application-specific MR components (e.g., a navigation component for the museum application), as well as an MR framework that defines how MR components can communicate with each other and can be integrated into an application. The MR framework provides the required infrastructure. MR components represent solutions for particular domain-specific problems (e.g., MR-based museum application), and they typically combine and extend MR gems towards advanced high-level features of an MR application. MR components feature a unified interface that allows easy configuration.

Figure 17 shows the different layers of AMIRE starting from the gem layer including the libraries and C++ solutions. The second layer is the component layer that defines the structure and the interface of the components that are used in the application. The MR framework is the glue of the AMIRE project, because it integrates both the MR gems and the MR components. A detailed description of the AMIRE framework can be found in Haller, Zauner, Hartmann, and Luckeneder (2003) and Zauner, Haller, Brandl, and Hartmann (2003). Some of the most important features are:

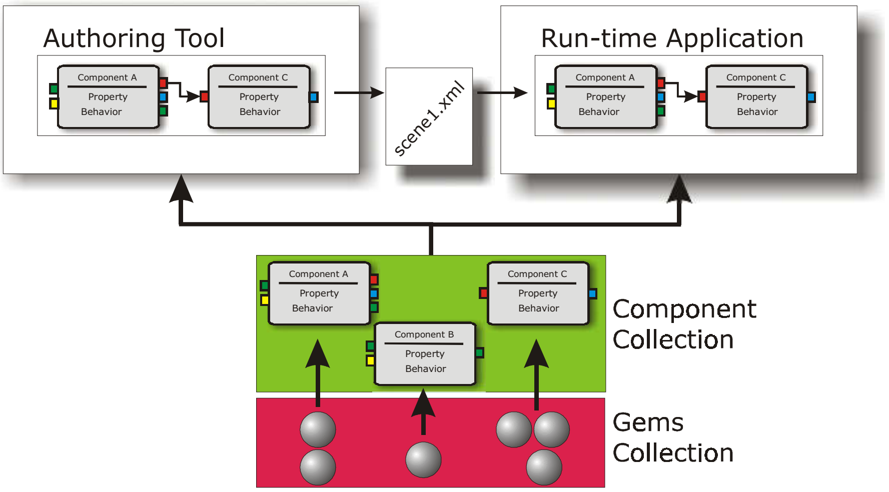
Figure 17. The different layers of AMIRE



- A generic configuration mechanism of components by so-called properties.
- The communication between components is based on in- and out-slots.
- The framework provides base conventions for 2D and 3D components. Only with these conventions is it possible to build a generic authoring tool that uses MR for the authoring process.
- A prototype-oriented approach is based on two kinds of component prototypes—basic components and composed components.
- The authoring tool supports the dynamic loading of C++ and XML-based libraries at runtime.
- An MR application can be saved and reloaded. The file format is XML.
- The integration of an object recognition unit (cf., ARToolKit).

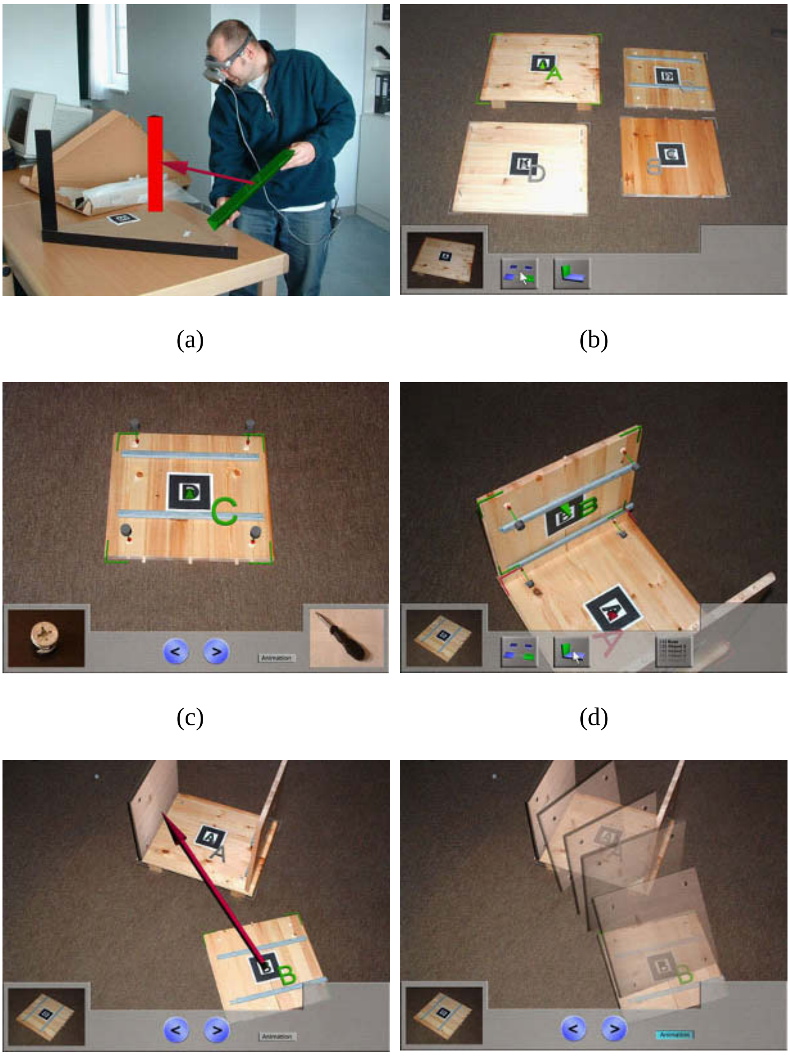
Figure 18 depicts the two-process approach of AMIRE, starting from the authoring tool to the run-time application. Both applications are based on the same AMIRE framework and on the same component repository. Of course, the authoring tool includes more components that do not necessarily have to be integrated into the run-time application. Due to the abstraction via components, an exchange of the underlying gems can be guaranteed: whenever the technology changes, programmers only have to include the new DLL, but the code for the run-time application doesn't have to be changed again. This is, of course, a great advantage, especially in a field in which new hardware and new base technologies come out every year.

