A FRAMEWORK FOR COMPUTER-SUPPORTED COLLABORATION IN ARCHITECTURAL DESIGN

by

Wassim M. Jabi

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
(Architecture)
in The University of Michigan
2004

Doctoral Committee:

Professor Harold J. Borkin, Co-Chair
Professor Judith S. Olson, Co-Chair
Professor Edmund H. Durfee
Professor James A. Turner
In loving memory of my brother
Haytham Jabi, M.D.
1954-1995
ACKNOWLEDGMENTS

I would like to thank my colleagues and family members who have significantly contributed to the development of this dissertation.

First, I would like to thank my advisor, Professor Harold Borkin. Throughout my years at the University of Michigan, Professor Borkin offered advice and encouragement. His extensive experience in the area of computer-aided design helped me place my work in a larger context. He focused my attention on overlooked issues and helped me refine my dissertation argument. Professor Borkin also introduced me to Carl Luckenbach, FAIA to whom I am thankful for giving me my first job and sponsoring my application for immigration to the United States. Without his help, I would not have been able to remain in the United States, obtain my Ph.D. and become a U.S. citizen.

I would also like to think the members of my doctoral committee. My Co-Chair, Professor Judith Olson introduced me to the topic of computer-supported cooperative work and has offered an invaluable resource and connection to the research work in that field. I would also like to thank her for her patience and willingness to continue her participation in this committee. Professor James Turner introduced me, through several courses, to the topic of graphics and computer-aided design programming. Without his help and knowledge, I would not have been able to program any of the prototypes in this dissertation. Professor Durfee made me aware of important methodological issues and offered feedback from a computer science perspective. I thank him for his help and suggestions on how to improve this dissertation.

I wish to thank those who I have collaborated with. Dr. Theodore W. Hall and I tested an early prototype between Ann Arbor, Michigan and Hong Kong and co-authored
several research papers. Dr. Hall has offered valuable help with issues including the topic of the dissertation, algorithms, and even dissertation formatting. More importantly, however, he offered his friendship and support. Professor Brian Johnson, Professor Glenn Goldman, Michael Hoon and Professor Bob Martens have collaborated with me and co-authored research papers.

I extend my thanks to Professor Urs Gauchat, my Dean at the New Jersey School of Architecture, New Jersey Institute of Technology for his support of my career and encouragement to obtain my Ph.D. degree.

The Doctoral Program in Architecture at the University of Michigan supported my work by providing academic, logistic, and financial help. I thank the Doctoral Program Chairs, Professor Robert E. Johnson, Professor James Chaffers, and the late Professor Emmanuel-George Vakalo, for their support of my studies. I thank the assistant to the Chair, Ms. Jean Ellis, for her friendship and her help with university policies. I also wish to thank my fellow doctoral students, especially Iakovos (Jack) Potamianos, Scott Johnson, Patrick Tripeny, Mark Luther, and George Yeh, for their camaraderie.

I wish to thank my parents, the late Adnan and Da’ad Jabi – may they rest in peace. They supported me and my brothers under very difficult circumstances and instilled in us the value of good education. I also thank my brothers Samer and Muhammad for their support and financial help. My eldest brother, Haytham, who passed away unexpectedly in 1995, was the one who encouraged me to apply for post-graduate programs in the United States and offered selfless help and guidance once I arrived. I owe much of who I am to him.

Over the years my classmate and wife, Vassiliki Mangana, never gave up on me. Her patience, love and encouragement saw us through very difficult periods. I thank her for her unconditional love, honesty and steadfastness.

Our two daughters, Maye and Sarah, are the real reason I never gave up on finishing this dissertation. I thank God for blessing us with them.
# TABLE OF CONTENTS

**DEDICATION** ........................................................................................................ ii  

**ACKNOWLEDGMENTS** ......................................................................................... iii  

**LIST OF FIGURES** ................................................................................................... viii  

**CHAPTER I. INTRODUCTION** ........................................................................ 1  
  Philosophical Framework and Research Methodology .......................... 3  
  Summary of Findings and Contributions ............................................. 11  
  Notes to Chapter I ................................................................................. 14  

**CHAPTER II. A BRIEF HISTORY OF COMPUTER-SUPPORTED COOPERATIVE WORK AND DESIGN** .................................. 15  
  Computer-Supported Cooperative Work: The 1960s ......................... 15  
  Computer-Supported Cooperative Design: 1986-2001 ..................... 18  
  From Design Process and Methods to Making .................................. 23  
  Notes to Chapter II.............................................................................. 27  

**CHAPTER III. THE ROLE OF ARTIFACTS IN COLLABORATIVE DESIGN** ........................................................................................ 29  
  Definition Of Artifacts.................................................................... 29  
  Case Study ....................................................................................... 30  
  Summary of Findings...................................................................... 48  
  Notes to Chapter III ........................................................................ 50  

**CHAPTER IV. A CONCEPTUAL FRAMEWORK FOR COLLABORATIVE DESIGN** ................................................................. 52  
  Participants..................................................................................... 57  
  Chronology of Activities................................................................. 57  
  Categories of Activities................................................................. 58  
  A Protocol of Interaction in Collaborative Design ......................... 58  
  Explanation of the overall model..................................................... 59  
  The Basic Mechanisms of the Model............................................. 60  
  Discussion...................................................................................... 62
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>System Architecture and Early Experiments</td>
<td>Requirements</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Share-Kit</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Architecture of SYCODE</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYCODE Implementation</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discussion</td>
<td>72</td>
</tr>
<tr>
<td>VI</td>
<td>General Purpose vs. Domain-Specific Groupware</td>
<td>Introduction</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples of Current Groupware Applications</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WebObjects</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WebOutliner Implementation</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Asynchronous Collaboration</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Seamless Transition to Synchronous Collaboration</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Summary</td>
<td>87</td>
</tr>
<tr>
<td>VII</td>
<td>Immersive Synchronous Distributed Design Collaboration</td>
<td>Introduction</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background and Motivation</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Related Work</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Experiment 1: Distributed Design Reviews Using General-Purpose Collaboration Software</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Case Study</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recommendations</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discussion</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Experiment 2: Distributed Design Reviews Using Modified Gaming Engines</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Early Research</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coursework</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Observations</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recommendations</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discussion</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notes to Chapter VII</td>
<td>123</td>
</tr>
<tr>
<td>VIII</td>
<td>Web-Based Systems for Collaboration</td>
<td>CAMEO-Review</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implementation</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ViSTA</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implementation</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discussion</td>
<td>129</td>
</tr>
</tbody>
</table>
CHAPTER IX. CONCLUSION .................................................................................. 137
  Summary of Findings ..................................................................................... 137
  Misconceptions .............................................................................................. 140
  Conclusions and Recommendations ............................................................. 145
  Going Forward: Collaboratories, Virtual Design Studios and Access Grid Nodes ................................................................................. 148

BIBLIOGRAPHY .................................................................................................. 151
LIST OF FIGURES

Figure
3.1. Simple building envelope design............................................................. 31
3.2. Traditional and computerized modes of representation......................... 32
3.3. Compatibility with the site....................................................................... 33
3.4. The proliferation of artifacts in an architectural office......................... 34
3.5. General view of office showing two of the four project spaces. ............. 35
3.6. A framework for design work................................................................. 36
3.7. A hierarchical view of the same framework........................................... 36
3.8. Divergence and convergence while exploring alternatives.................... 39
3.9. Individual sketching within a group setting.......................................... 39
3.10. Artifact generated to support a design discussion.............................. 40
3.11. Casual interactions............................................................................... 46
3.12. Informal design discussion in the studio space................................... 47
3.13. Annotation of finalized artifacts......................................................... 48
4.1. Conceptual model of teamwork in architectural practice.................... 53
4.2. Samples of design artifacts collected at various stages of the process... 54
4.3. Samples of design artifacts collected at various stages of the process... 55
4.4. Samples of design artifacts collected at various stages of the process... 56
4.5. Top-level NIAM representation of protocol elements and roles............ 60
4.6. An example flow of Tasks, Proposals, and Artifacts............................ 61
5.1. The Share-Kit’s client-server architecture........................................... 69
5.2. The various agents in SYCODE............................................................ 71
5.3. Progression of design through divergence and convergence......... 72
5.4. A SYCODE client running on a NeXT Workstation...................... 73
5.5. A SYCODE client running on a Silicon Graphics Workstation......... 74
6.1. Microsoft NetMeeting screen shot............................................. 77
6.2. Webex.com virtual office screen shot......................................... 78
6.3. Groupboard screen shot.......................................................... 79
6.4. Apple WebObjects’ tool set. ..................................................... 79
6.5. Apple WebObjects’ 3-tier system. ............................................. 80
6.6. Initial WebOutliner screen....................................................... 82
6.7. Expanded building program..................................................... 83
6.8. Toolbar and commands........................................................... 84
6.9. Zoomed-in on a sub-item to filter out information....................... 84
6.10. Editing name and area of an element........................................ 85
6.11. The building program computes and maintains an aggregate area... 85
6.12. Synchronous collaboration on an item in the hierarchy................. 86
7.1. A Design review using an interactive SmartBoard...................... 96
7.2. A matrix of possible design review settings.............................. 97
7.3. Rear Projection SMART Board 3000i (left, photo courtesy of Smart Technologies, Inc.) and An RPS/Complete Rear-Projection System (right, photo courtesy of Draper, Inc.)...................................................... 101
7.4. Screen captures illustrating the Doom II environment............... 108
7.5. Screen captures illustrating the Quake II environment................ 110
7.6. A Screen capture illustrating the 3D world and material editors..... 113
7.7. Screen captures illustrating advanced lighting and texture mapping... 113
7.8. Screen capture illustrating the presence of other avatars............. 114
8.1. CAMEO-Review database schema.......................................... 127
8.2. Reviewers’ view......................................................................... 128
8.3. ViSTA object model and database schema ......................................... 131
8.4. ViSTA login screen ........................................................................... 131
8.5. ViSTA main page .............................................................................. 132
8.6. ViSTA preferences page ................................................................. 132
8.7. ViSTA slide search page ................................................................. 133
8.8. ViSTA slide sorter page ................................................................. 133
8.9. ViSTA slide show page ................................................................. 134
CHAPTER I
INTRODUCTION

The development of appropriate research frameworks and guidelines for the construction of software aids in the area of architectural design can lead to a better understanding of designing and computer support for designing (Gero and Maher 1997). The field of research and development in computer-supported collaborative architectural design reflects that of the early period in the development of the field of computer-supported cooperative work (CSCW). In the early 1990s, the field of CSCW relied on unsystematic attempts to generate software that increases the productivity of people working together (Robinson 1992). Furthermore, a shift is taking place by which researchers in the field of architecture are increasingly becoming consumers of rather than innovators of technology (Gero and Maher . In particular, the field of architecture is rapidly becoming dependent on commercial software implementations that are slow to respond to new research or to user demands. Additionally, these commercial systems force a particular view of the domain they serve and as such might hinder rather than help its development. The aim of this dissertation is to provide information to architects and others to help them build their own tools or, at a minimum, be critical of commercial solutions.

Researchers employ conceptual frameworks to analyze the nature of design and the processes of collaboration, identify opportunities for intervention, and provide patterns and guidelines for the construction of software aids. The findings in this dissertation do not advocate a specific solution or a specific computer system. Rather, they constitute a set of experiences and derived recommendations and guidelines, that,
when taken as whole, serve as a framework for others to examine, reconfigure, extend, and build on. The term *framework* is preferred rather than *foundation* because the former implies more flexibility and open-endedness than the latter.

Pioneers in this field, starting in the 1960s have proposed several such frameworks. In most cases, these frameworks were based on theoretical and speculative work that was largely not founded on any actual implementations. Despite the fact that these frameworks predate contemporary advances in networking, the literature review reported in this dissertation has uncovered the fact that they have biased the development of collaborative systems for the last forty years and still influence research to this day. For example, most of the current research work in this area focuses heavily on the synchronous aspects of collaborative design. As reported in the literature review, this more recent focus started in 1993 with the publication of three papers in the *CAAD Futures* conference all addressing the issues of synchronous design.

The assumptions regarding the conceptual framework of collaborative design systems is challenged through several methods. First, a critical review of the work done since 1962 is undertaken. Second, an illustrative case study is conducted in which designers were observed. Third, conceptual constructs are designed based on the literature review, experience and the case study. Fourth, several software applications and experiments are devised to analyze conceptual constructs as well as uncover new issues. These constructs and experiments act as case studies that uncover issues such as the role of artifacts in collaboration, competition, team hierarchy, access privileges, social awareness and information filtering.

Issues of design and collaboration in the field of architecture are the focus of this dissertation. Design is considered connected to and not fundamentally different from other collaborative activities. Thus, this dissertation includes discussions of collaborative activities that are not directly interpretable as design activities. Yet, a case is made that these activities include design albeit in an indirect manner. Furthermore, a second case is
made that lessons learned from these activities have a direct use in collaborative design. Collaborative activity was analyzed in an architectural studio within professional practice, the design studio, the classroom, and the lecture hall within schools of architecture. Furthermore, computer-based systems were deployed in support of co-located and distributed collaborative activities of academic organizations and professionals – again in the realm of architecture.

The topic of this dissertation is part of the general field of computer-supported collaborative design (CSCD) as it applied to the field of architecture. Simply put, CSCD is the study of how computers and technology in general can support the various aspects of collaborative design. Design activity is said to be collaborative when it includes more than one participant. The field of CSCD is derivative from the field of computer-supported cooperative work (CSCW). The terms cooperative and collaborative are used interchangeably in this dissertation in contrast to their use by other researchers. Elsewhere, you will find distinctions being made between cooperation and collaboration as follows: cooperation is used to signify that multiple participants are working on the same topic or problem. No indication should be derived by the term that the participants are working in a positive and helpful mode. The term is meant to be more neutral. Collaboration, on the other hand, is sometimes used to signify a positive mode of work and a teamwork effort to solving problems. In this dissertation, both terms are used in their more neutral and original etymological meaning: operating, laboring, or toiling with others (as in co-operating and co-laboring).

**Philosophical Framework and Research Methodology**

Architecture is traditionally viewed as a field that interpolates art and science. Within architecture, the discipline of design studies has resisted many attempts to define it as an integrative science with related and accepted research methodologies (Margolin
Therefore, research in the area of design relies on a diverse set of methodologies and philosophical frameworks to help investigate and gain insight into its processes, products, contexts, history and theories. This dissertation employs several research methodologies that were deemed appropriate for the topic of the investigation. In general, however, a purely scientific methodology that includes analysis, synthesis and evaluation has been rejected. Instead, a phenomenological approach is advocated that places the phenomenon of design within the context of everyday life and blurs the line between the subjective and the objective.

**Phenomenology as a philosophical framework and a research method for the study of collaborative design, artifacts, and computers**

The phenomenon of design is studied through the observation of and participation in the everyday activities of designers. In an attempt to go beyond the description of the situation, fundamental issues of design work were uncovered guided by a phenomenological research method and philosophy as put forward by Martin Heidegger. Therefore, phenomenology, which was taken here as a departure point for the study of collaborative design, artifacts, and computers, is introduced and the basic issues that relate to design are discussed. Yet, it is important to note here that this discussion is not a comprehensive overview of the issues and ideas of phenomenology. Rather, only ideas that are relevant to the research topic are extracted and discussed.