Figure 18. From the authoring tool to the run-time application



In contrast to SAVE, AMIRE uses XML for the description of the components interface and for the description of a scenario. A prototype-oriented approach is used to create new component instances. This means that AMIRE has prototypes of specific components and an interface to make clones of each prototype. Two kinds of component prototypes are available. The first one is a native kind, completely written in C++ and packed into dynamic libraries such as the DLL format of Microsoft Windows systems or the shared object format of UNIX systems. The second kind is based on the existing set of prototypes and is called a composed component prototype. It consists of a component network, an export list of slots, and a configuration export for the components in the network. A composed component prototype is handled like a native prototype. Hence the author does not see any difference between using instances of a native and a composed component prototype. Authoring an application without generating and compiling additional C++ sources requires the dynamic loading of libraries. The AMIRE framework provides an interface to load and to replace a library at runtime. Currently, two kinds of libraries are supported, namely C++-based libraries and XML-based libraries of composed components. As mentioned before, the persistence of an application is supported by an XML-based file format. Such an XML file contains a list of library dependencies, the component instances, and the connections between them. Furthermore, an XML format for libraries of composed component prototypes is defined.

Figure 19. Some snapshots of the FaiMR program, which is based on the AMIRE framework: Figure (a) shows a general overview of the application. Figures (b), (c), and (d) show the view of the furniture expert who can assemble the furniture. The assembly scenario file is stored in an XML file. The end user has the view as depicted in (e) and (f), where he/she has a new furniture part. The arrow and the animation show how the furniture has to be assembled.



(a) (b)

(c) (d)

(e) (f)

In addition to the refinery training application, we developed another application, a furniture assembly instructor program (called FaiMR) that is based on the AMIRE framework: printed instruction manuals for furniture assembly often have one disadvantage in common—it takes a lot of time to make sense of their meaning since they show several steps of assembly together in a few pictures. Furthermore, it is hard to find the connections between the instructions printed in 2D and the real parts. The idea of this work is to connect the instruction directly to the parts of a piece of furniture. To do this, mixed reality is used, which combines reality (recorded by a Web cam) with additional information using common computer graphics in 2D and 3D which are overlaid. A closer description of this application can be found in Brandl (2003) and Zauner (2003).

Future Trends

Even if the field of mixed reality is rather new and even if there are a lot of unresolved problems, such as in tracking and hardware devices, fast development for prototyping of MR is becoming more and more important. Newcomers—even non-programmers—should be involved in this fascinating world; because of their artistic and usability knowledge, their constructive input for new ideas would be very fruitful for new MR applications. But actually, especially these persons are hesitant to get involved, because of the lack of adequate authoring tools. Performance is of course a very important factor in a real-time application like MR applications.

The component-oriented design itself should be based on different, well-established solutions (libraries)—this was, for example, the main goal of AMIRE. In the optimal case, there are already good solutions for different problems; the Virtual Reality Peripheral Network (VRPN) is a good example for tracking (www.cs.unc.edu/Research/vrpn). It is a well-designed library that supports a lot of different trackers. The same trend can be found in the game industry: Ten years ago, every game company implemented its own game engine, because the projects were smaller and the teams consisted of fewer than 12 people. But now, projects are becoming more and more complex, and not everything can be implemented starting from scratch.

Combined with a component-oriented approach is the use of adequate authoring tools for end users who should be able to develop their own MR applications with lesser or no programming skills. In the rare cases end users have programming skills, they often lack scripting or 3D modeling skills. But they are specialists in their field and they know what they would like to have, but they cannot experiment with the new media. Therefore, the usability and the graphical user interface of these tools have to be as simple as possible.

The Need for More Authoring Tools

Nowadays, in the computer game industry, every big game is implemented by using corresponding authoring tools (cf., GTk-Radian depicted in Figure 20). There are already different, well-defined roles in the production process of a game—from the programmers who develop the engine, through the game developers to the game designers and level designers. And the game developers would never touch the engine—they are concerned with the game itself, and they build the application with the corresponding authoring tools that are offered together with the game engine. The quality of these tools is so good that even teenagers are able to develop great new levels for their own games.

What we need is more effort in the development of good authoring tools for the creation of VR/AR/MR applications. Currently, the assortment of such tools is not very large. Figures 21 and 22 depict two commercial authoring tools designed for VR/MR applications. Both tools are designed for designers and expert users in the sense that they should know about the basics of 3D. Especially Virtools (www.virtools.com) offers a large number of components that can be connected together to build powerful applications that are used in different VR hardware devices (also for CAVE applications).

Figure 20. GTk-Radiant is one of the most-used level design tools for Quake

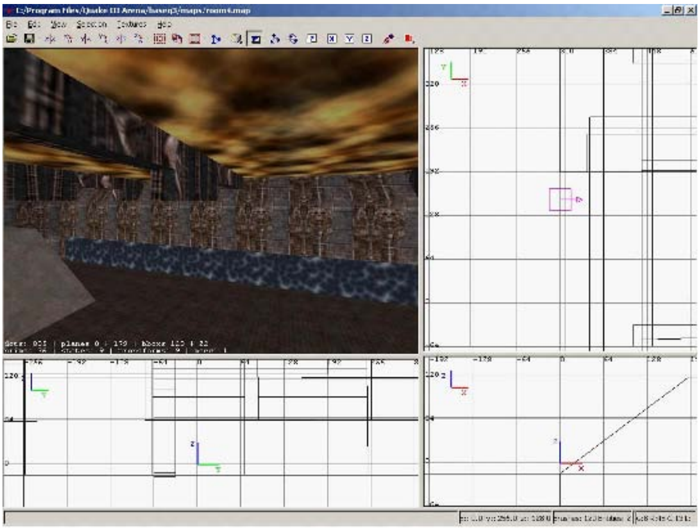
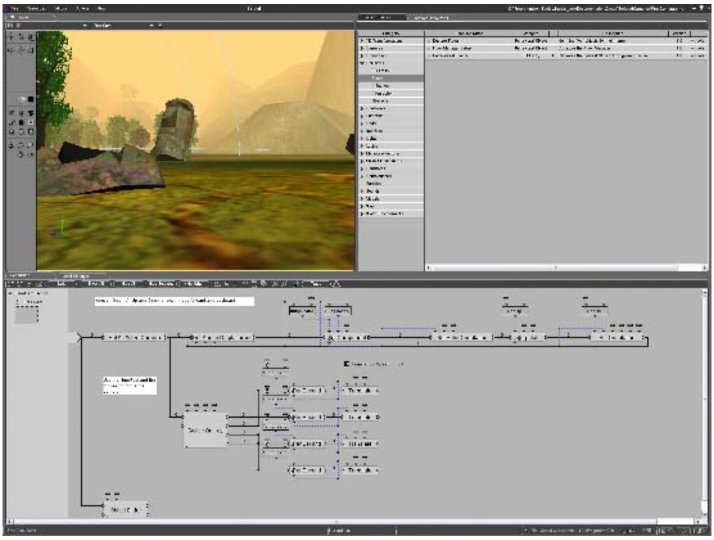


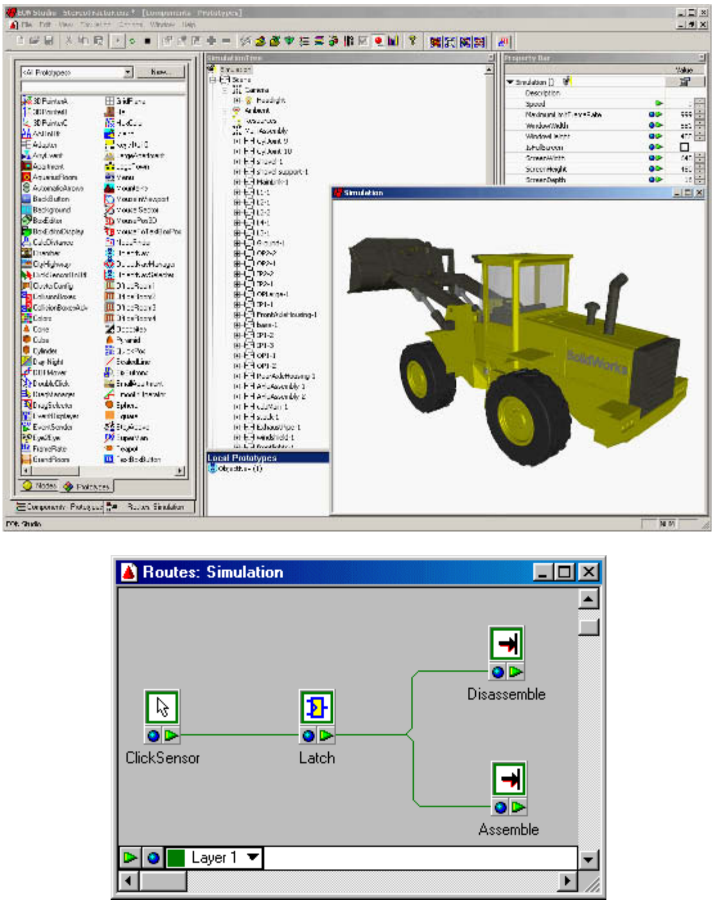
Figure 21. The authoring tool Virtools is frequently used for VR applications (www.virtools.com)



An authoring tool for developing an MR application can be divided into the following sub-authoring tools:

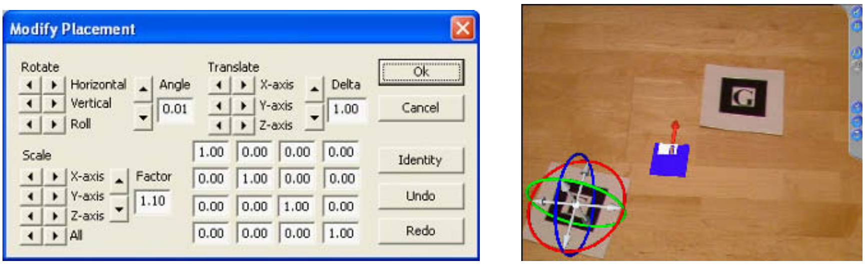
- **Placement tool**: This tool allows the user to place the virtual objects into the virtual world.
- **Configuration tool**: The properties of components are always different from each other and should have to be modified accordingly. A simple user interface with the possibility of a flexible change of these properties should be a key issue of this tool.
- **Connection tool**: Finally, components have to be connected together. With our slot paradigm, the optimal graphical user interface is to drag and drop the slots and a visual line should show the connections. One of the biggest problems in this tool is the visualization method of the connections. In our opinion, all components should be visualized. This means that the graphical user interface can become very, very complex—but it does not hide information from the user. Different abstraction layers would help a lot, because not everyone is interested in all the possibilities.