Understanding the nature and structure of *Being* was Heidegger’s topic of inquiry in his seminal work *Being and Time*. Rather than starting from a purely theoretical point, however, he was convinced that understanding *Being* can only start by understanding the phenomena of *everydayness* (what humans do in the world on a daily basis). Although the term phenomenology etymologically means the study of phenomena, Heidegger defines it differently:
… to let that which shows itself be seen from itself in the very way in which it shows itself from itself.... The term ‘phenomenology’ is quite different in its meaning from expressions such as ‘theology’ and the like. Those terms designate the objects of their respective sciences according to the subject-matter which they comprise at the time ... ‘Phenomenology’ neither designates the object of its researches, nor characterizes the subject-matter thus comprised.¹

He goes on to discuss the nature of the phenomenon: “Manifestly, it is something that proximally and for the most part does not show itself at all: it is something that lies hidden, in contrast to that which is something that belongs to what thus shows itself, and it belongs to it so essentially as to constitute its meaning and its ground.”² Thus, contrary to the common understanding of the word, the phenomenon is not what one observes initially, it is, rather, something that lies behind that which appears. Nevertheless, this appearance is so closely related to the phenomenon that it has to be the starting point of inquiry into the phenomenon. Heidegger then explains three ways in which a phenomenon can be covered up:

There are various ways in which phenomena can be covered up. In the first place, a phenomenon can be covered up in the sense that it is still quite undiscovered. It is neither known nor unknown … Moreover, a phenomenon can be buried over … This means that it has at some time been discovered but has deteriorated to the point of getting covered up again …

The third way in which a phenomenon can be covered up is through disguise which happens when:

… what has been discovered earlier may still be visible, though only as a semblance … This covering-up as a ‘disguising’ is both the most frequent and the most dangerous, for here the possibilities of deceiving and misleading are especially stubborn.³

Using this basic philosophy, a participant/observer research methodology was employed to analyze design-related phenomena and understand the ways in which designers interact with others, their settings, tools, and artifacts. However, this analysis
does not comprise a complete phenomenological interpretation. Rather, it constitutes an initial attempt to describe certain aspects of design and synthesize them in such a way that some insights into its nature appear.

The definition of design in this dissertation bares a lot of resemblance to that of Winograd and Flores’s which is in turn derived from Heidegger’s writings. Bödker et al. summarize it as follows:

... the point of departure in design is that the different participants understand the situation they come from. They are used to act in situations of ‘normal resolution’. This goes for users as well as designers. The normal resolution or understanding includes the blindness created by the tradition they come from. The design process is characterized by a breakdown of this understanding, by which a situation of irresolution is created. Design is resolving these situations of irresolution, based on commitments between the participants. This is neither objective problem solving nor rationalistic decision making. It is concerned human activity; where different traditions and backgrounds meet.4

This understanding of design has at its core the concept that design is a concerned collaborative (but sometimes competitive) effort involving multiple participants that bring to bear their traditions, tools and artifacts. The view of the ways in which people interact with tools and artifacts is again derived from Heidegger’s writings on spatiality and space. Given the opaqueness of Heidegger’s writings, their explanation as offered by H. Dreyfus’s in his book Being-in-the-World (Dreyfus 1991) is relied on:

Dis-stance … makes it possible to encounter degrees of nearness and remoteness, accessibility and inaccessibility. Once an object has been brought into the referential nexus, dis-stanced, it can be more or less available, i.e., more or less distant from particular individuals, more or less integrated into each individual’s activities. The degree of availability is the nearness of concern … 5

This human concern for the world (or, as Dreyfus puts it, Dasein’s concernful being-in-the world) is what initiates the spatial interactions with the available artifacts
that occur in this world. According to Heidegger this is possible because Dasein (which can be thought of as the human being-in-the-world) is spatial:

To encounter the available in its environmental space remains ontically possible only because Dasein itself is ‘spatial’ with regard to its being-in-the-world. . . . Dasein is ‘in’ the world in the sense that it deals with beings encountered within-the-world, and does so concernfully and with familiarity. So, if spatiality belongs to it in any way, that is possible only because of this Being-in.6

The concepts of spatiality and dis-stance, as described above, thus, relate directly to the ways in which designers immerse themselves in a world of design concerns. The activities that take place in these worlds consist of interactions with artifacts, tools and other individuals based on their degree of availability or remoteness. Design, therefore, can be alternatively defined as the consequence of a series of stances that the designers take towards artifacts, tools, and others.

Heidegger’s writings also inform us of the different modes in which we cope with everyday experiences. These modes of being are best summarized by Coyne and Snodgrass. According to them, Heidegger’s ontology can be understood in terms of four modes of being. The first mode is that of the available (or the ready-to-hand). At this level of being we are unreflective in our interactions with the world and artifacts around us:

According to Heidegger our primary experience of the world is undifferentiated. We are absorbed. As we engage in our activities things are simply available. … The well known example of this experience of the available is our use of an item of equipment such as a hammer … The most primordial experience we have of a hammer is through its ‘readiness-to-hand.’ Readiness-to-hand is not grasped theoretically or by looking at and contemplating the hammer, but by unselfconsciously using it.7

The second level of being is that of the unavailable or the meaningful. At this level we experience some form of breakdown that suddenly allows us the opportunity to gain an understanding of our environment. When the situation does not meet our
expectations we become aware of the factors influencing it and thus it becomes
meaningful.

The third level in Heidegger’s ontology is that of detached theoretical
understanding:

At the next level in Heidegger’s ontology we encounter the world of
detached theoretical understanding. (Heidegger calls this the ‘occurrent.’) This is where we stand before something in a detached manner, engaging in theoretical reflection. It is also the realm of the object in science, the object of observation and experimentation.⁸

The last level is that of bare facts. According to Heidegger this mode of being occurs in the realm of the subject, detached from the world and the phenomena:

The next level in Heidegger’s ontology is that of bare facts, sense data, the most rarefied and abstract understanding possible. (This is the ‘pure occurrent.’) This is the realm of the self-sufficient subject engaged in pure contemplation or perhaps undirected curiosity. … These four levels of the available, the meaningful, detached theoretical reflection and bare facts are the important components of Heidegger’s ontology … [This] philosophy involves us in a fundamental ‘reversal’ of the understanding provided through the Cartesian ontology.⁹

Thus, looking at collaborative design through these modes of being will allow us to reach an understanding of how designers cope with their everydayness and an understanding of how new equipment will fit in their world of tools. Any future interventions can be better designed if these modes of being are taken into account. Future computer tools designed in accordance with these modes of being may solve some of the problems that current tools face. The way to find out whether this is actually the case is not through abstract argumentation, but through the building of such tools (or prototypes). This constructive, user-centered style of design is crucial for the success of these tools especially that computer-assisted collaborative design is not yet part of the everyday experience of designers.
Computer interfaces provide us with a clear example of the relevance that the above ‘modes of being’ present for the study of design work. By standing between the user and the task to be accomplished, interfaces constitute a tool (equipment) with the potential of breaking down the unreflective process of work. Incompleteness or problematic operation of interfaces are common examples of such breakdowns. According to Heidegger’s philosophy, however, breakdowns may be necessary or even desirable since they allow further reflection and thus ‘understanding’ of the particular equipment in use or the task to be accomplished. On the other hand, they can also serve purely pragmatic purposes: interface features, for instance, that protect novice users from accidentally destroying documents (such as warning and confirmation dialogue panels) are both desirable and necessary to alert the user and ensure the safety of the work.

Contrary to first impressions, then, breakdown is not necessarily a negative attribute; it only needs to be accounted for. Computer tools, therefore, should be designed to anticipate breakdown as much as they are designed for normal operation. Care and attention need to be paid to how a system behaves when it breaks down and how the user(s) will reflect on that situation. These principles will become major issues in the development of collaborative software because breakdowns will be prevalent due to multiple users participating in the use of the system. Each user will bring in a view of the world that might conflict with other views leading to situations of irresolution. In an ideal case, the collaborative system should fade into the background allowing the users to interact as directly as possible with the task at hand and with other users. Unfortunately, this will not always be the case and thus a need exists to design it in ways to manage breakdown.
A Constructivist Approach

Phenomenology provides a philosophy and a method that uncovers the fundamental ways in which humans cope and interact with their daily practices. The case study reported in this dissertation illustrates that design is one of those daily practices and not a mysterious activity that occurs detached from all other activities. Thus, using a phenomenological approach both as a research method and as a philosophical framework allowed a deeper understanding of the phenomena of design.

As mentioned above, the second focus of this dissertation is the analysis of design and collaboration through the construction of computer aids and prototypes. This research methodology is based on a constructivist approach that views human beings as “oriented actively toward a meaningful understanding of the world in which they live” (Neimeyer and Neimeyer, 1993). Thus, one of the research methodologies that is applied in this dissertation consists of an analysis of a situation that usually contains participants and artifacts (humans and objects) and a set of relationships among them. This analysis is followed by mapping the found situation into a set of entities, relationships and processes that are then encoded as computer-based object-oriented components. These components are then activated through software and deployed. Computer aids and prototypes allow for a construction of a set of representations and algorithms that attempt to model and understand a situation. What is asserted in this dissertation is that insights into the nature of design and collaboration can be gained by the construction and analysis of such mappings regardless of the efficacy of the constructed prototype. Once constructed, some of the data and methods may be deployed such that evaluation and further refinement can take place. While the prototypes were designed with care to address problems and provide solutions, the purpose of building them was not to compete with commercial counterparts. A comparison to other products is considered irrelevant because the essential research purpose of constructing them was not to judge how well they serve
their task as compared to other solutions, but to use them as analytical devices to uncover and discuss relevant issues and derive lessons that others can learn from. This dissertation includes experiments that have succeeded and others that have failed. Both types contribute to the knowledge in the field of architecture.

Several prototypes have been built during the course of this dissertation. Some of which remained laboratory experiments, others have been widely deployed in real world settings. The degree of deployment and evaluation, however, is of little relevance to the research issues they uncovered. Additionally, these prototypes sometimes required specialized software programming and at other times the assemblage of readily-available commercial solutions. Both approaches were found useful in the analysis of design and collaboration.

**Summary of Findings and Contributions**

Simplistically, design activity can be regarded as the creation of artifacts and as such needing computer-based support in order to facilitate their co-creation, and sharing. In part, this holds true. Through direct observations of architects at work, a review of the literature on the nature of design, and the construction of computer aids to design activities, design was found to be a complex and multi-faceted activity. Thus, this dissertation refutes the traditional notion that one integrative system can efficiently and usefully support design activity. In contrast, this dissertation introduces various aspects of design, architectural education and practice, analyzes and emphasizes the role of artifacts in collaborative design, and provides guidelines for implementing computer-supported collaborative systems. The common thread that ties all the various research activities described here is a concern for fundamentally understanding collaboration and seamlessly supporting it through suitable technologies.
Chapter II includes a brief history of computer-supported cooperative work and design that relate to the field of architecture and highlights the seminal works in these fields. Additionally, it critically traces some of the ideas and misconceptions that have filtered to current thinking about these topics.

Chapter III examines the role of artifacts (drawings, models etc.) in supporting collaborative design. In this chapter a participant-observer case study in an architectural firm is presented to illustrate how real collaborations currently function and define opportunities for technology interventions.

Chapter IV uses the work presented in the case study to propose a conceptual framework for collaborative design. The proposed framework is specific enough to be used by others as a foundation for an object-oriented system that reasons about and supports the various entities involved in design collaboration.

Chapter V further develops the conceptual framework and introduces the first implementation. Through this experiment, an argument is made that collaboration can take place only after a common protocol of interaction (i.e. a shared language) is developed.

Chapter VI reports on the first experiment using web technologies. The worldwide web has rapidly exploded and several commercial entities are offering general purpose tools for online collaboration such as text messaging, shared whiteboards and video conferencing. In this chapter a case is made that collaborative design requires domain-specific tools rather than general purpose ones.

Chapter VII describes two systems that use immersive synchronous techniques to supported collaboration. The first system is an assemblage of several off-the shelf commercial products. The second system uses modified gaming engines and deploys them in service of immersive design presentations. This chapter illustrates the potential of and limitations of synchronous commercial systems and provides lessons to others who wish to assemble and use ready-made solutions rather than develop custom ones.
Chapter VIII discusses two web-based collaborative systems that have been used outside the design studio. The first system was used to manage digital assets and control access to them while the other system was used to review proposals based on the scientific blind-review standard. While those two systems are not directly concerned with design, they have provided important insights and challenged several assumptions regarding the nature of collaboration. A case is made that the conceptual framework developed in Chapter IV was essential in the design of these systems. Furthermore, a second case is made that these systems contribute important collaboration patterns and lessons that can be used in support of collaborative design.

The conclusion of the dissertation summarizes the key findings of and outlines the misconceptions regarding collaboration. It then proceeds to make several recommendations. The dissertation concludes with a discussion of the need to develop a far more connected design studio environment using technologies already developed in collaborative research laboratories (collaboratories), virtual design studios, and access grid nodes. The dissertation ends with plans for future work in the area of computer-supported collaborative design.
Notes to Chapter I


2 Ibid., 59.

3 Ibid., 60.


7 Richard Coyne and Adrian Snodgrass, *What is the philosophical Basis of AI in Design?*, (Sydney, Australia: University of Sydney, 1991), 7 (Working Paper).

8 Ibid., 7.

9 Ibid., 8.
CHAPTER II
A BRIEF HISTORY OF COMPUTER-SUPPORTED COOPERATIVE WORK AND DESIGN

To understand the current framing of computer-supported cooperative design as a set of technologies that enable geographically dispersed users to work collectively on a common problem, one must trace the evolution of the field to the original ideas advocated by its pioneers. Unfortunately, since much of the current work derives its paradigm from those early ideas, many of the misconceptions of the early work still find their way into current thinking. Thus, the following section traces as closely as possible the evolution of the field starting in the 1960s and ending in the year 2001. While doing so, the important foundational ideas that were put forward will be highlighted and critically examined. The period 1970-1985 is omitted because it contained little work that concerned itself with the topic at hand. This review also excludes the exploding body of work since 1995 in keeping with the relatively limited scope of this dissertation. Much of the work since then, while significant, builds on a solid foundation of ideas founded between 1986 and 1995. However, this dissertation includes a bibliographical list that will enable the reader to pursue and fully investigate that body of work independently.

Computer-Supported Cooperative Work: The 1960s

One can trace the beginnings of the field now known as computer-supported cooperative work (CSCW) to the work of Douglas Engelbart in the 1960s. Specifically, to his seminal paper *Augmenting Human Intellect: A Conceptual Framework* which was published as a research report to the Air Force Office of Scientific Research in October
1962 (Engelbart 1962). In this report Engelbart describes how computers can help a team solve problems and make decisions more efficiently. In his introduction, he outlines some of the disciplines, including design that might benefit from technology augmentation. However, he immediately rejects the idea of developing domain-specific solutions to solve a domain-specific problem. He derides these solutions as *isolated clever tricks*.

Later in his report, in a section titled *Team Cooperation*, he outlines his vision of how the system will work:

> Let me mention another bonus feature that wasn’t easily foreseen. We have experimented with having several people work together from working stations that can provide intercommunication via their computer or computers. That is, each person is equipped as I am here, with free access to the common working structures. There proves to be a really phenomenal boost in group effectiveness over any previous form of cooperation we have experienced. They can all work on the same symbol structure, wherever they might wish. If any two want to work simultaneously on the same material, they simply duplicate and each starts reshaping his version -- and later it is easy to merge their contributions. The whole team can join forces at a moment’s notice to ‘pull together’ on some stubborn little problem, or to make a group decision. Most points of contention are resolved quite naturally, over a period of time, as the developing structure of argument bears out one, or the other, or neither stand.