Figure 22. EON Studio uses a component-oriented approach that allows experts to implement their VR applications in a rapid way (www.eonreality.com)



Currently, in the field of AR, we have no commercial tool available and only prototypes developed in research projects (e.g., in ARVIKA, DART, STAR, AMIRE). The DART project is built as a collection of extensions to the Macromedia Director multimedia-programming environment, and therefore primarily developed for designers who want to develop their own AR application (MacIntyre, 2003). Another approach is postulated by the AMIRE project, in which the authoring tools are based on a component-oriented approach, but not on an existing multimedia authoring tool like Director. The AMIRE goal is to take the real environment and to place the objects there. Figure 23 depicts how the authoring in an MR environment could look—especially the placement of the virtual objects into the world could be realized in a very intuitive way using the

Figure 23. AR-based placement tool of AMIRE



Angle 0.01

Delta 1.00

Factor 1.10

1.00	0.00	0.00	0.00
0.00	1.00	0.00	0.00
0.00	0.00	1.00	0.00
0.00	0.00	0.00	1.00

real environment. The setting of the properties for the component and the routing of the components is done in 2D.

We expect more input in this area in the coming years, because current solutions are not the best in regards to usability and GUI interfaces.

Conclusions

The implementation of mixed reality applications is a very difficult task, because it includes a lot of different skills (e.g., sound programming, real-time graphics, different hardware in- and output devices, possibly AI techniques, etc.). Therefore, we need a good architecture. We described a component-oriented approach for developing mixed reality applications. The benefits of this development approach are a fast implementation that allows an easy integration of corresponding authoring tools.

We achieve the following benefits if we use a component-oriented approach:

- **Flexibility:** Each component can be connected to another if it corresponds to the right interface.
- **Reusability:** Components can naturally be reused in the framework.
- **Extensibility:** Different components have to be added to the framework afterwards. A redesign of the framework has to be avoided by adding new components.
- **Easy communication:** The component communication has to be as simple as possible and as fast as possible.

The main advantage of the component-oriented approach is the high degree of flexibility and extensibility: Even if the underlying technology is different, we can change the tracking system quite simply, but the system runs just as well.

References

Azuma, R. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 355-385.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B. (2001). Recent advances in augmented reality. *IEEE Computer Graphics and Applications, 21*(6), 34-47.

Bauer, M. et al. (2001), Design of a component-based augmented reality framework. *Proceedings of the International Symposium on Augmented Reality* (ISAR 2001), New York.

Bimber, O., Fröhlich, B., Schmalstieg, D., & Encarnação, L.M. (2001). The virtual showcase. *IEEE Computer Graphics & Applications, 21*(6), 48-55.

Brandl, A. (2003). *Entwicklung einer interaktiven Möbelanleitung auf Mixed Reality Basis*. Master's Thesis, Upper Austria University of Applied Sciences, Media Technology and Design, July 2003, Hagenberg, Austria.

Dachselt, R. (2001). Contigra—towards a document-based approach to 3D components. *Proceedings of Structured Design of Virtual Environments and 3D-Components, a workshop at the Web3D Workshop,* Shaker Verlag, Aachen.

Dobler, D., Haller, M., & Stampfl P. (2002). ASR—augmented sound reality. *Proceedings of the ACM SIGGRAPH 2002 Conference Abstracts and Applications* (p. 148), San Antonio, Texas.

Dörner, R., & Grimm, P. (2001). Building 3D applications with 3D components and 3D frameworks. *Proceedings of Structured Design of Virtual Environments and 3D-Components, a workshop at the Web3D Workshop,* Shaker Verlag, Aachen.

Friedrich, W. (2000). *ARVIKA—Augmented reality for development, production, and service*. Tagungsband des Informationsforum Virtuelle Produktentstehung (IVIP).

Geiger, C., Reimann, C., & Rosenbach, W. (2000). Design of reusable components for interactive 3D environments. *Proceedings of the Workshop on Guiding Users Through Interactive Experiences: Usability-Centred*

Design and Evaluation of Virtual 3D Environments, Paderborn, Germany.

Haller, M. (2001). A component-oriented design for a VR-based application. *Proceedings of the International Workshop on Structured Design of Virtual Environments and 3D-Components at the Web3D 2001 Conference,* Paderborn, Germany.

Haller, M., Dobler, D., & Stampfl, P. (2002). Augmenting the reality with 3D sound sources. *Proceedings of the ACM SIGGRAPH 2002 Conference Abstracts and Applications* (p. 65), San Antonio, Texas.

Haller, M., Holm, R., Priglinger, M., Volkert, J., & Wagner, R. (2000). Components for a virtual environment. *Proceedings of the Workshop on Guiding Users Through Interactive Experiences: Usability-Centred Design and Evaluation of Virtual 3D Environments,* Paderborn, Germany.

Haller, M., Zauner, J., Hartmann, W., & Luckeneder, T. (2003). *A generic framework for a training application based on mixed reality.* Technical Report, Upper Austria University of Applied Sciences, Media Technology and Design.

Kato, H., Billinghurst, M., Blanding, B., & May, R. (1999). *ARToolKit.* Technical Report, Hiroshima City University, Japan.

MacIntyre, B., Gandy, M., Bolter, J.D., Dow, S., & Hannigan, B. (2003). DART: The designer's augmented reality toolkit. *Proceedings of the Second International Symposium on Mixed and Augmented Reality* (pp. 329-330), IEEE, Tokyo.

Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems, E77-D*(12).

Piekarski, W., & Thomas, B.H. (2003). An object-oriented software architecture for 3D mixed reality application. *Proceedings of the Second International Symposium on Mixed and Augmented Reality* (pp. 247-256), IEEE, Tokyo.

Schmalstieg, D., Fuhrmann, A., Szalavari, Z., & Gervautz, M. (1996). Studierstube—an environment for collaboration in augmented reality. Extended Abstract, *Proceedings of Collaborative Virtual Environments '96,* Nottingham, UK.