A year later, at the 1963 Spring Joint Computer Conference held in Detroit, Michigan, Steven Coons for the first time writes about the need for a computer-aided design system that enables synchronous interaction with multiple users (Coons 1963). It is this vision that laid the foundation for what is currently called the field of *Computer-Supported Cooperative Design* (CSCD):

> The Computer-Aided Design System should be capable of carrying on conversations with, and performing computations for several designers at several consoles substantially all at once. In this way each designer can be immediately aware of what the other designers are doing, and thus avoid one of the truly severe problems of intercommunication that designers face today.
Thus, instead of being the result of a recent realization that design proceeds by teamwork, the focus on collaboration is due to recent technological advances, both on the hardware and software sides, which, coupled with a maturity in the knowledge-base of group behavior and dynamics, have made it possible to support and enhance collaborative processes through technological interventions.

In the area of computer-supported collaborative design one must also note the importance of the work of Ivan Sutherland (Sutherland 1963). While studying at the Massachusetts Institute of Technology, Ivan Sutherland invented the first interactive computer graphics system (called Sketchpad). In his Ph.D. thesis, Sutherland used a lightpen to create drawings directly on a nine inch cathode ray tube (CRT). Sutherland’s work is the basis of all modern graphical user interfaces and is echoed in the recent introduction of the Tablet PC concept which allows a user to sketch on and interact directly with the screen of a portable computer. More relevant, however, current synchronous systems rely on Sutherland’s ideas of direct sketching to enhance collaboration. As illustrated in Chapter VII, synchronous collaborative systems that support design and graphics usually rely on the direct creation and transmission of human-scaled drawings to provide fuller immersion and a more natural interface.

These pioneering visions of CSCW and CSCD set the tone of research for the next forty years. However, given that Engelbart and Coons dealt mostly with theoretical situations and visionary solutions, their intuitive response was to specify a utopian and unified system. That is, in their writings, they viewed the problems of cooperative work and design as a congruent set that can be addressed with one system and one approach. Engelbart mistakenly believes that merging different solutions is an easy task and resolving conflicts can occur naturally. Similarly, Coons simplistically believes that synchronicity and social-awareness are the only needed features in addressing the problems that designers face. As noted below, the nature of cooperative design and the needed technologies to support its many aspects is far from simple. While the ideas of
 Engelbart and Coons were visionary, the work since then has evolved to encompass other aspects of cooperative design.

**Computer-Supported Cooperative Design: 1986-2001**

Most of the computer-aided design research efforts in the 1970s and early 1980s concentrated on the problems of automating the design process as well as the geometric and conceptual modeling and visualization of buildings (Jones 1970; Mitchell 1977; Turner 1988; Stiny 1989). The implications of technological augmentation to cooperative design had to wait until the technology caught up in the second half of the 1980s with the advent of hypermedia, the maturity of computer networks, and the invention of the internet. The CSCW field became widely known to researchers with the first conference on the subject in 1986 held in Austin, Texas. The definition of cooperative design as a type of cooperative work came two years later at the CSCW 88 conference held in Portland, Oregon. At that conference Susan Bødker, Pelle Ehn and others published a paper titled *Computer Support for Cooperative Design* in which they theoretically defined cooperative design as a type of cooperative work (Bødker et al. 1988). However, in contradiction to Englebart’s unified vision of computer augmentation to group work, they warned in their paper against the generalization of solutions in one domain of cooperative work to another. It is after the publication of this paper that researchers, especially in the field of architectural design, began looking at computer-supported cooperative design as closely related to, but distinct from the field of computer-supported cooperative work.

In 1990, William J. Mitchell coins the intriguing term *society of design* in the last chapter of the book *The Electronic Design Studio* (Mitchell, McCullough and Purcell 1990). The term, inspired by Marvin Minsky’s 1985 *society of mind*, hints at Mitchell’s thinking about the convergence of CAD and network technologies to provide users with
various modes of interactions and exchange of artifacts. He does not write specifically about design collaboration in that chapter; however he sets the stage for his later work in that area. In 1991, Carl Tollander publishes a chapter titled “Collaborative Engines for Multiparticipant Cyberspaces” in Michael Benedict’s book *Cyberspace: First Steps* (Benedikt 1991). In this chapter Tollander describes a collaborative cyberspace as metric space influenced by the actions of multiple participants. He defines collaboration as a “set of operations over a set of cyberspaces, in which a group of selective influences under the control of the participants direct the evolution of space.”

Meanwhile, those studying the nature of the design process and the practice of architecture have started to dispel the myth of the genius architect and to give group work its due credit. In her book (Cuff 1992) Dana Cuff notes: “good buildings are not designed by committee. Instead, excellent projects are designed by a few leaders, who, working together, are able to move the project along and coordinate the group of contributors.” She later also notes that “one of the first items of propaganda that must be challenged is the primacy of the independent practitioner working with relative autonomy. This myth … is not unique to architecture but is embedded in the cultural system of all professions.”

Cuff identifies, through the above comments, an emerging attitude within the field of architecture: the practice of architecture does not fit a collaborative ideal where all contributors play an equal role, but excellence in design does depend on a hierarchical collaborative process that involves many participants, each playing a different role.

In the same year, Jerzy Wojtowicz, James Davidson and William Mitchell co-author a paper in the proceedings of the Association for Computer-Aided Design in Architecture (ACADIA) titled *Design as Digital Correspondence* (Wojtowicz, Davidson and Mitchell 1992). It is the first paper in an ACADIA conference that discusses the potential of computers as aids for collaborative design. The paper discusses a case study in which students in two studios at institutions a continent apart were asked to co-design
a project using a shared electronic bulletin board to post design artifacts (using the FTP protocol), review and exchange comments using electronic mail and speakerphone, and conduct an electronic, long distance design review. As evident in the title and the case study, the authors believed that cooperative design can be conducted in a manner similar to holding a conversation with back and forth communication. While they did include synchronous collaboration in their project, the emphasis was clearly on asynchronous contributions to a common repository of information. This may have been due to the difference in time zones as well as the relative immaturity and lack of availability of real-time synchronous software and corresponding infrastructure.

The convergence of computer-aided design and the study of teamwork starts to take place by the late 1980s and early 1990s. Bryan Lawson, in the second edition of his book *How Designers Think*, comes tantalizingly close to defining the then nascent field of computer-supported cooperative design (Lawson 1990). The last two chapters of his book are titled *Designing with Others* and *Designing with Computers* respectively – and although he does not fully extrapolate the fact that designing with computers might enable designing with others, one cannot help but assume that he was starting to form that connection.

Starting in 1993, the field of computer-supported cooperative design begins to dramatically expand. As an example, the CAAD Futures ‘93 conference included a special section on “Cooperative Design” in which three papers were published (Maher, Gero and Saad 1993; van Bakergem and Obata 1993; Bhat, Gauchel and Van Wyk 1993). Answering the call of Engelbart and Coons, the authors of the first paper (Maher, Gero, and Saad 1993) focused on the use of synchronous collaboration tools and representations that can support multiple interpretations. The authors of the second paper (van Bakergem and Obata 1993) focused on the synchronous aspects of design as well, but called for project-specific information to be included in any collaborative system. The authors of the third paper (Bhat, Gauchel and Van Wyk 1993) focused on issues of communication
(both synchronous and asynchronous) and reached the conclusion that communication frequency and volume are scalable as the number of clients and the complexity of the design expand.

Conferences in Europe, especially those addressing Design Decision Support Systems and Cybernetics begin to focus on Collaborative Design (Jabi and Hall 1995a; Jabi 1996a; Jabi 1996b). In Asia, the CAAD Futures 1995 conference in Singapore included two separate sessions on cooperative design with a total of fourteen papers such as (Jabi and Hall 1995b). In North America, ACADIA also witnessed a similar expansion in papers submitted on the topic. Since the ACADIA 1995 conference, computer-supported cooperative design has been formally identified as a research topic that is solicited in the call for papers. The 1999 conference included nine papers on the topic; a peak number for ACADIA (Ataman and Bermudez 1999). The 2000 proceedings included five papers while the 2001 proceedings included four papers (Clayton and Vasquez de Velasco 2000; Jabi 2001).

Currently, the number of published papers has expanded to the point where it is difficult to track and document. However, the field still has very few books dedicated to the subject. In 1995, Jerzy Wojtowicz edited a concise manuscript titled *Virtual Design Studio* that included essays reflecting on the topic of conducting design studios at a distance (Wojtowicz 1995). The manuscript also included documentation of the collaborative Virtual Village Studio Project (VVS) conducted by Harvard University, University of British Columbia, MIT, and the University of Hong Kong. The project to date remains as one of the largest prototypical implementations of cooperative design studios. While not directly addressing computer-supported cooperative design issues, William Mitchell’s book *City of Bits*, published in 1996 in both print and online, is considered a seminal work on the nature and the potential of an interconnected world where processes that used to require real space are replaced with counterparts in a virtual space (Mitchell 1996). Many of the ideas and warnings he includes in his book can be
applied to the design of computer aids and collaborative environments. Four years later, Mitchell published a sequel titled *E-Topia* in which he turns his attention to the evolution of the city under the influence of the various technologies especially e-commerce (Mitchell 2000).

In 2000, Stephen Scrivener, Linden Ball and Andrée Woodcock edited and published the proceedings of the *CoDesign* 2000 conference held at Coventry University (Scrivener, Ball and Woodcock 2000) that included fifty papers on the topic. The book is an indication of the maturity of the field and is a good survey of the various sub-topics and themes currently being pursued in this area of research.

In 2001, Martijn Stellingwerff and Johan Verbeke edited and published the proceedings of the *ACCOLADE* workshop held in Delft, the Netherlands (Stellingwerff and Verbeke 2001). The *ACCOLADE* workshop started with the premise that design offices are in need of multi-disciplinary and multi-located virtual collaborative architectural design methods. The participants affirmed that the architectural design process is non-linear, multi-disciplinary, and highly creative. Thus, it is in need of sophisticated systems that enable representation, manipulation, and communication of three dimensional data. The introductory text echoes almost exactly the writings of Coons forty years earlier: “In order to collaborate during an architectural design process there is a need for a system, which provides for synchronous work sessions of multiple design parties.” The Workshop papers included topics dealing with human-computer interaction, 3D online learning environments, the use of gaming in collaborative design, web-based environments for collaboration, and interface design issues.
From Design Process and Methods to Making

Many design methodologists, over the years, have described design as a process consisting of definite steps or stages. The goal behind such descriptions is to draw a picture of design as a rational, systematic and deterministic process hoping first to externalize the methods and rationale used by designers in order to emulate or enhance them\(^4\) and second to define a “science of design”\(^5\,6\). Most of these methods, in a gross simplification, divide the design process into three stages: analysis, synthesis and evaluation. The weaknesses of this abstraction are counter-acted by an extensive use of return loops to preceding steps which is due to a realization that the design process is not as linear or simple as first thought. What is put into question, then, is the degree of insight into the design process gained by describing it as a mere sequence of steps. Lawson provides a plausible reason for the inherent weakness of these design process charts: “They [design process charts] seem to have been derived more by thinking about design than by experimentally observing it, and characteristically they are logical and systematic.”\(^7\)

While the desire to account for design in a scientific way seems to underlie the above categorizations, their rejection had become evident even at the time when systematic design methods were popular. As G. Broadbent notes, “Systematic design methods, so far, have tended to complexity and abstraction to such an extent that few practicing architects believe that they have much validity in the ‘real’ world.”\(^8\) The reason for such rejection, he continues, is that “… up to now, there has been a tendency in developing ‘systematic’ design methods, to take techniques which happen to be available, and to force these onto design, without questioning their actual relevance.”\(^9\)

Although the description of architectural design as a sequence of steps still re-emerges, slightly modified, this rationalistic model is now being regarded as a misguided approach to understanding design.\(^10\) Researchers such as Richard Coyne and Adrian
Snodgrass, Donald Schön, N. John Habraken and Mark Gross, for instance, argue for alternative design theories and methods in which the understanding of design’s nature stems from the return to the everydayness of the designer. By utilizing the hermeneutics of Gadamer and Heidegger and the pragmatism of Dewey, Coyne and Snodgrass have argued for a method that “represents a paring away of esoteric and abstract theories and models used to explain everyday phenomena, such as language and thought.”

According to this approach design is viewed as a phenomenon of human praxis that can be explicated through interpretation: “The approach adopted in unfolding this argument is to follow the hermeneutical account of how understanding in any area of endeavour is acquired.” Similarly, Schön’s research found that the patterns of reasoning used by designers while designing do not “significantly differ from reasoning in everyday life.”

In addition, he introduces the triad of rules, types and design worlds which he believes form the vehicle to understand “the designer’s ways of knowing.” The idea of design worlds, he continues, is:

inconsistent with an objectivist point of view, according to which things are what they are independent of our ways of seeing them … But design worlds are consistent with a constructionist perspective … where miscommunication, novelty, and diversity of approach are exactly what we would expect.

In an attempt to explicate the complexity of design, Habraken and Gross employ games as metaphors to designing. They seek to provide a more profound understanding of design, but due to their methodology, they are also able to introduce the issue of collaboration into their inquiry (Habraken and Gross 1988):

When designing involves the interaction of many actors, all involved with transforming an artifact, we see similarities with board games … One of the most difficult aspects of understanding designing has always been that too many divergent acts occur simultaneously, defying simple description. We found it useful to develop a set of games that each isolates and focuses a single aspect, each giving a clearer picture of what just some of designing is about.
According to the above views, then, design should be approached as a collaborative activity taking place in various settings. As Donald Schön has noted, design should be treated “not primarily as a form of ‘problem solving’, ‘information processing’, or ‘search’, but as a kind of making.” Yet, a more profound understanding of this notion of design as making could be reached through further exploration of the context in which it occurs; that is the design settings. As the primary environments in which designers bring to bear their private and shared design worlds, design settings not only enable communication, but also promote the co-operative resolution of a design situation.

Ömer Akin views architecture as a direct effect of a dialogue between representation and the idea (Akin 1982). He writes:

Architecture, then, rises from a representation of an architectural idea that in turn results from the very special and painstaking interaction between representations of the designer’s mind and representations on paper, in books, in computers, and/or in physical models. Representation allows thought and design to take place.

If Akin’s comments are extended to include collaboration, one can discern the fundamental dialectic and collaborative nature of what is called designing: a concerned social activity that proceeds by creating architectural elements to address a set of requirements and their re-thinking as a result of architectural conjecture.

The history of computer-aided design seems to have completed a circle of evolution starting with the rejection of the manual and intuitive practice of design, passing through a phase that favored rational and systematic design, and ending at a recognition that design using computers is perhaps a craft that is closer than one may think to those manual methods rejected by the early pioneers of CAD (McCullough 1996). Thus, describing design as a sequence of steps is no longer sufficient to understand the complexity of the social interactions that take place in it. Rather, this evolution and eventual view of digital design as craft naturally leads to the hypothesis that artifacts play a central role in enabling the collaborative process. In this
dissertation, starting from the viewpoints offered in the previous examples, design is viewed not primarily as a process, but as a multi-participant activity that takes place in various settings. Settings are environments in which one or more designers bring to bear their private and shared design worlds, types and rules by employing artifacts as the primary vehicle through which they build and share design ideas. What is of more interest here are those settings in which designers engage in activities and interactions that aim at the co-operative resolution of a design situation. This hypothesis will be examined in the next chapter.
Notes to Chapter II


9 Ibid.


11 Ibid., 125.

12 Ibid.


14 Ibid., 182.
15 Ibid., 183.


19 Artifacts are defined more precisely later on; for now the reader can think of such things as sketches, models, and working drawings as typical examples.
CHAPTER III
THE ROLE OF ARTIFACTS IN COLLABORATIVE DESIGN

When viewed in relation to collaborative design, artifacts acquire an important role. Since they constitute the embodiment (representation) of design ideas and actions, they become a major means through which communication is achieved. Rather than constituting mere information holders, whose importance has been acknowledged, but also underestimated, artifacts are at the heart of designing.

Beginning with the premise that artifacts are important to design, the general purpose of this chapter is to describe their importance and roles, specifically in a collaborative design setting. Such a description leading to a better understanding of the intrinsic characteristics of design can subsequently facilitate the development of base-line knowledge and guidelines for the future development of computerized tools that support groups of designers working together in teams. It is hoped that, having this knowledge, these tools will be more compatible with both the nature of design activity and its related artifacts.

Definition Of Artifacts

Webster defines artifact as: “a product (as a structure on a prepared microscope slide) of artificial character due to extraneous (as human) agency.”¹ Although this definition includes all the tools and equipment that humans produce, of interest are artifacts that are used as representations of knowledge by recording it in some physical form. Other researchers have included gestures and ‘airborne artifacts’² in their
definitions. However, in this chapter, the study is limited to longer-lasting physical artifacts even though the importance of studying ephemeral artifacts is acknowledged.

**Case Study**

The following section describes a case-study in which a participant-observer method was used to analyze the nature of design through the everyday activities of designers in practice. Focusing on the actions of designers as they unfold in their natural setting, the architectural office, allows for a kind of knowing that is not available otherwise. This knowledge, as Schön writes, can be derived not through technical rationality, but through the actions of the designer: “It seems right to say that our knowing is in our action. And similarly, the workday life of the professional practitioner reveals, in its recognitions, judgments and skills, a pattern of tacit knowing-in-action.”

While the professional setting of the architectural office seems to provide adequate information for a case study, the academic setting of the design studio could be viewed as an equally valid alternative. Yet, even Schön, who used the design studio as a source of insight into designing, states that the studio overlooks some aspects of architectural practice. He writes:

What students and studio masters do in the studio is in some ways unlike the reality of architectural practice. It tends, for example, to minimize the laborious rendering of technical detail on which many young practitioners spend a great deal of their time. It tends to leave out such important processes as dealing with building code inspectors and interest groups. Few studios pay attention to breakdowns of equipment or work-stoppages on the site.

Thus, due to the richness and complexity of design activities in a professional setting that could subsequently lead to a comprehensive understanding of the nature of
design, an architectural firm was selected for this case study. The essentially collaborative nature of design in an office, as contrasted to the often individualistic work in the studio was another reason for this selection. Finally, since the ultimate aim of this project is to provide technology support for design teams in the field, it is crucial to study the end-users of such systems so that their needs and methods of work could be adequately served.

**Firm and Project Description**

A mid-size architectural firm with approximately thirty employees was chosen for this case study. The relatively large scale projects, including commercial, institutional and corporate buildings, that the firm usually undertakes, constitute the main reason for its selection. The extended scope of the projects renders collaboration necessary and it therefore offers an opportunity to observe its particulars.

The analyzed project was for a power plant to be located on a university campus. For the design of the power plant the State of Michigan commissioned an engineering firm, that calculated the space needed to locate the machinery and designed a simple building envelope to house it (see Figure 3.1 below). They, then, sub-contracted the architectural firm to design (mainly) the outer shell of the building.

![Figure 3.1: Simple building envelope design.](image-url)
The design of the power plant began with forming a team of:

1. The president of the firm and project leader.
2. The project architect (involved in architect-client meetings, developed cost estimates and specifications).
3. A design critic (participated in meetings)
4. A designer (designed and generated most of the drawings)

The project leader started the design activity by requesting the development of three types of schemes: traditional, modern and high-tech. The designers started developing different design alternatives using both traditional and computerized modes of representation (see Figure 3.2 below). The initial design phase extended over approximately six months and included multiple modifications of the scheme resulting from aesthetic and functional considerations, but also from the financial concerns of the client.

![Figure 3.2: Traditional and computerized modes of representation.](image)

The computer modeling of the new structure within its natural landscape constituted a considerable part in which the author was personally involved. Although
mainly intended for presentation purposes, the computer models acquired an important role in the formulation of various aspects (compatibility with the site, refinement of volumetric and stylistic details, colors) of the final scheme (see Figure 3.3 below).

![Figure 3.3: Compatibility with the site.](image)

**Initial Observations**

The first thing one notices when visiting an architectural office is the large number of artifacts that surround the designers (see Figure 3.4 below). A partial listing of these artifacts illustrates their diversity and varied uses: blueprints, drawings, sketches, photographs, letters, lists, messages, notes, field reports, models, computer images, catalogues, magazines, disks and samples. This seems to confirm Schön’s notion of designers being immersed in design worlds while designing: “These [design worlds] are environments entered into and inhabited by designers when designing. They contain particular configurations of things, relations and qualities, and they act as holding environments for design knowledge.” In this dissertation the notion of design worlds is viewed as an abstract state of the designer which may exist only in the mind. Thus a distinction is made between a design world and a design setting. The latter signifies the physical and social environments that are, usually, constructed by a group of designers.
Examples of design settings include the designer’s desk, the studio, the project room, the crit space and the conference room.

![Figure 3.4: The proliferation of artifacts in an architectural office.](image)

A second observation that is immediately noticed is the configuration of the office (see Figure 3.5 below). The office layout shows two concepts: teamwork and hierarchy. Teamwork is reflected in the layout of four areas with low partitions. Each of these four areas serves as a project space for a team of designers working on the same project. The initial goal of the president of the firm was that designers switch desks whenever a new design team is formed. In practice, however, teams seemed to be formed of members whose desks are in the same project space thus rendering place trading unnecessary.

Hierarchy, on the other hand, is readily observed through the delineation of a main studio space, secretarial pool, offices for senior architects, accountants, marketing, partners and offices for the president and his partner. The office also includes other support spaces such as a library, conference room, copy room, kitchen, model building and photography areas, and a lobby.
Collaborative Design

Designers sometimes work alone and sometimes with others. Similarly, groups may work alone or with other groups, mostly possessing different expertise. Therefore, a general framework could be proposed that views design work as being divided into three categories: Individual work, teamwork and multi-disciplinary work (see Figure 3.6 below). This framework is used as a starting point to explain the different design settings in relation to the artifacts and interactions taking place within them. Figure 3.6 shows the relationship between the designer and the artifact and delineates the issues and activities associated with them. On the other hand, another aspect of the same framework is related to the hierarchical order of its categories (see Figure 3.7 below). According to this order, individual work forms the basis of teamwork which, in turn, forms the basis of multi-disciplinary work. The pyramid, however, neither starts at individual work nor ends at multi-disciplinary work. While omitted from Figure 3.7 below, other types of activities, such as national and international collaborative efforts may form the upper layers of the pyramid while cognitive and neural functions may form the lower ones. The two layers
addressed in this study (individual work and teamwork) could be readily observed in one physical location and thus were the most convenient to study. Furthermore, they seemed to be more relevant to the goals set forth in this dissertation.

Figure 3.6: A framework for design work.

Figure 3.7: A hierarchical view of the same framework.
Individual work

Even though this dissertation focuses on collaborative processes, the processes that are carried on individually should remain of concern because they form the building blocks of any collaborative process. The current technologies used in the architectural office, such as drafting pens and computer-aided design systems, are, to a large extent, tools for the single user. The introduction of new group tools does not necessarily mean that the single-user tools will become obsolete. Therefore, there is a need to understand how these tools are used and whether they can still be incorporated in a group-oriented process. Furthermore, a large portion of design ideas are still generated privately and then communicated and developed through teamwork. As a result, understanding the characteristics of individual work is an important first step in understanding collaborative work.

Donald Schön describes individual design as a process of dialogue, mediated by an artifact, between the designer and the design situation:

A designer makes things. Sometimes he makes the final product; more often he makes a representation— a plan, program, or image— of an artifact to be constructed by others…. [The designer] shapes the situation, in accordance with his initial appreciation of it, the situation “talks back,” and he responds to the situation’s back-talk.

In a good process of design, this conversation with the situation is reflective. In answer to the situation’s back-talk, the designer reflects-in-action on the construction of the problem, the strategies of action, or the model of the phenomena, which have been implicit in his moves.8

Therefore, artifacts produced in this design setting share two dichotomous characteristics: divergence and convergence.9 Artifacts are divergent because they are created and used to broaden the design space being explored. They are convergent, on the other hand, because, by depicting design alternatives, they help eliminate the undesirable, thus helping in converging on the acceptable. One piece of evidence found to support this
claim is a sheet produced in private for the purpose of studying façade alternatives (see Figure 3.8 below). What is interesting about this sheet is that it contained a large number of small drawings, all depicting façade alternatives for the proposed design. It is obvious from studying this sheet that the designer was trying to efficiently explore as many alternatives as possible. The solution was to create a large number of small drawings that could fit on the same sheet, thus enabling a comparison the alternatives. Time was also saved because of the drawings’ small size. In such cases, the artifacts are usually incomplete and many form alternative depictions of a singular design issue. However, while generating different alternatives, the designer is also converging on a design solution. This shows itself in the more refined elevations such as the three elevations with shadows.

Individual work does not always occur at the designer’s desk. In many cases, a designer re-creates this setting within a group setting. During one of the meetings, it was mentioned that the project leader started drawing sketches on the side which he later showed to the rest of the participants (see Figure 3.9 below). This corroborates Tang and Leifer’s observations:

For example, in Session A, one participant began to sketch a drawing in a private area of the workspace, demonstrating apparent loss of interest in the discussion and sustaining his own interest by doodling. The others eventually directed their attention to his drawing, and yielded a turn to him, asking him to explain it.

Artifacts created within an individual setting vary in roughness, completeness and clarity. These attributes depend on the creator’s preferences, the mode of creation and intended mode of use. If an artifact is created at a meeting it will, most likely, be rough and incomplete because it is drawn fast. Furthermore, artifacts may be generated to support a conversation and thus they need to keep up with the discussion which results in their ‘sketchy’ nature (see Figure 3.10 below). Speed is also a reason certain artifacts are not chosen as the vehicle to discuss ideas. For example, it is rare to see three dimensional
models patiently constructed during design discussions. If models are used, they are usually prepared ahead of time while, in few cases, they are assembled quickly from cardboard or even everyday objects such as lighters, pens and erasers.

Figure 3.8: Divergence and convergence while exploring alternatives.

Figure 3.9: Individual sketching within a group setting.
Teamwork

The Individual Versus the Group. Webster defines teamwork as: “work done by several associates with each doing a part but all subordinating personal prominence to the efficiency of the whole.”\(^{12}\) This definition hints at a tension that exists between the individual and the group which shows itself through four issues: democracy, the sense of belonging to a group, group norms and ownership privileges.

Democracy. For teamwork to be successful each participant should be able to voice an opinion. Democracy, of course, does not mean that all participants have the same role. Rather, it means that each of them can contribute to the design process in an open and safe atmosphere of cooperation. Participants should not be afraid to voice opinions that are in conflict with other views. Ideally, democracy in teamwork also implies that each participant’s benefit from joining the team should be proportional to the
work contributed toward achieving the goals of the team. Unfortunately, this is not always the case. Grudin discusses this issue in the context of introducing a new groupware system:

Given the different preferences, experience, roles, and tasks of members of a group, a new groupware application will never afford every member precisely the same benefit. When it is introduced, some people will have to adjust more than others. One can hope that the differences are not great— but often they are. There should be a collective benefit individually, even if some benefit more than others.\textsuperscript{13}

The concept of democracy is very much related to the concept of group norms, which will be discussed later, in that group norms determine whether a democratic process can be achieved.

\textit{The sense of belonging to a group.} The main purpose of initiating teamwork is that the task is too complicated for one individual to accomplish. With a team of designers the design task can be subdivided into smaller and simpler sub-tasks. Even though team members may work separately on design sub-tasks, they maintain a continuous, open channel of communication among themselves. This continuous communication is one of the factors that establish a sense of belonging to a group. It is conceivable that a designer could belong to several teams at the same time. It was observed, however, that when this was the case the designer would work for a period ranging between one to two weeks (or until a good stopping point is reached) on one project and only then switch to another. This mode of work allows the designer enough contiguous time to focus on and get immersed in the “design world” of that project. While inhabiting this world the designer has access to the design knowledge related to the project. Once the designer leaves one design world for another, it becomes very difficult to answer questions related to the previous one. The designer can only regain access to that knowledge by re-entering that design world and inhabiting it for some period of time. This phenomenon seems to be part of everyday experience in that it occurs whenever a
switch between any two relatively demanding tasks takes place. However, project managers rapidly switch contexts many times during the course of one day. They are able to do that because the nature of the problems they address does not demand a spatial and temporal investment in any physical artifacts. While a designer or draftsperson, on the other hand, has to allocate considerable space and time for creating and interpreting the artifacts needed to address the situation at hand.

*Group Norms.* Team members may differ in their vision of the means to reach the stated goal of the team or they may even differ in their vision of the goal itself. Actually, according to (Olson 1989), one of the reasons to form groups is to build such a shared vision: “… among the more important [reasons we work together] are … [1] The members of an organization are more likely to agree to a decision or plan if they have participated in or even just observed the process that lead to it. [2] Participating in a group can also enhance the formation of shared goals and perspectives.”¹⁴ In order to push the design forward and achieve the desired goals, groups usually agree on rules and procedures that are to be followed even if these rules disagree with a participant’s preference. Bryan Lawson discusses the characteristic of group norms when he writes: “One of the most significant factors in the formation of effective groups seems to be the development of group norms. Such norms may include conventions of dress, speech and general behavior and serve to suppress the individuality of members in favor of an expression of attachment to the group.”¹⁵ He also writes that norms are not easily agreed upon:

Norms are often not developed without some pain. It is sometimes said that groups go through phases of “forming”, “storming” and “norming” before “performing”. This is because norms to some extent must grow out of the collection of individuals. As each tries to impose his or her character on the group, conflicts are likely to arise before common perceptions of the group’s goals and accepted norms develop.¹⁶
Group norms can cause some problems in that they become habitual and may suppress, otherwise, good design ideas:

Despite the presence of diverse views and backgrounds, groups often prematurely close off exploration of alternatives. A large number of studies of brainstorming showed that fewer ideas were produced by groups working face-to-face than a comparable number of individuals working alone. Factors such as competition for time, anxiety about verbal skills, presence of authority figures, and general fear of disapproval all contribute to this inhibition effect.17

On the other hand, if group members are very individualistic and therefore non-cooperative then the group’s efficiency will suffer. There seems to be a balance between conformity and individuality in order to achieve optimum efficiency. This balance involves a cycle of “falling-into”18 the group where one is no longer in touch with one’s self and then regaining individuality, by taking a public stand, in order to maintain authenticity to one’s self. The group norms, format and makeup partially determine the efficiency and quality of the process by determining whether such a balance can be achieved.

Ownership Privileges. Designers interact with their artifacts and other individuals’ artifacts through an implicit set of ownership privileges. This set includes: control through lawful claim or creation, management and use. The architectural firm owns the design-related artifacts and thus can create, control, manage and make use of them. However, ownership privileges may be delegated such that the owner may not manage or even use the artifact. Some of the sketches collected were created by and belonged to the project leader, but were kept (thus managed) by another designer. The delegation of privileges can result in tensions when, for example, the manager realizes the boundaries of those privileges. In many instances during conversations with one of the designers, dissatisfaction with the lack of control over the course of the project was expressed. Another source of tension resulted when the creator of an artifact, after having
had control privileges during the creation phase, lost the privilege to the group or to other individuals. When asked about the fate of one of the drawings, for instance, an interviewed designer shrugged his shoulders in dissatisfaction, exclaiming: “it is out of my hands now.” Thus, the understanding of ownership privileges are crucial to the understanding of the interactions between individuals and artifacts and between individuals and groups.