Zauner, J., Haller, M., Brandl, A., & Hartmann, W. (2003). Authoring of a mixed reality assembly instructor for hierarchical structures, *Proceedings of the Second International Symposium on Mixed and Augmented Reality* (pp. 237-246), IEEE, Tokyo.

Glossary

A

Access model: Model that specifies the different roles or stereotypes of a specific software system as well as the access capabilities for each kind of user.

Aesthetics: The study of the particular pleasures offered by communications media of all forms.

Agent's cognitive module: The bulk of the agent, according to the classical definition of agent as a continuous perception-cognition-action cycle. In this module, perceptions are analyzed and decisions about actions are made.

Agent's perceptual module: A solution to modeling the agent's perception by focusing on its perceptual sensors.

Agent-based architecture: A software architecture in which the basic building block is a software agent. In an agent-based architecture, each agent is capable of performing a certain set of tasks, and is capable of communicating with other agents to cooperate with them in the execution of those tasks.

Algebraic semiotics: The study of optimal representation via mappings of systems of signs, using methods from algebra and sociology.

Authoring tool: End users can use authoring tools to develop their own VE applications without having programming skills.

Avatar: Comes from Sanskrit and means reincarnation.

B

Believability: (As opposed to realism) Sensation felt by a human user of a virtual environment that catches his/her attention, not exactly because of its fancy graphics, but because of the richness of its avatars' behaviors.

Blackboard: A common data structure that will be used concurrently by several agents to build plans collaboratively. Each agent can observe the contents of the blackboard and decide proactively when it is appropriate to modify it.

Blend: A combination of two (or more) sign systems into a single sign system.

C

Clarity of perception: A measurement of the ability to distinguish what kind of object is being perceived by the agent.

Collaborative task planning: In an IVET, the students will be posed with problems consisting of determining how to reach a certain final state in the virtual environment from the current state. The system should also have the capability of building solution plans. A plan consists of a sequence of tasks that the students should perform in the VE. The construction of the plan will be performed collaboratively among different agents contributing with different types of knowledge.

Collaborative virtual environments (CVEs): Software programs that support users in managing communication across multiple media across the network. They are populated by objects and user representations, and provide a means of communicating, socializing, and exchanging ideas, as in real-life social systems.

Component: Component as a separated entity with a specific size. It is characterized by dependencies and the framework permits a dynamic loading of components. Components can be either large or small, but they have to be of a clear structure. A visible component has its geometry, a property, and finally it is characterized by a behavior.

Composed component: Components can be very simple, but they can also be composed of other simple components. They are often called composed components or compound components.

Conceptual model: Abstract representation of a system that describes its static components, relationships, and dynamics in terms of elements of the universe of discourse instead of using technical terms and implementation units.

Content model: An abstract characterization of the perceptual content of interactive media.

Conventional groupware: A system that allows remotely located participants taking part in a collaborative activity (shared activity) to view the context and interact through it.

D

Data-sharing mechanism: The mechanism by which a particular virtual environment is shared between different processes implementation.

Deliberation: Conscious, attentive process that uses general purpose resources to focus and address the primary concerns and goals of the agent.

Design guidelines: Design guidelines provide a way of encapsulating a research's results and providing application designers with direct advice and design solutions.

Design methodology: The study of the method of design.

E

Environment model: The methods and structures by which the system allows description of collaborative virtual environments.

F

Framework: The structure of a typical client or platform in terms of the services it provides to the user.

G

Gem: Similar to existing gems collections (game programming gems, graphic gems), there exists the terminology MR gem. A gem represents an efficient solution (e.g., software code, library) to a specific MR problem.

H

Hypermedia: Associative structure of multimedia nodes that can be freely browsed.

Hypermedia design: Systematic process oriented towards producing usable and useful hypermedia systems.

I

Information rich virtual environments: Virtual environments augmented with abstract information such as text, numbers, and graphs.

Information visualization: The representation of information using graphical media; a special class of semiotic morphism.

In-slot: Each component is composed of in-slots, which can receive data from the previous component.

Intelligent tutoring system (ITS): A tutoring system intended to adapt the teaching and learning process to the needs of every individual student. To that aim, the system should have knowledge and competence in four distinct areas that give rise to the four classical components in the architecture of an ITS: expert module (knowledge about the subject matter); tutoring module (competence about teaching and learning); student module (knowledge about the student); and communication module (competence about communicating with the student).

Intelligent virtual agent (IVA): An autonomous embodied agent usually in a 3D interactive graphical environment or virtual environment (VE), which draws on artificial intelligence (AI) and artificial life (Alife) technology so as to interact/communicate intelligently with its environment and with human users/IVAs.

Intelligent virtual environment for training (IVET): Results from the combination of a virtual environment (a 3D graphical model) and an intelligent tutoring system (ITS). The goal of this kind of system is to train one or more students in the execution of a certain task. IVETs are able to supervise the actions of the students and provide tutoring feedback.

Interaction Analysis (IA): Has its roots in the social sciences, and perceives knowledge and action as fundamentally social in origin, organization, and use. It studies human activities, such as talk, non-verbal interaction, and the use of artifacts and technologies. It is primarily defined by its 'analytic foci' or ways into a videotape. Such foci include: structure of events; temporal organization of activity; turn-taking; trouble and repair; and spatial organization of activity. Important to *Interaction Analysis* is the data analysis by a group of analysts.

Interaction machine: A model of computation that incorporates interaction with the environment in which the machine exists; inherently more powerful than Turing machines.