**Group Design Settings.** Teamwork in the architectural firm studied was carried out in two settings: formal and informal. The formal mode of teamwork took place in the design meeting room while the informal mode took place in the design studio. The processes and the artifacts that exist in each setting warrant a closer look:

**The Formal Setting.** The typical scenario of this mode is as follows: the design team reaches a “milestone” in the design process and decides it is time to meet and discuss the design project’s status and where it should go next. A memorandum is distributed announcing the time and place of the meeting. There is usually a specific meeting room with audiovisual equipment and display space. The designers usually prepare artifacts for this meeting. During the meeting many complex processes take place. Three major processes can be readily observed: individual work, design review and “multilogue”. Individual work is exemplified by sketching and talking. Participants contribute to the meeting through this type of individual work. Design review forms a large part of the processes in a design meeting. Usually, the design meeting starts with a presentation followed by a period of discussion. Unlike a lecture, however, the discussion does not take the form of a question and answer session directed towards the speaker. It takes the form of parallel conversations and brainstorming. It might also turn into a chaotic situation that is soon brought back to order by the design team leader who tries to reach a consensus and to outline future directions for the design project. Finally, the design team members leave the meeting with an idea of what is to be done next.
Sometimes, these ideas differ appreciably among them and thus another design meeting is required.

What can be derived from this scenario is that the design team tries to reach a common vision, i.e. an agreement, rather than to develop the design. In such meetings, conflicts are resolved and decisions are made, but no extensive artifacts are produced. The artifacts brought into the meeting usually consist of drawings, models, notes, schedules, material and color samples and so on. The artifacts created during the meeting usually take the shape of rough sketches, notes and annotations of other artifacts. Once this common vision is constructed, the team members end the meeting and go back to the design studio where they continue in an informal mode of teamwork.

The Informal Setting. As mentioned previously, the studio space at the architectural firm studied was planned according to a team concept; where each team would have a project space of its own. One can readily detect from the artifacts surrounding them that all the designers in one location of the studio space are working on the same project. The studio space is surrounded by shelves of catalogues, magazines and material samples. In addition, the studio often contains life-size samples of equipment and hardware such as door knobs, tiles and bricks. These artifacts allow the designer to “project” them into the design in order to better predict the outcome of decisions. This notion of projection is articulated by Coyne and Snodgrass as follows:

The act of designing involves the projection of a partial design onto a particular design situation. The match or otherwise between this projection and the situation as it presents itself brings objects to light and changes the game.20

Teamwork in the informal mode is ubiquitous, but inconspicuous. During the course of a typical day the designers in a team will interact in a complex and continuous way similar to the process of multilogue described above. The range of interactions covers a simple request for information, an informal and spontaneous gathering, a
handing of a memorandum, an illustration of a concept, a discussion during lunch break and many more. These interactions can be characterized as inconspicuous because they are so much immersed in day-to-day habits and practices that they are usually overlooked. Furthermore, these interactions are informal because: 1) they do not require any special scheduling or coordination effort; 2) they do not need a designated space in order for them to occur; 3) and they do not appear to the participants as an interruption of their work, but rather as an expected and integral part of it. The general impression one gets from the studio space in the case-study is that the designers seem to be working individually. Yet, when observed for longer periods of time, it becomes evident that they continuously interact, sometimes in parallel with performing their individual tasks. For example, a designer was observed casually stopping by the desk of a fellow team member to ask a question (see Figure 3.11 below). The latter responded, again casually, neither lifting sight from the drawing, nor halting work. Thus, one can conclude that designers carry on design tasks transparently as part of their everydayness not unlike other daily tasks.

Figure 3.11: Casual interactions.

The importance, proliferation and extensive use of artifacts are readily observable in the design studio. During one of the informal meetings, a senior level architect was observed explaining a design idea to another junior-level architect (see Figure 3.12
below). Throughout this meeting, the senior architect was expressing himself mainly through drawing actions. During the discussion, the two architects were joined by another who started as an observer, but began to play a more central role when he left the discussion for a brief moment and returned with specification catalogs. Again, in this instance the artifact plays a central role in the modes of communication among the designers. Furthermore, during the whole span of this meeting, at least one architect was observing the artifacts being discussed rather than observing the other participants. Most of the time, the listener(s) were looking at the artifacts while the speaker’s gaze alternated between the artifacts and the audience. That suggests that the artifacts, especially the ones being discussed, are the true center of attention and play the role of discussion enabling devices. In addition, this description indicates that there are many artifacts that take the form of catalog items which are standardized. The role that this reference material plays in the design process in general is not just that of constraining and specifying, but also that of focusing in that it carries the design from a rough and inaccurate form to a more refined one. Other forms of reference material, such as photographs in architecture magazines, act more as sources of imagery and design concepts.

Figure 3.12: Informal design discussion in the studio space.

Finally, it is interesting to note that the attention and care by which artifacts are annotated, categorized and stored also indicates their significance for architectural practice. A finished drawing usually contains a “title block” that clearly identifies the number and title of the project, creation and modification dates, the names of the
responsible architect and the draftsperson who created it (see Figure 3.13 below). The reason behind such care and accuracy is that these artifacts are legally binding and the firm can be liable for any mistakes.

Figure 3.13: Annotation of finalized artifacts.

**Summary of Findings**

The case study uncovered several aspects of collaborative design, most of which were intuitive:

- Collaboration relies on individual work. Thus, collaborative systems must not ignore the needs of the individual.
- Collaboration is synchronous and asynchronous. Collaborative systems must seamlessly alternate between the two modes.
- Artifacts are central to collaboration. Collaborative systems must find appropriate representations for artifacts and methods of creation, editing, controlling access and distribution.
• Artifacts and modes of design depend on the setting in which they take place and their intended mode of use. Collaborative systems must capture information about the setting of the tasks they are supporting and provide tools appropriate to that setting.

• Designers are given tasks and respond to these tasks by proposing alternatives. Designers are usually assigned sub-tasks and form teams to work on larger problems. They work synchronously and asynchronously.

• Designers develop proposals and reach favorable solutions by expanding and contracting the solution space. Collaborative systems must allow multiple parallel solutions to exist and at the same time allow for evaluative processes to prune the solution space. Additionally, the need for evaluation is tightly coupled with the notion that competition can be viewed as a form of collaboration.
Notes to Chapter III


2 These are virtual artifacts that are constructed in the air to convey an image to others (also known as airboards).


7 Due to time and space limitations this category was not studied.


9 The use of these two terms here differs from Coyne and Snodgrass’s usage of them in “Is Designing Mysterious?”, 126.

10 It is here that an overlap between individual settings and group settings, which will be discussed next, occurs.


Olson, G., “The Nature of Group Work”

This term is loosely borrowed from M. Heidegger’s notion of *falling* although his use of the term is much more profound than its use here. For a clear discussion of Heidegger’s use of the term see: Dreyfus, H. L., *Being-in-the-World*, Division I (Cambridge, MA: MIT Press, 1991): 225-237.

This term is borrowed from Duke, R., *Gaming: The Future’s Language*. (New York: Sage Publications, 1974). The term was originally derived from the word “dialogue” and refers to a mode of parallel discussion among more than two participants.


They usually occupy a prominent position as well.
CHAPTER IV
A CONCEPTUAL FRAMEWORK FOR COLLABORATIVE DESIGN

Based on the observations of architectural teamwork in a professional setting described in the previous chapter, a model was derived that describes the contributions of the design participants and indicates the various activities and transactions that took place during a typical design process cycle (see Figure 4.1 below). The model also indicates the chronology and intensity of some transactions. Additionally, during the observation of this case study, various sketches and drawings were collected, categorized by creator, and associated with the particular activities that produced them (see Figures 4.2, 4.3, 4.4 below). To cross-reference the artifacts and activities, a roman letter is included in the upper right corner of some activity boxes in figure 4.1 below: This roman letter corresponds to a sub-section in figures 4.2, 4.3, and 4.4: For example, the activity “Transfer Drawings (Engineering Firm)” includes the letter “A”, which corresponds to the artifact depicted in figure 4.2-A.

What follows is a description of the conceptual model elements and an explanation of the chronology of activities carried out within it. Based on that analysis, a protocol of interaction is proposed that maps the essential elements of the analyzed model. It is hoped that this protocol will form the basis of a domain-specific, computer-supported environment for enabling collaborative architectural design among geographically dispersed participants. Portions of the protocol have been implemented within an early computer prototype that supports synchronous collaborative design (see Chapter V).
Figure 4.1: Conceptual model of teamwork in architectural practice.
A. Transfer drawings (Engineering firm)

B. Develop alternatives (Designer)

C. Present alternatives (Team)

Figure 4.2: Samples of design artifacts collected at various stages of the process.
D. Discuss Alternatives (President & Team)

E. Develop Digital Terrain and CAD Model (CAD Consultant)

F. Develop Modifications (Designer)

Figure 4.3: Samples of design artifacts collected at various stages of the process.
G. Develop CAD Model (CAD Consultant)

H. Discuss Modifications (Team)

I. Present Solution

Figure 4.4: Samples of design artifacts collected at various stages of the process.
Participants

In this case study, the participants consisted of an engineering firm and an architecture firm. In a reversal of typical design contract scenarios, the engineering firm subcontracted the project to the architecture firm in order to design the exterior shell. The participants within the architecture firm were: The firm’s president, the project architect, a designer, a design critic, and a CAD consultant. For the purpose of representing activities carried out by the architecture firm team as a whole, a column is added with the title “Team.”

Chronology of Activities

The chronology of the first phase of the project was as follows:

- The engineering firm negotiates the contract with the architecture firm and transfers the project documents.
- The architecture firm president assembles a design team.
- The team meets and plans for creating alternatives.
- The individual members create alternatives.
- The design team presents, discusses, and modifies design solutions.
- A CAD consultant is hired to develop CAD models and integrate the project with a digital terrain model.
- The 3D CAD model is presented and modifications are requested.
- A preferred solution is pursued and submitted for further development.
- The developed solution is presented internally and to the client.
- Smaller modifications are requested and a second development cycle is executed.
Categories of Activities

Several general categories of activities have been derived from the afore-mentioned conceptual model: (1) Request, (2) Work, (3) Plan, (4) Discuss, (5) Present, (7) Choose. These categories involve individual as well as collaborative work and can take place synchronously as well as asynchronously. For example, developing a solution can be an individual task while discussing the different alternatives is by nature a collaborative activity. Requesting information and its transfer can be carried out immediately (synchronously) or over a longer period of time (asynchronously). The scope of this dissertation does not permit the presentation of the full analysis carried out. However, based on this analysis, a protocol of interaction for supporting collaborative design was derived and is included below.

A Protocol of Interaction in Collaborative Design

A protocol of interaction maps and abstracts part of the activities performed by designers in the early stages of an architectural project. In that sense, a protocol of interaction can be thought of as a model. The usefulness of this model depends on its capacity to represent reality in a consistent manner thus enabling its understanding. Broadly, a model formalizes a set of declarative and procedural units that are usually mapped from their counterparts in reality. In the domain of interest addressed in this dissertation, a protocol of interaction is a model that formalizes a set of declarative and procedural units that govern the way in which a group of participants build design objects.

The units in the proposed model consist of two main entities: A Participant (PAR) and a Collaborative Design Object (CDO). PARs represent the human designers using the system to collaboratively design objects represented by CDOs. These objects are the focus of the collaborative activity and can have several sub-types: (1) Task, (2) Proposal,
and (3) Artifact. Examples of attributes shared by Tasks, Proposals, and Artifacts are: (1) Identification, (2) Creator, (3) Date of Creation, (4) Date Last Modified, (5) Access Control List, (6) Location, (7) Status, (8) Version Number. Due to the object-oriented design of the protocol (Booch 1994), these attributes are defined at the CDO level and inherited by the Task, Proposal, and Artifact objects.

**Explanation of the overall model**

The representational units mentioned above were illustrated using the NIAM notation (Turner 1989). The following sections include a description of the units and the relationships among them which are best understood when compared to their graphical representation (see Figure 4.5 below):

- One or many members of PAR, edit (i.e. create, delete, modify) one or many members of a CDO.
- A CDO can have other embedded CDOs in it, but cannot embed itself (Strict hierarchy).
- A CDO has three sub-types: Task, Artifact, and Proposal. These subtypes inherit the attributes of a CDO such as the ability to embed other CDOs in themselves. Since PAR can edit a CDO, PAR can edit a CDO subtype.
- A Task starts with (i.e. takes as input) an Artifact.
- A Task ends with (i.e. produces) an Artifact.
- A Task is satisfied with (i.e. can be solved by) zero, one or many Proposal(s).
- A Proposal starts with (i.e. takes as input) an Artifact.
- A Proposal ends with (i.e. produces) an Artifact.
The Basic Mechanisms of the Model

The participants join a common session and start by viewing a shared window containing the status of the project at hand. Using this overall view of the project, participants can navigate the various nodes to explore, in more detail, the status of the various sub-nodes. Figure 4.6 below illustrates a conceptual diagram of a possible flow of tasks (T), proposals (P) and artifacts (A) in an imaginary project. Participants can collaboratively create new task nodes and arrange them in a hierarchy. They can subdivide a task into several sub-tasks, attach competing proposals to a given task, and specify the related artifacts.

Figure 4.5: Top-level NIAM representation of protocol elements and roles.
Each new task has an input artifact (e.g. A1, A5) and an output artifact (A5, A8). Tasks can be subdivided into sub-tasks (e.g. T1.1, T1.2) and the input artifact of the parent task can be passed along to the sub-task or proposal (e.g. A1, A6). Competing solutions are attached to tasks in the form of proposals (e.g. P1.2.1, P1.2.2). The output artifact of a proposal is considered acceptable when it becomes the output artifact of the parent task (e.g. A8 was chosen when evaluated against A7). Additionally, the output artifact of a task can result from a merger of artifacts from other sub-tasks or proposals. For example, the final artifact (A9) is the result of merging the output artifacts of sub-task 1.1 (A5) and sub-task 1.2 (A8).

Figure 4.6: An example flow of Tasks, Proposals, and Artifacts.
Collaboration happens at all levels of the flow depicted in figure 4.6: At the level of task design, participants conduct a meeting to decide on what tasks need to be created and who should work on them. This is usually done synchronously. Asynchronous collaboration happens when an individual works on sub-tasks or creates proposals and attaches them to the parent task. Proposals can embed a series of artifacts that can be played in sequence (thus emulating design moves). On the other hand, a proposal can embed at minimum the output artifact that represents the outcome of the proposal. Synchronous collaboration as well as competition takes place when a task or a proposal is assigned to a group of participants. Additionally, synchronous collaboration and competition occurs when participants meet to evaluate and defend proposals, review the status of the project, and merge artifacts from sub-tasks and proposals into output artifacts of tasks. This merging process is left up to the participants in that they verbally discuss the proposals, manually copy and paste parts of proposals into the final artifact and edit the final artifact to make it coherent. Participants, based on their role and access privileges, can observe the status and content of tasks, proposals, and artifacts.

Discussion

The conceptual framework put forward in this chapter largely resulted from the analysis of the case study presented in Chapter III. However, the framework also serves as a hypothesis regarding the essential representations for collaboration in architectural design. One initial hypothesis is that collaboration depends on a shared common protocol of interaction (e.g. language). That is, regardless of what a receiver does with a received message, collaboration fundamentally depends on the receiver’s ability to understand the message. The framework makes an analytical and intuitive hypothesis that a collaborative process includes participants (humans), artifacts (designed objects), tasks to accomplish (problems to be solved, buildings to be designed), and competing proposals (as observed
in the case study). The conceptual framework also asserts the hypothesis that artifacts, tasks, and proposals should be hierarchical and as such contain sub-artifacts, sub-tasks, and sub-proposals. Furthermore, the conceptual framework and case study point to the hypothesis that privacy, hierarchy, synchronicity and asynchronicity are essential for collaboration. Finally, the conceptual framework asserts that artifacts are the essential vehicles and enablers of collaboration. The role of artifacts in collaboration is proposed as being focal in organizing participants and enabling them to interact. In the following chapters, this conceptual framework will be tested through several approaches and experiments.

In the next chapter a basic hypothesis is tested that states that a common protocol of interaction among software systems is fundamental to collaboration. As such, two collaborative software modules that can inter-communicate were developed independently. What was shared was only a specification of the messages being transmitted. This was agreed upon through electronic mail conversations. No actual software code was shared. This was done to isolate the elements of the protocol of interaction from the implementation and to test its interoperability.
CHAPTER V
SYSTEM ARCHITECTURE AND EARLY EXPERIMENTS

The field of Computer-Supported Collaborative Work (CSCW) has produced numerous systems that allow the sharing of virtual workspaces (Bly 1988), (Tang and Minneman 1991). These general-purpose systems typically allow multiple users to view and annotate a shared virtual whiteboard represented either as a video image or a digital bitmap. The main disadvantage of these systems is that they do not easily allow computers to interpret the artifacts being shared – a feature found to be useful for collaborators (Olson and Olson 1991). That is, although these methods seem to allow the complete sharing of a virtual workspace, they lack the parsimony and structure to make them useful for synchronous collaborative design. Additionally, current collaborative groupware technologies – such as unstructured shared whiteboards, multi-media electronic mail, desktop conferencing systems, and distributed single-user software – are inadequate for the particular needs of collaborative design processes. On the one hand, CAD systems do not support simultaneous, multi-user discussion and co-production of architectural documents. On the other hand, the limited data structures of generic groupware systems do not capture architecturally significant semantics, such as mass, material, space, and function.

To expand the capabilities of current CAD systems and to go beyond the limited expressiveness of general purpose collaboration tools, a new generation of design-oriented software is needed. Such software should combine collaboration technologies with a rich and meaningful representation scheme. Furthermore, it should support the early design phases, wherein many of the most important decisions are made and
collaboration is most important: client debriefing, data collection, architectural program formulation, and schematic design generation. These activities are crucial to the evolution and quality of the final design, and they are receptive to and can benefit from computer support. Furthermore, these are precisely the areas where current CAD systems are weakest.

In this chapter, the system architecture of a Synchronous Collaborative Design Environment (SYCODE) is introduced. SYCODE’s goal is to enable teamwork among geographically dispersed designers using the Internet. SYCODE is a long-term project which aims at supporting the creation of global design teams. This chapter provides an overview of the theoretical foundations of SYCODE and an outline of its overall architecture, communication protocols, and data structures. In addition, it includes results from the first experimental phase, that involved the simultaneous development and testing of SYCODE clients on heterogeneous computer systems at remote sites - Hong Kong and Ann Arbor, Michigan.

The case study described in Chapter III concluded with the finding that, in architectural design, supporting focused tasks is more beneficial than supporting general-purpose ones and that artifacts, to a great extent, act as facilitators of collaboration. A disadvantage of collaborative whiteboard systems that represent the shared workspace as a video image or a digital bitmap is that they do not easily allow for the interpretation of the shared data by the connected computers (Tang and Minneman 1991). Additionally, researchers are finding that these general-purpose multimedia applications are hindering richer ways of collaboration:

Collaborative multimedia applications have been somewhat lacking in imagination, focusing on “talking heads” video as a way to create telepresence, or on data retrieval for simple information “foraging” and sharing tasks. We argue that collaborative multimedia technologies should be used as means of providing richer, deeper ways to collaboratively compose shared artifacts such as documents, movies, data visualizations,
simulations, designs such as architectural drawings, bulletin boards, libraries, and animations. We should also create ways to collaboratively analyze data within these artifacts. (Nardi 1994).

General-purpose groupware applications such as those described in the previous section are very well suited for their intended purpose. They allow geographically dispersed users to meet, communicate, point at, annotate and share synthetic artifacts in near-real time. Some also allow the remote control of a computer or application. Yet, as Nardi notes, there is a whole class of needs that is not addressed in these applications. For example, if a musician would like to collaboratively create a piece of music, then someone must write a collaborative music authoring application from the ground up or convert a single user application into a multi-user one. In most cases, the latter has proved to be inadequate due to differences in user interaction modes. The same can be applied to the field of architecture. Architects sometimes need to carry out domain-specific tasks that general-purpose collaborative tools simply cannot handle. The example of co-creating a building program (the specification of the spatial requirements for a building) is used to illustrate the need for domain-specific groupware. It is important to note that this is not an argument for the replacement of current collaboration software, but simply for the need to augment them.

**Requirements**

The primary requirement for SYCODE was to provide an environment that enables the sharing of semantically relevant data structures while maintaining a low communication overhead.

Several early collaborative systems could facilitate communication only among identical hardware platforms, operating systems, or graphic display systems and some even require specially equipped rooms (Tatar et al. 1991). However, realizing the diversity in computer systems, an early requirement for SYCODE was to isolate it, to the
largest degree possible, from any particular hardware platform or graphic display system. At a minimum, it was required to communicate between two different hardware platforms to illustrate the potential for a platform-independent implementation.

Many of the early collaborative tools could only support the hardware platform they were developed on (Watabe et al. 1990). To succeed in connecting users from various backgrounds, the extensive variety in hardware, configuration, display systems and operating systems must be accommodated. To illustrate the feasibility of achieving this goal, SYCODE client applications were developed independently, based on a verbal description of protocols, with minimal sharing of actual source code. The first site – in Hong Kong – used a Silicon Graphics computer, the C programming language, and the X Window System, while the second site – in Ann Arbor, Michigan – used a NeXT workstation, a combination of C and Objective-C programming languages, and Adobe’s Display Postscript. Though their user interfaces and implementation details are different, these prototypes allow multiple users to share a virtual design space – both within and between the remote sites – in which to create and manipulate architectural elements.

Parsimony is another important goal sought in the design of the SYCODE architecture. There are at least two layers of issues in implementing domain-specific software for computer-supported collaborative work. The higher layer, concerning the semantics of architectural design teamwork, is the principal focus of this research effort. Nevertheless, the lower layer, concerning the technical details of long-distance network communications, imposed limits on what can actually be accomplished with the existing infrastructure. If one wishes to move beyond theorizing about computer-supported collaborative work and discuss actual implementations, then one must come to terms with the limitations imposed by the network. As the experiments in Chapter VII further illustrate, low transmission rates and network traffic result in time delays that are
significant enough to become issues at the higher semantic layer of the design system (as well as in the details of how to program the main event loop).

For example, when a client changes an element of the model, when is the change considered to be officially done: Immediately? After receiving a round-trip confirmation from the server? Or after all clients have received the message from the server? How can a client know that all other clients have received the message, except by requesting a confirmation from each – thus multiplying the communication traffic and compounding the problem?

If each client considers actions to have immediate effect, then how are discrepancies in other clients’ simultaneous states of the model resolved? Where does the official state reside, and how is it restored and redistributed in the event of a network failure?

As a developer of a collaborative design system, one may adopt the philosophy that these are transient technical problems that can be resolved with improved network infrastructure and robust software. However, the current weaknesses of long-distance network computing suggest an alternative philosophy that may lead to a richer system design. If one adopts the view that discrepancies are inherent in human interaction, then the goal is not to eliminate them, but to manage them. Technical glitches in the computing infrastructure are only one source of discrepancy.

The aim in designing SYCODE is not to automate design, but to support collaboration between human designers, working in proximate or remote locations, simultaneously or in shifts. Thus, the acceptance of that discrepancy from a variety of sources is inevitable, and must be managed at the higher semantic layer of the system as well as in the details of programming event loops and Internet socket communications.
The Share-Kit

SYCODE is based on the Share-Kit (Jahn 1994) which is a general-purpose toolkit that facilitates the construction of synchronous groupware. It supports the development of new systems as well as the conversion of existing single-user systems to groupware applications. The Share-Kit uses a client-server paradigm that allows computer programs, called clients, to send and receive information by connecting to the same session offered by a central computer program called the server (see Figure 5.1 below). In this figure, as an example, clients A and B are communicating through session 1 while clients C and D are communicating through session 2. The different shapes for the clients illustrated below indicate the fact that they can differ in implementation details, exist on heterogeneous platforms, and react differently to the same message. However, they have in common a communication module that embodies a finite set of message protocols.

The Share-Kit’s basic mechanism depends on a shared function and a shared data representation. A shared function is written in the C programming language and declared to the Share-Kit, along with relevant C data structures, to establish a common interface or protocol for collaboration. Using this protocol definition, the Share-Kit generates a communication module that broadcasts messages through a central server to all connected clients whenever a shared function is called from any of the clients.

Figure 5.1: The Share-Kit’s client-server architecture.
Additionally, The Share-Kit offers a suite of management functions that enable synchronous collaboration, state updating, and state consistency. For example, when a client calls a shared function, the Share-Kit server distributes the message in real-time (aside from any network delays) to all connected clients. Moreover, when a new client connects to a session already in progress, the server automatically instructs one of the already connected clients to update the new comer with the current state. It is important to note, however, that the server itself does not store a description of the current state. Finally, while updating a new comer, the server defers and buffers all requests from other clients to insure complete consistency of the current state among all connected clients. Other features include: an optional token mechanism that grants exclusive broadcast rights to the client in possession of the token; and the ability to query the number of connected users.

The Architecture of SYCODE

SYCODE’s underlying architecture represents concepts and entities found in professional practice. The representation scheme is divided between agents and artifacts. Agents consist of representations of either human participants or software modules that possess certain skills. Artifacts, on the other hand, represent the virtual counterparts of objects that architects and others deal with in their profession. Given this distinction, SYCODE can then represent the interactions and relationships among agents, among artifacts, and among agents and artifacts.

SYCODE defines several types of agents: 1) One agent is a Group Overall Director (G.O.D.) that has ultimate authority to perform any action in SYCODE (see Figure 5.2 below). The G.O.D.’s responsibilities include defining protocols for interaction, assembling design teams, and initiating collaborative projects. 2) Each design team member is also an agent possessing certain rights with regard to artifacts. 3)
Artificial agents consist of computer programs that interact with other entities in SYCODE. One such artificial agent, called the manager, remains active at all times, contains the current status of the project, and maintains book-keeping information and preferences. The G.O.D. interacts with the manager to set all preferences and provide the needed information to manage the project at hand.

Figure 5.2: The various agents in SYCODE.

Artifacts in SYCODE are organized in a hierarchical fashion. At the top of the hierarchy is a project artifact that corresponds closely to its architectural counterpart as found in professional practice. It includes links to component artifacts and to the various agents that are involved in creating and modifying them. In addition, it maintains a record of its evolution through a directed graph representing states of the project through time. A state artifact can be thought of as a snapshot of the project at a certain time. Thus, SYCODE always represents the current status of a project through a current state artifact. A state is composed mainly of workspaces in which to create entities. Currently, the design of SYCODE requires one public shared workspace where the current state resides. In addition, each participant controls a private workspace, access to which may be granted or denied to any subset of other participants. For entities in private workspaces to
become a part of the current state, they must eventually be incorporated in the public workspace. The entities in the workspace may be either declarative or procedural. Examples of declarative entities include spatial and non-spatial requirements, primitive shapes, instantiations of shapes, and architectural spaces. Examples of procedural entities include proposals for action and calls for votes.

As illustrated in figure 5.3 below, a collaborative design activity is envisioned as an evolving, but iterative process of divergence from an initial agreed-upon state to multiple concurrent states and their subsequent convergence to a single state through a process of evaluation or by the actions of agents with authority such as the G.O.D.

![Figure 5.3: Progression of design through divergence and convergence.](image)

**SYCODE Implementation**

The initial experimentation phase involved the installation and testing of the *Share-Kit* on different hardware platforms at the University of Michigan at Ann Arbor and the Chinese University of Hong Kong, the definition of an initial protocol for interaction, and the independent development of two software modules that can communicate by sharing a common protocol. The user is presented with a dialog box that specifies the server machine name, the session name, and an identifying alias. After establishing a connection to the server, the user can create different entities in the graphic window (see Figure 5.4 below). A second client, connecting to an existing session,
receives a complete and synchronized copy of the current state, after which actions taken by any of the connected clients are broadcast to each (see Figure 5.5 below). Consistency in the chronology of actions, as seen by each client, is maintained by routing all actions through the server. Thus, client actions that modify the current state do not take effect until distributed by the server and echoed back to the originator.

In informal tests, the transmission of an action between Ann Arbor and Hong Kong consumed an average of three to four seconds for a one-way trip (six to eight seconds for a round-trip). A client having a faster network path to the server experienced almost no delay between initiating an action and getting feedback. On the other hand, a client having a slower connection to the server experienced a time lag during which its action produced no feedback until the server distributed the request to all clients and finally echoed it back. To improve local user interaction, the client was modified to provide immediate graphic indication of pending actions. As network capacity increases problems with time lag should diminish.

Figure 5.4: A SYCODE client running on a NeXT Workstation.
Figure 5.5: A SYCODE client running on a Silicon Graphics Workstation.

**Discussion**

The SYCODE experiment essentially corroborates the assertion that a common protocol of interaction is a basic requirement for collaboration. The SYCODE clients were able to share and represent a common set of actions and messages that were specified through e-mail conversations. Aside from communication and interface issues, if one client could not understand the received message, no further collaboration could take place. It is important to note here that what is being communicated is a set of deep data structures and not simply strokes and shapes. As such, SYCODE is put forward as a domain-specific tool rather than a general purpose shared whiteboard. As will be illustrated in later chapters, the foundational ideas in SYCODE were further deployed by the author in a variety of prototypical and real-world systems.
CHAPTER VI
GENERAL PURPOSE VS. DOMAIN-SPECIFIC GROUPWARE

Introduction

In the mid 1990’s the World Wide Web (WWW) and the Hypertext Markup Language (HTML) emerged as a powerful new shared environment and a language for publishing information respectively. It became apparent from the outset that the web would become the standard basis for collaboration. It also became apparent that technological advances and higher bandwidth could serve the unique nature of architectural design such as its requirement for specialized vertical knowledge that goes beyond the sharing of marks on paper or the multi-casting of video images.

This chapter briefly surveys current commercial groupware applications and outlines the need for vertical and integrated support of synchronous and asynchronous design collaboration. This section also describes WebOutliner, a software prototype developed by the author that uses a three-tier persistent object-oriented, web-based technology for a richer representation of hierarchical architectural artifacts using Apple’s WebObjects technology. The prototype contributes to earlier work (Jabi 1996a, 1996b, 1998) that defined a framework for a shared workspace consisting of Participants, Tasks, Proposals, and Artifacts. These components are also hierarchical which allows users to filter information, analyze and compare design parameters and aggregate hierarchical amounts. Given its object orientation, the represented artifacts have built-in data and methods that allow them to respond to user actions and manage their own sub-artifacts. In addition, the prototype integrates this technology with Java tools for ubiquitous
synchronous web-based access. The prototype uses architectural programming (defining the spatial program of a building) and early conceptual design as examples of seamlessly integrated groupware applications.