Internal-external interaction: Whether in a collaborative activity the users' interactions take place face-to-face in the real world, or via the computer using the tools the system provides. Face-to-face user interaction in the real world is referred to as being 'external' to the system. User interactions and communication via the tools that the system provides is referred to as 'internal' to the system (e.g., between virtual actors within the environment).

M

Magic interfaces: Interfaces that are not inspired by natural interaction and thus less intuitive but potentially more effective.

Mixed reality (MR): Involves the merging of real and virtual worlds somewhere along the reality-virtuality continuum, which connects completely real to completely virtual environments.

N

Network topology: The way in which a set of clients are networked together. Typically this is client-server (everyone connects to a central server forming a "star") or peer-to-peer (everyone connects to everyone).

O

Out-slot: Each component is composed of out-slots, which can send data to another component.

P

Path planning: For many tasks to be carried out in a virtual environment, it is necessary to navigate along the space avoiding collisions with objects and possibly minimizing distance. A path planning agent will calculate the best trajectory for each displacement that an avatar must do in the VE, from geometrical information related to the VE.

Pedagogical agent: A software agent that is in charge of the supervision of the learning process in an IVET. Pedagogical agents can be embodied and inhabit the virtual environment together with the students, or they can be just a piece of software that interacts with the student via voice, text, or a graphical user interface.

Presentation design: Design of the appearance and organization of the user interface.

Process model: Both an ordering of the activities that comprise a design method and a characterization of the linkages between them.

R

Reaction: Automated, pre-attentive process triggered by the agent in response to any change in the environment state or in the agent internal state.

S

Scalability: The ability of a collaborative virtual environment system to support large numbers of users and large virtual environments.

Scene graph: Provides a high level of abstraction in computer graphics and stores the whole scene in the form of a graph of connected objects (often called nodes).

Semiotic morphism: A representation of meaning and/or functionality, given as a mapping from one sign system to another.

Semiotics: The study of signs and systems of signs.

Semiotics: The study of the way humans find meaning in the world around them.

Single display groupware: A system that allows the participants taking part in a collaborative activity (shared activity) to view the context of interaction through a single, shared display.

Space-based relationship: Relation among two or more information items which establishes the position in a 2D or 3D space of an item, taking into account the position of another item.

T

Time-based relationship: Relation among two or more information items which establishes when an information item starts, ends, or how long it takes, taking into account when another item starts, ends, or how long it takes.

V

Virtual actors: Graphical forms that represent the collaborative virtual environments' inhabitants. They provide an appropriate body image to the users who participate in the collaborative activity to represent them to others, as well as to themselves.

Virtual environment (VE): A computer-synthesized, three-dimensional environment in which a plurality of human participants, appropriately interfaced, may engage and manipulate simulated physical elements in the environment, and in some forms may engage and interact with representations of other humans—past, present, or fictional—or with invented creatures.

Virtual environment modeling: Specification of a VE using concepts and relationships of a conceptual model.

Virtual reality melting pot: A theory that many related technologies are melding together through mutual advances in hardware, software, theories, and methodology into a larger technology for manipulating human senses in virtual, augmented, and real spaces.

Virtual world: The class of media experiences that provide a sense of immersion and closure.

Virtuality: A sense of being engaged with non-physically present entities through material mediation in the immediate real world.

About the Authors

Maria-Isabel (Maribel) Sánchez-Segura has been faculty member of the Computer Science Department at the Carlos III Technical University of Madrid since 1998. Her research interests include software engineering, interactive systems, and usability in interactive systems. She holds a BS in Computer Science (1997), an MS in Software Engineering (1999), and a PhD in Computer Science (2001) from the Technical University of Madrid. Dr. Sanchez-Segura is author of several papers related to the improvement of virtual environments development from the software engineering point of view, published recently in journals such as *Software Practice and Experience, Interacting with Computers,* and *Journal of Systems and Software.* She is also the author of more than 20 papers presented at several virtual environments and software engineering conferences, and is one of the instructors (joint Carnegie Mellon and Technical University of Madrid researchers) in both tutorials held at the ACM CHI conference in Vienna in April 2004 and the IEEE ICSE conference in Edinburgh in May 2004.

* * * *

Ignacio Aedo, who holds an undergraduate degree and a PhD in Computer Science from Polytechnic University of Madrid, has been actively involved in the domain of interactive systems research since 1990. In this period, he has participated in several national and international projects related to interactive systems. He is the author of several books about interactive systems, as well as papers published in journals and conferences. His main research interests include human-computer interaction issues, Web engineering, control access models, and technology in education. He is also a member of the scientific committees of several publications and conferences related to his interest areas.

Antonio de Amescua holds a PhD in Computer Science and is a full professor in the Computer Science Department of the Carlos III University of Madrid. He also has 21 years of experience working at the Polytechnic University of Madrid and for the Iberia Airlines company. He has edited books and international research papers in the areas of software engineering methodologies and software process improvement. He was the research project leader for the development of the Information System Development Methodology for the Spanish Administration and has participated in projects sponsored by the European Union.

Angélica de Antonio has been faculty member since 1990 in the Languages, Systems and Software Engineering Department (of which she is currently Sub-Director) at the Technical University of Madrid (UPM), where she also coordinates the doctoral program since 2000. She is Director of the "Decoroso Crespo Laboratory" of the UPM since 1995, where she has led several R&D projects in the areas of intelligent tutoring systems, e-learning, virtual environments, and intelligent agents. Professor de Antonio was a Resident Affiliate at the SEI (Carnegie Mellon University) during 1995. From 1991 to 1995 she was researcher at the Artificial Intelligence Laboratory (UPM), and Assistant Director of the SETIAM section of CETTICO (Center of Technology Transfer in Computer Engineering), specialized in the transfer of computer technologies to assist the disabled.

Kirstie L. Bellman started the Aerospace Integration Sciences Center (AISC), which focuses on advancing system and model integration methods, new analytic techniques, and evaluation tools for assessing the impacts of new technologies. She has 35 years of experience in both laboratory research and the development of models and information architectures for large government programs. Her published research spans a wide range of topics in cognitive science, neuroscience, and computer science. While at the U.S. Defense Advanced Research Projects Agency, she extended virtual worlds to education, business, and research environments. With academic partners, Dr. Bellman is developing new mathematical approaches to the analysis of virtual worlds. She received an award from the Office of the U.S. Secretary of Defense for excellence in her DARPA programs. She was also honored as a Fellow, in 2000, by the American

Association for the Advancement of Science, for work in complex systems and virtual worlds.

Paloma Díaz earned her undergraduate degree and PhD in Computer Science from the Polytechnic University of Madrid. She is full professor at the University Carlos III of Madrid, where she is the head of the DEI Laboratory (dei.inf.uc3m.es). Her research interests mainly concern topics such as hypermedia and Web engineering, software development methodologies, and formal models for representing, reusing, and interchanging information and the application of information and communication technologies in education.

Juan Manuel Dodero works as a Lecturer in the Computer Science Department at the University Carlos III of Madrid (Spain) since 1999. He received his undergraduate degree in Computer Science and MSc in Knowledge Engineering, both from the Polytechnic University of Madrid, and his PhD in Computer Science from the University Carlos III. He has prior experience as an object technology consultant and a R&D engineer for several Spanish companies. His research interests include technologies to support education and learning, knowledge management, computer-supported cooperative work, and multi-agent systems.

Daphne Economou is a Lecturer in the Department of Cultural Technology and Communication at the University of the Aegean. She received a PhD on the topic of Virtual Actors in Collaborative Virtual Environments for Learning in 2001. She has working experience at Sony Broadcast & Professional Research Labs (BPRL) as a Human Factors Engineer, leading the research effort on a number of key research projects. In 2002 she returned to academia and has been leading the MSc Interactive Multimedia program at the University of Westminster. Her research interests include user-centered and ethnographic approaches to system design, and the use of virtual reality technologies in education.

Clive Fencott, BA, MSc, DIC, PhD, lectures and researches on the theory and design of virtual environments in general, and computer games in particular. He is particularly interested in predictive content modeling and its experimental verification via a range of techniques including eye-tracker technology. He uses his research findings extensively in teaching the Computer Games Design degree and on master's modules at the University of Teesside. He has some 40 publications to his name and is currently writing a book on the theory of computer games. He is also currently supervising five PhD students in this field.

Emmanuel Frécon was born in France and has been living in Sweden for more than 10 years. He received his MSc from the National Institute of Applied Science, Lyon, France (May 1993). He spent his last year of studies as an ERASMUS student at the Royal Institute of Technology, Stockholm, Sweden, where he presented his thesis on the integration of teleconferencing facilities in a virtual environment. He has been working at SICS, the Swedish Institute of Computer Science, since august 1994. His main research interest is the design and implementation of collaborative virtual environments systems, with a focus on distribution issues and application support. He is one of the main architects of the DIVE system. His recent work is published in several international journals such as *Presence: Teleoperators and Virtual Environments*, *IEEE Computer Graphics*, and *Applications and IEEE Communications Magazine,* and a number of IEEE and ACM conferences. He is co-editor of the book, *Inhabited Information Spaces—Living with your Data.*

Joseph A. Goguen is Professor and Director of the Meaning and Computation Lab at the University of California, San Diego. He was a Professor at Oxford, Senior Scientist at SRI, Senior Member of CSLI at Stanford, and a Professor at UCLA. He has given distinguished lectures and invited or keynote addresses at conferences on requirements engineering, semiotics, formal methods, metaphor theory, software re-use, sociology of science, and consciousness. He is author or co-author of more than 240 publications, including two books, (co-)editor of five others, and is known for his founding role in algebraic methods in computer science and in fuzzy logic.

Michael Haller is a Professor at the Upper Austria University of Applied Sciences, Department of Media Technology and Design, Austria, where he directs the EU-funded project AMIRE (Authoring Mixed Reality). He received his Master's of Science (1997) and his PhD (2000) from the Johannes Kepler University of Linz (Austria), where he worked on a component-oriented approach for virtual environments. Currently, he is performing research in real-time computer graphics, augmented reality and virtual reality, and human-computer interaction. He is author and co-author of more than 30 reviewed scientific publications.

Pilar Herrero is an Assistant Professor in the Department of Languages, Systems and Software Engineering in the School of Computer Science at the Universidad Politécnica de Madrid. Dr. Herrero received her European PhD in Computer Science from the Universidad Politécnica de Madrid in 2003 for her work on designing a human-like perceptual model for intelligent virtual agents.

She is a member of Decoroso Crespo Laboratory, a laboratory for the application of information technologies and communications to education. She is also a member of the Mixed Reality Laboratory (MRL) and the Communications Research Group (CRG) at the University of Nottingham in the UK.

Ricardo Imbert is an Assistant Professor in the Department of Languages, Systems and Software Engineering in the School of Computer Science at the Universidad Politécnica de Madrid. He is currently finishing his PhD on Cognitive Architectures for agents with behaviors influenced by personality and emotion, at the Universidad Politécnica de Madrid. Professor Imbert has been a member and Project Leader at the Decoroso Crespo Laboratory of the same university since 1996; the lab houses a research group of computer scientists blending technologies, such as virtual reality, software agents, and intelligent tutoring systems to create innovative computer learning environments.