**Examples Of Current Groupware Applications**

Software that supports collaborative work (groupware) usually belongs to one of two categories: synchronous and asynchronous. Synchronous groupware supports the real-time aspects of collaborative work such as group meetings. Examples of synchronous groupware include chat programs, video-conferencing, application sharing, and shared whiteboards. Asynchronous groupware supports the longer-term aspects of collaboration such as workflow and project management, issue discussions, and review processes. Examples of asynchronous groupware include e-mail, newsgroups, shared workspaces, and task management software. Some aspects of collaborative work, such as voting, can happen either synchronously or asynchronously. Those types of activities need software support that is flexible enough to operate in either mode.

Given the lack of bandwidth, real-time videoconferencing has met limited success and is only available to large corporations and to a relatively small number of distance-learning centers within academic institutions. Instead, many users of the Internet rely on desktop conferencing applications such as *CU-SEEME* by First Virtual Communications, Inc., *NetMeeting/MSN Messenger* by Microsoft, *iChat AV* by Apple, or *Instant Messenger* by AOL for near real-time audio and video communication (see Figure 6.1 below). *NetMeeting/MSN Messenger* also has whiteboard capabilities, the exchange of files and the sharing of applications. Aside from reliability problems and network congestion, *NetMeeting/MSN Messenger* on its own provides only general-purpose collaboration functionality. The sharing of single-user applications is not the most effective way to multi-author a document. *NetMeeting/MSN Messenger* shines in its one-
on-one videoconferencing capability. However, *CU-SEEME* has the advantage of its ability to multi-cast several video images such that a group meeting can take place.

A new breed of web-based collaboration technologies is based on the idea of a shared virtual workspace. In some cases, the application may use a synthetic three-dimensional space that multiple users can inhabit and meet through a representational avatar. In other cases, these applications, such as the one *webex.com* offers, allow you to have a virtual office (see Figure 6.2 below). Using a web browser, users can upload documents, leave messages, request meetings, check shared calendars, and conduct real-time meetings using chat software, videoconferencing, a shared whiteboard, and multicasting of *PowerPoint* presentations.

![Microsoft NetMeeting Screen Shot](image)

Figure 6.1: Microsoft *NetMeeting* screen shot.
With the advent of the Java programming language, several smaller shareware applications (applets) are available that allow whiteboard functionality, document annotation, and real-time textual chat. An example of that class of software is Groupboard from www.groupboard.com (see Figure 6.3 below). Again, this type of application is suited for casual meetings that need only to sketch and chat without the need to carry out organized tasks that need any specialized software.

While current commercial applications may support one or more collaborative functions, many have failed to fully integrate the synchronous and asynchronous requirements of collaborative work. Furthermore, the majority of groupware applications support general-purpose collaboration and, thus, are not equipped to support domain-specific functionality. The next chapter more fully explores the benefits and limitations of general purpose tools by deploying them in real-life (as opposed to laboratory) settings. The next section, however, reports on software developed by the author to address the aforementioned shortcomings – the lack of seamless integration of synchronous and
asynchronous collaboration, and the lack of domain-specific functionality. The software is built using *WebObjects* from Apple Computers, Inc. Thus, the next section starts with a brief introduction to the *WebObjects* platform.

![Groupboard screen shot](image)

**Figure 6.3:** *Groupboard* screen shot.

---

**WebObjects**

Apple’s *WebObjects* software is an application server, an object-oriented software development environment and a database integration tool (see Figure 6.4 below). It provides a framework for developing three-tier client-server applications as well as dynamic publishing (see Figure 6.5 below). The three tiers are (1) Storage, (2) Application-Layer, and (3) Web Interface.

![WebObjects tool set](image)

**Figure 6.4:** Apple *WebObjects*’ tool set.
Figure 6.5: Apple WebObjects’ 3-tier system.

By using components in templates that are linked to actions and data in the WebObjects application, the server can act on user requests dynamically, configure a response HTML page and return that to the user for further action. WebObjects provides mechanisms for maintaining information, called states, during a session or even after a session has terminated. Because of that functionality, multiple participants (clients) can view and edit the same information while being logged on to the same WebObjects application. In addition, WebObjects interoperates with various database systems through a tool called Enterprise Object Modeler (EOModeler) that maps the relationship between a database and a set of objects. Basically, an enterprise object corresponds to one or more records in a database. The WebObjects application manages these objects such that the developer can concentrate on the business-logic of the application at hand. WebObjects’ ProjectBuilder allows the developer to create the necessary software to handle the
business logic of the application in question. Finally, a WebObjects’ InterfaceBuilder constructs the necessary web templates and connects its elements to software variables and methods.

**WebOutliner Implementation**

In contrast to general-purpose commercial systems, WebOutliner was developed by the author to explore the merits of the proposed theoretical framework (see the previous chapter) and to analyze the difference between general-purpose tools and domain-specific ones. Additionally, WebOutliner was used to further explore the role of artifacts as enablers of asynchronous and synchronous collaboration.

Fundamentally, WebOutliner is a web-based application that represents and manipulates hierarchical elements using Apple’s WebObjects technology. WebOutliner is composed of reusable components that can be adapted for several needs. As we will see in Chapter VIII, many of those components were re-used in other domain-specific systems. In its current implementation, WebOutliner manages the components of a building program. Following an object-oriented approach, WebOutliner’s components belong to one of three categories: (1) Model, (2) View, and (3) Controller. The Model is mainly composed of a Tree-like object that stores data and methods that represent an architectural space. A space can contain within it other sub-spaces and so on. The total area of a space is computed as the sum of the areas of its descendant spaces plus an additional area that is unassigned to any of its descendants. A space manages its own sub-spaces. For example, if a space is copied and pasted as a sub-space of another space, it will copy and paste with it its own entire descendant sub-spaces. In practical terms, a user can copy a whole department with multiple sub-rooms from one level and paste it on a second level. All the sub-rooms will be copied with it. If a space is edited, added, or deleted, the area of its parent space will be re-computed to reflect that change.
Asynchronous Collaboration

When a user connects to *WebOutliner*’s web site, the current status of a building program is displayed. The user is asked to enter a name for identification purposes. A magnifier button is provided that enables the user to search for a particular item in the hierarchy (see Figure 6.6 below).

![Initial WebOutliner screen](image)

Figure 6.6: Initial *WebOutliner* screen.

Next, a hierarchical list is presented. By clicking on the triangle next to an item, a user can collapse or expand an item to reveal or hide its sub-items (see Figure 6.7 below). By clicking on the underlined item name, that item is selected and a toolbar of options is displayed that allow the editing and manipulation of that item (see Figure 6.8 below). The user is able to copy/cut/paste/delete items to edit the hierarchy, and can re-order the items that have a common parent item. The user can also zoom-in on an item temporarily making it the root item. Using this mechanism, a user can filter the overall building program and concentrate only on part of it (see Figure 6.9 below). By clicking on the Edit...
Item button, a user can change the name of an item and enter a numeric value. In this example, a numeric value is used to represent area (see Figure 6.10 below). Obviously, it is a simple matter to extend the application to include other parameters such as cost and volume. Once the area is entered, the overall area is aggregated using a recursive algorithm (see Figure 6.11 below).

Figure 6.7: Expanded building program
Figure 6.8: Toolbar and commands.

<table>
<thead>
<tr>
<th>Name</th>
<th>Amount</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Level2</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.1 Lecture Hall</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.2 Restrooms</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.3 Storage</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.4 Studios</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.4.1 test studio</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.4.1.1 Untitled</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.4.2 Workshop</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 6.9: Zoomed-in on a sub-item to filter out information.
Figure 6.10: Editing name and area of an element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Amount</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS</td>
<td>0.00 sq. ft.</td>
<td>3,000.00</td>
</tr>
<tr>
<td>1 Level1</td>
<td>0.00 sq. ft.</td>
<td>3,000.00</td>
</tr>
<tr>
<td>2 Level2</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Amount</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Lecture_Hall</td>
<td>800 sq. ft.</td>
<td>800.00</td>
</tr>
<tr>
<td>2.2 Restrooms</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.3 Storage</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.4 Studios</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
<tr>
<td>2.4.1 test</td>
<td>0.00 sq. ft.</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 6.11: The building program computes and maintains an aggregate area.

Seamless Transition To Synchronous Collaboration

By clicking on the upper left button, a user can go to that item’s page to examine it in more detail. It is in this simple move that a user transitions to synchronous collaboration. If another user happens to perform the same function (i.e. both participants
are interested in the same item at the same time), a page will be presented that contains real-time Java applets for chat, and a shared whiteboard. Users are immediately aware of who else is working on this item and can communicate in near real-time (see Figure 6.12 below).

The power of WebObjects is evident here in that the design of the application only has one template for the item’s page. However, by assigning the item’s unique ID to that template, multiple users can synchronously collaborate on multiple items at the same time. For example, John and Mary can be collaborating synchronously on the design of the Lecture Hall while Ed and Jane can be discussing the design of the Restrooms. At the same time, Bob can add more spaces to Level 1.

It is important to note that, currently, the represented artifact is a simple GIF image. Yet, theoretically any synthetic artifact can be uploaded embedded in this page, and displayed using the proper plug-in (e.g. VRML file, animations).

Figure 6.12: Synchronous collaboration on an item in the hierarchy.
Summary

This chapter illustrated the need for domain-specific collaborative software. Integrating a database, an intelligent object-oriented application server, and a web-based interface proved to be a powerful and flexible solution that can be used to customize behavior and add intelligence to shared artifacts. As we will see in Chapter VIII, the same setup was found to be effective in addressing real-life collaboration problems and two systems were deployed in real-life settings that used the same solution as WebOutliner and re-used many of its concepts and components. The specification of a building program is only one example of a domain-specific functionality. This new class of tools should be regarded as complimentary to current general-purpose tools.

WebOutliner is a system that uses intelligent artifacts as the main focus of collaboration. Furthermore, interest by the participants in these artifacts allows for a seamless transition from a private asynchronous mode of work to a shared synchronous mode. Artifacts in WebOutliner intelligently manage – and query – their own sub-artifacts to maintain overall system consistency and report a the overall aggregate condition to the system’s users. WebOutliner is also a system that enables information filtering by zooming in on a sub-node and making it the root of the hierarchy. As such, a user can work on that node and its descendants regardless of other nodes in the hierarchy.

WebOutliner was developed as one experiment within a larger set of experiments that attempt to provide a domain-specific, intelligent, and shared workspace for collaboration. As the reader will see in Chapter VIII, the same technology – an integrated database, an object-oriented application server, and a web-based interface objects – was found to be effective in solving real-life collaboration problems. Furthermore, the methods developed in WebOutliner were re-used in these systems, but extended to add the ability to address issues of hierarchy, privacy, and controlled access to shared information. The added dimensions of competition and evaluation are also addressed in
these system as important aspects of collaboration. However, before those systems are presented, the next chapter steps back to report on two experiments that use software tools that are not domain-specific. A case is made in the next chapter that both general-purpose tools and tools designed for other domains can be effective in supporting collaboration in some aspects of architectural design, but are fraught with extensive limitations.
CHAPTER VII
IMMERSIVE SYNCHRONOUS DISTRIBUTED DESIGN COLLABORATION

Introduction

With the wider availability of high-bandwidth communication networks and the maturity of commercial collaboration software, schools of architecture are experimenting with computer-aided distributed design collaboration. The most common use of this technology is to support distributed design reviews. A distributed design review enables geographically-distant participants to discuss a common design project using computer-supported collaborative technologies such as videoconferencing, voice over IP, and shared applications. A more recent development is the use of immersive virtual reality environments to inhabit, navigate and share designed virtual environment. This chapter will describe two lines of research: The first is to use interactive and immersive hardware coupled with more traditional videoconferencing and virtual meeting software. The second line of research describes the use of modified gaming engines for use as immersive, real-time, distributed and co-inhabited design environments.

While potentially beneficial to students, and attractive to teachers, there are a number of challenges facing the integration of synchronous distributed design collaboration into the design studio by technically inexperienced faculty without significant technical support. In order to support the claim that domain-specific tools are required to effectively support collaboration in architectural design, this chapter outlines the potential and limitations of using commercial software that was not intended for use in the applied domain. In particular, the first experiment emphasizes the limitations of
adapting general-purpose software to a specific domain while the second experiment emphasizes the difficulties of adapting a technology created for a different field (gaming) to a new application (distributed design reviews). The two experiments reveal that while collaboration processes are fundamentally the same across domains, the specific content of collaboration is different and thus requires specific solutions. Yet, the potential of these new technologies should not be ignored. The two experiments illustrate that although limited, the technologies can be used in distributed design reviews. This chapter describes a number of real-world reviews, in a variety of contexts, distilling a set of guidelines and recommendations for future users of this technology.

**Background and Motivation**

Design education relies heavily on various types of conversation between design students and instructors, or mentors, including lectures, small group discussion, and design reviews. Design reviews are one of the most important forms of pedagogical communication between design instructors and students (Cuff 1993). These conversations between the designer and one or more reviewers fall into two broad categories: desk crits and formal reviews. *Desk crits* are informal (but not necessarily casual) exchanges, usually held on a regular basis, where the studio instructor examines the student’s recent work and discusses strategies, plans, and challenges. It is commonly held at the student’s desk. The conversation is primarily between these two principals, but may expand to include others. That is, while nominally private, eavesdropping is encouraged.

In contrast to desk crits, *reviews* are formal events, generally held at important project development milestones (including end of term). They often take place in special review rooms or areas and involve formal (stand-up) presentations by each individual or group that they have worked hard to prepare. The principals include the presenting student(s) and a group of invited guests (the jury). Conversation sometimes strays from
consideration of the current project, involving give and take between guest critics. Classmates of the presenting student(s) are expected to pay close attention to the discussion.

Both types of design review are collaborative conversations. That is, they involve the input of and communication between at least two individuals: the student and the instructor, with appropriate turn-taking, both verbally and graphically. In fact, justification for the design studio teaching strategy often relies on the aggregate studio culture created by successive shared and overlapping design conversations.

Studio instructors occasionally travel, or support practices in two cities, taking them out of town on a weekly basis. They might wish to follow and review their students’ work from afar, rather than compressing or eliminating studio hours. Conducting some or all of the normal desk crits via Internet connection would be a significant benefit to these faculty and their students.

Formal reviews benefit tremendously from guests who bring with them informed and articulate alternative views. These special viewpoints may be essential to all phases of the design development for a community-based project or a design for a different culture. However, financial and logistical difficulties, that are common when trying to invite local jurors to one’s design review, are often insurmountable when the potential jurors are at great distance. The ability to use the Internet to involve remote expertise at a minimum cost would significantly expand the pool of candidate reviewers.

With the wider availability of high-bandwidth communication networks and the maturity of commercial collaboration software, schools of architecture are experimenting with computer-aided distributed design reviews. A distributed design review enables geographically-distant participants to discuss a common design project using computer-supported collaborative technologies such as videoconferencing, voice over Internet protocol (VoIP), and shared applications.
While potentially beneficial to students, and attractive to teachers, there are a number of challenges facing the integration of synchronous distributed design reviews into the design studio by technically inexperienced faculty without significant technical support. The experiment reported in this chapter illustrates some of the potential pitfalls to avoid and distills a set of guidelines that will help faculty make routine utilization of such reviews.