Gonzalo Méndez is a graduate student at the Computer Science School of the Technical University of Madrid. He started working on educational software in 1997 and on virtual environments in 1998. His first work combining both fields was in the PRVIR project, an Intelligent Virtual Environment for Training in Nuclear Power Plants, in 1999. Since then, he has taken part in several projects that involve intelligent tutoring and virtual reality, such as MAEVIF. He has worked with Dr. Jeff Rickel on the integration of HeSPI, a virtual environment for planning maintenance tasks in nuclear power plants, and STEVE, an intelligent tutor for procedural training. His research interests at the moment include intelligent virtual environments for training and object-oriented software engineering.

Susana Montero earned a degree in Computer Science from the Universidad Carlos III de Madrid. Starting in 1999, she is a Lecturer in the Department of Computer Science at the same university. Her research interests include hypermedia development methodologies, CASE, knowledge representation, and their applications to hypermedia development process.

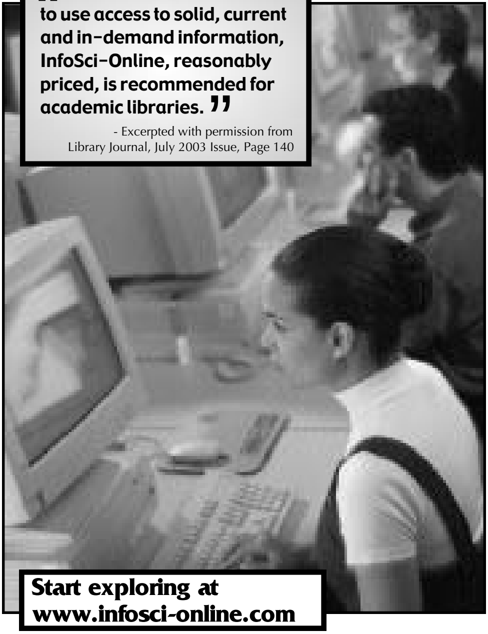
Steve Pettifer is a Lecturer in the Department of Computer Science at the University of Manchester. After working with ICL (now Fujitsu) on distributed database technology, he returned to academia and received his PhD on the topic of Distributed Environments for Virtual Reality Applications in 1999. His interests include virtual environments, high performance graphics, scientific visualization, human computer interaction, and the application of all these in collaborative scenarios.

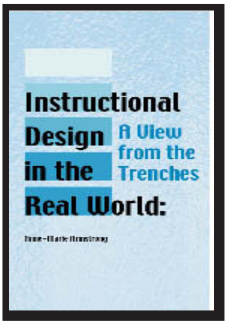
Jaime Ramírez earned his bachelor's degree in Computer Science from the Technical University of Madrid in 1996, and his PhD in Artificial Intelligence from the Artificial Intelligence Department of the Computer Science School at the Technical University of Madrid in 2002. His research activities mainly have been related to the field of verification of knowledge base systems, and the development of applications based on virtual reality and intelligent learning. Currently, he teaches computer programming courses at the Technical University of Madrid, and he works in the Decoroso Crespo Laboratory at the Technical University of Madrid as a collaborator professor.

Anthony Steed is a Lecturer in the Department of Computer Science, University College London. He received his PhD from Queen Mary College, University of London in December 1996. He has worked on many aspects of virtual environment systems, from evaluation of groups of users using collaborative virtual environments through to scalable rendering algorithms. He has published more than 50 papers and is co-author of the book, *Computer Graphics and Virtual Environments: From Realism to Real-Time*.

Chadwick A. Wingrave is currently a graduate student of the Virginia Tech Department of Computer Science. There, he is a member of the Center for Human-Computer Interaction and the 3D Interaction Group, led by Dr. Doug A. Bowman. He actively researches human cognitive behavior and response to computer interfaces, as well as the architectures and tools necessary to create them. His short-term goals are directed at creating more usable interfaces that allow for an increased utility of the technologies of virtual reality. Ultimately, he sees computer technology as a tool for the benevolent reshaping of human life.

Index

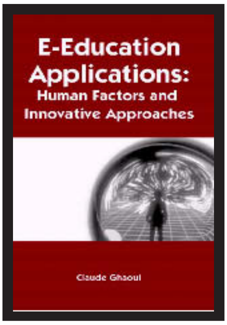




2004 RELEASE

E-Education Applications:
Human Factors and Innovative Approaches

Claude Ghaoui, PhD, Liverpool John Moores University, UK


E-Education Applications:
Human Factors and Innovative Approaches
Claude Ghaoui

E-Education Applications: Human Factors and Innovative Approaches enforces the need to take multi-disciplinary and/or inter-disciplinary approaches, when solutions for e-education (or online-, e-learning) are introduced. By focusing on the issues that have impact on the usability of e-learning, the book specifically fills-in a gap in this area, which is particularly invaluable to practitioners. The book is aimed at researchers and practitioners from academia, industry, and government, for an in-depth coverage of a broad range of issues, ideas and practical experiences on this subject. It aims to raise more awareness in this important subject, promote good practice, and share and evaluate experiences (advantages, disadvantages, problems faced and lessons learned).

ISBN 1-931777-92-6 (h/c) • US$74.95 • *ISBN 1-59140-292-1 (s/c)* • US$59.95
• 377 pages • Copyright © 2004

> *"The consideration of people with diverse needs, capabilities and cultural differences must direct scientists and education stakeholders to look at the World in a new light. Widening access requires reducing disabling conditions under which users of education work."*
>
> ***– Claude Ghaoui, PhD***
> ***Liverpool John Moores University, UK***

It's Easy to Order! Order online at www.idea-group.com or call 717/533-8845 x10
Mon-Fri 8:30 am-5:00 pm (est) or fax 24 hours a day 717/533-8661

INFOSCI
Information Science Publishing
Hershey • London • Melbourne • Singapore

An excellent addition to your library