The use of distributed design reviews also introduces students to an environment which challenges them to construct effective and communicative presentations consistent with likely future professional requirements, but quite different from traditional, paper-based presentations. Donald Schön has identified talking and drawing as the two most fundamental components of what he termed a language of design (Schön 1983). A distributed design review provides a means to study the interconnectivity of talking and drawing and to compare it to what usually happens in traditional design settings. This chapter concentrates on these two activities as they occur during distributed design review sessions.

**Related Work**

In this project, two main issues were of interest: 1) Making optimum use of the Internet to link remote jurors to a formal review by identifying the most important features of their interaction, and 2) Simplifying and rendering more natural the process of turn taking in both the verbal and graphical exchanges during reviews, minimizing the attention required by the interface. This work builds on that of a number of other researchers.

While this chapter concentrated on design reviews, a narrow and well-defined activity within the design studio, the work is part of a larger interest in researching the requirements for an effective distributed collaborative design studio – or what William J.
Mitchell referred to in 1992 as the Virtual Design Studio – or VDS for short (Wojtowicz, Davidson and Mitchell 1992). The general requirements for an effective computer-supported cooperative design system have been outlined by several researchers (Tang and Leiffer, 1988; Jabi, 1996; Turner and Cross, 2000). The VDS project has described many of the opportunities and pitfalls of internet-based collaboration. In the first VDS experiment, students at two different institutions were asked to co-design a project, review and exchange comments using e-mail and conduct an electronic, long distance design review. The authors viewed asynchronous digital correspondence as vital to the collaborative process but de-emphasized the use of synchronous tools. This may have been due to the duration of the project, the difference in time zones as well as the relative immaturity of real-time synchronous tools at the time the experiment was conducted. In 1994, a year after the completion of the first VDS studio, a second VDS studio was conducted involving five time zones and included students and tutors from six different institutions (Wojtowicz 1995). VDS 94 clearly illustrates the use of synchronous videoconferencing as an effective tool for collaboration. Yet, asynchronous communication remained an important tool for the VDS project. Wojtowicz said it simply and clearly: “E-mail exchanges were the life-blood of the Virtual Design Studio.”

Later researchers have pursued various aspects of design collaboration including comprehensive collaboration and review environments that have been created to support VDS projects (Wenz and Hirschberg 1997; Kolarevic et al. 1998; McCall et al. 1998; Maher M.L. and S.J. Simoff 2000).

While this small-scale project concentrated on more traditional presentation techniques (still and video images using Microsoft PowerPoint and simple web pages), others have described the potential and pitfalls of multi-modal, 2D and 3D immersive techniques. For example, A paper by Daily et al. describes an integrated immersive virtual reality environment that supports multi-site connections using avatars, high-fidelity audio, and shared artifact manipulation. They conclude in their research that
while it is possible to communicate design concepts and ideas with a natural, intuitive interface over great distances, a key future goal would be to overcome the widely disparate interface modalities they encountered (Daily et al. 2000). Their findings corroborate one of the hypotheses in this dissertation – mainly that to ensure their success, collaborative systems should accommodate a variety of hardware and operating system platforms (see Chapter V). Gross and Do have described a number of shared drawing (Qian and Gross 1999) and design annotation systems (Jung et al. 2002), while Johnson has described a formal review tool used in online competitions (Johnson 1999). Kalisperis et al. also report that Virtual Reality (VR) setups can be affordably constructed and implemented in the initial phases of the design process (Kalisperis et al. 2002). Their research intersects with this research in that both are interested in an immersive and natural environment for design presentation. Both believe that the technology belongs in the studio and should be made available to students throughout the design process. However, the Kalisperis et al. work concentrates on the visualization of design, rather than the conversation of the remote reviews.

**Experiment 1: Distributed Design Reviews Using General-Purpose Collaboration Software**

The review events involved about 15 students, two or three co-located jurors, and one remote juror. The hardware setup for this experiment included a large interactive whiteboard (Smart Technologies, Inc. SmartBoard 580) that served as a surface for both user input and display (see Figure 7.1 below). The SmartBoard software allows the screen image (as projected by a computer projector) to be synchronized with the touch-sensitive screen of the SmartBoard. Thus, the presenter is able to interact directly with the large projected image without requiring the use of a mouse or keyboard. Additional hardware included computers with an Internet connection at both ends that were connected to speakers and microphones to enable a voice conversation. After
experimenting with a shared microphone as well as multiple microphones, it was concluded that one high-quality, omni-directional microphone is sufficient to capture all voices within a medium-sized room (approximately a 12 foot radius). Since the software also provides a separate video channel, the use of video through webcams was tried, but was found to be of little use other than to initially introduce the participants to each other.

The software included readily available collaboration software that allowed shared audio and video, shared network sketching, and shared applications (Microsoft NetMeeting/MSN Messenger). The NetMeeting/MSN Messenger software allows participants to share almost any Windows-compatible application, which in this case was Microsoft PowerPoint, which the students used to assemble their presentations. Sharing applications simply means that the remote participant can view and optionally control a single graphical user interface, visible in both locations. Effectively, the remote participant can follow a presentation as it progresses, can see the marks and annotations being made and can (optionally) take control of the presentation and create marks. As Donald Schön predicted, all that was needed was the combination of talking and sketching. Coupled with voice, this technique was effective in carrying out a fluid design discussion.

The large size of the projected presentation engaged the local jurors and audience. This also mimicked traditional stand-up presentations in which the presenter stands in front of and points to the presentation board. As mentioned above, the SmartBoard provides a projection surface and also acts as a large digitizing tablet. The software essentially creates a virtual transparent layer on top of the projected image, on which one may mark using one of three color-coded ink-less pens. The software recognizes, through a sensor built into the board, what pen is being used and creates marks on the screen of the correct color. An eraser works in a similar, but reverse fashion. The fact that the marks are projected back unto the screen and spatially aligned gives the illusion that the user is sketching in situ with virtual ink. Alternatively, if the user presses on or drags
across the screen with a finger, the SmartBoard simply passes the event through to the operating system, which treats the action as a mouse button press or drag and responds accordingly. The resulting effect is a flat 2-dimensional version of a hybrid physical-virtual environment – a cybrid (Anders 1998).

The remote juror is not required to have a comparable setup. At a minimum, a standard Windows desktop PC with a mouse and keyboard is required. If available, the juror would benefit from the use of a small digital drawing tablet which provides a much more natural interface for annotation and sketching.

Figure 7.1: A Design review using an interactive SmartBoard.

Case Study

As discussed above, design reviews tend to vary along two dimensions. The first dimension indicates the degree of formality while the other dimension indicates the geographic location of the participants. These two dimensions yield four possible design...
review settings (see Figure 7.2 below). The setup described above has been used and documented via still photography and digital video for one semester in third year studio in three different settings. The SmartBoard was made available in studio for weekly informal use for desk crits. It was also used in studio for more formal design reviews that did not include a remote juror. Finally the setup was moved twice to a review room to be used for a formal design review that included three or four invited jurors – one being a remote juror. The fourth alternative, informal distributed desk crits, was not evaluated in this sequence. It is expected to be the subject of future research.

<table>
<thead>
<tr>
<th>Informal Local Desk Crits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In studio, the SmartBoard was used occasionally as a replacement for trace paper and pens. Although no remote capabilities were utilized, the studio instructor and student would stand in front of the board, discuss and sketch over the drawings. This proved to be an effective method in discussing the project at hand and did not require a lot of preparation. The media used included static images stored in folders, PowerPoint slides, AutoCAD drawings, 3D Studio MAX models, AVI animations, and spherical panoramas.</td>
</tr>
</tbody>
</table>
Control of the interface, while involving new paradigms, proved to be intuitive and the ability to non-destructively sketch over any image or model proved to be useful. In their end-of-term evaluation forms, all students commented that the use of the SmartBoard was a valuable addition to the pedagogy of the studio.

Local Formal Design Reviews

In studio, the SmartBoard was used occasionally for design presentations. The students prepared PowerPoint slides and used the board mainly as a projection screen. Again, the added ability to non-destructively sketch over the drawings in a natural manner enabled a more fluid presentation. The ergonomics were improved since the student did not have to fiddle with a mouse and keyboard and could stand in front of the screen and face the audience in a natural position.

Distributed Formal Design Reviews

As mentioned above, The SmartBoard and collaboration software setup was used twice during the term to conduct formal reviews. The first session (mid-term) was conducted with a remote juror in the same country, but with a 3 hour difference in time zones. The second review session (conducted at the end of the term) involved a remote juror who resides in a different country with a 7 hour difference in time zones. In both cases, the remote reviewer was connected via voice and application sharing. Below are some brief observations on those two sessions:

A slight operating system incompatibility necessitated the use of an older version of the collaboration software to initiate the review session. While workable, this solution was not immediately obvious and probably represents a weak point for the less technically experienced. While it is of concern that Netmeeting/MSN Messenger is
platform-specific, it was found to be more integrated and functional than available platform-agnostic counterparts.¹

Initially, the remote reviewer had microphone problems. This was temporarily resolved through the use of a mobile phone, but the lack of projected audio transformed the review into a one-on-one desk crit because the audience could not hear what the remote reviewer was saying. This was so unsatisfactory that the remote review was suspended until he located and connected another microphone to the computer.

There is a lag inherent in VoIP technology. In addition, the remote microphone may pick up some of the audio signal and play it back to the speaker. Due to the lag, it arrives back at the speaker’s end 1.5 to 2 seconds later. Initially, this caused critics and students to speak much more slowly and deliberately – artificially slow. Adjusting the volume and exercising the ability to mute the microphone when not speaking solved the problem.

While testing prior to the second review session, the network security implementations (firewall) prevented the collaboration software from functioning properly. Network administrators could not solve the problem in time. However, the problem was addressed by issuing the remote reviewer a temporary password to a Virtual Private Network (VPN) connection that allows the connecting computer to behave as if it was placed on the local network behind the firewall.

In session #2, the remote juror felt it necessary to speak faster, keep comments short and to the point, ask permission to start speaking and explicitly indicate the end of voice transmissions. However, the exchange did become more fluid towards the end of the review with natural interruptions, juror-juror comments and jokes.

In session #2, the local jurors almost never went up to the board to sketch or gesture. Instead, they either used the shadow of their hand in front of the projector or a laser pointer. Obviously, these gestures are not transmittable. Thus, the remote juror could not easily follow the parts of the drawing the local jurors were commenting on.
This indicates the need to find a way to remotely, wirelessly, and easily enable local participants to affect the screen. To compensate, the manager of the review tried as much as possible to follow the laser point with the cursor icon. In contrast, the student presenters used the SmartBoard to its full potential with sketching and interactive mouse clicks.

Animations embedded in PowerPoint caused a long delay and breakdown. However, they were shared separately and the report from the remote juror is that they worked reasonably well although a bit choppy.

Video (headshot) was tested and worked, but found to be unnecessary for an effective review. It consumed too much bandwidth and voice was found enough to convey meaning. Video was mainly used to introduce the two parties to each other.

Students and faculty informally reported that the whole process, while a bit strange due to its newness (e.g. one juror referred to the remote juror as “the voice”), was useful and informative.

The remote juror did not clearly know who was speaking resulting in use of sentences such as “the critic who last spoke” or “I agree with the comment just made...”. It might be helpful to have individual microphone channels that, when activated, display on the screen the identity of the person speaking.
One juror complained about the glare coming from the hotspot generated by the projector. It is hoped that a rear-projection system, as illustrated below, will solve that problem along with the shadow problem commonly associated with front-projections (see Figure 7.3 above). A rear-projection screen was recently purchased and briefly tested. Initial results indicate that it solves the above-mentioned problems.

**Recommendations**

Based on the experience with design reviews, several recommendations have been derived to help avoid some of the most common pitfalls associated with conducting distributed design reviews and make them productive conversations:

*Start Early (Plan Ahead).* While it may take only a matter of minutes to establish the review connection in the end, it may take several days, or even weeks, to verify the
preconditions for the review (room reservation, camera function, software, microphones, speakers, etc.). Similarly, it is necessary to get placed on the remote reviewers’ calendar early, to make sure that they have allowed adequate time with a minimum of distractions. 

Extensively test the setup. Modern systems are a complex of software and hardware, often involving many layers of data exchange and encoding, plus filtering by internet routers and firewalls. They are also a moving target, as university staff move to control hacking, or software piracy, and vendors sometimes rush products to market before they’re ready. While it has become significantly easier to make viable connections in the real-world, including those between heterogeneous systems, each component should be adequately tested to assure run-time viability for the review. An OS update may leave you unable to use your camera driver, or you may have to establish how to get NetMeeting and Messenger to talk to each other.

Familiarize yourself with the other participants. In the normal course of a face-to-face review, presenters and reviewers each have a part to play and each knows their part. Nonetheless, a significant amount of adjustment may take place in the review process due to the exchange of social cues at the review. In the case of a remote review, it is almost certain that the reviewer will not be familiar with the local scene. It may be necessary to address some of these issues in advance, in an explicit form, or protocol that sets out the timing of the review, the order of the students, the amount of time each has to present, the biases or spins being supplied by the studio mentor(s), and so on. These help set the stage for responding to the student work.

Use back-channel communication. During the review event it may become necessary for a remote reviewer to communicate with the host about the review. While it may be necessary to interrupt the presentation or conversation underway, it is quite disruptive to do so. Having an alternative back channel communications means (email, un-projected chat window, phone, etc.) provides for this kind of communication.
Adjust Expectations. Remote reviews take longer than their face-to-face counterparts. In the face-to-face review process, each student receives the focused attention of the reviewers for a period of time, but it is still possible to handle three to four reviews an hour. During remote reviews, you should expect to do two or three in an hour, rather than three or four.

Limit number of participants. These technologies are not well suited to large-scale reviews. In particular, reviews involving multiple remote reviewers are ill-advised because the social cueing that is necessary for a smooth conversation is disrupted by the time-lag found in voice-over-IP. For one remote reviewer it is likely that an informal wait protocol will evolve—analogous to that seen on TV when a reporter is half-way around the world—using long silences to signal conversational turn-taking. With two or more remote reviewers, there is no acceptable means of cueing turn-taking between the remote endpoints, with the result that the reviewers will often collide with and talk over each other.

Learn the software. In an environment which is as dependent on technology as is a remote design review, it is important that all participants have experience with the software and the specific way in which it will be used during the review, even if for only a few minutes. Each student, and the supporting faculty and remote reviewer, should practice with the software prior to the review. During this time it is possible that anomalies will be found (playing movies in PowerPoint, etc.) which profoundly affect the presentation plans of the presenters.

Encourage symmetry. One-way mirrors may connect two occupied spaces, but they don’t provide both occupants with the same sense of social presence. It’s important that the tools be set up and configured in such a way that every participant feels they are a part of the exchange.

Try to use high bandwidth connections when available. While viable reviews can certainly be completed using slower speed or even asynchronous review protocols, additional bandwidth provides the flexibility to conduct a review without carrying the
same cognitive load that slower-speed connections require. At the same time, almost any connectivity speed can be overwhelmed by an overly ambitious project (e.g. 1024x768x30fps animation).

*Design for graceful degradation.* You should expect one or more aspects of your setup to fail. To the extent that is possible, have backup plans in place for continuing the review (placing PPT files on the web, with telephonic, chat or email reviews, having a reserve of redundant equipment, etc.)

**Discussion**

Despite the inherent technical difficulties associated with introducing new technologies and processes, it is firmly believed that the ability to regularly conduct distributed design reviews is an invaluable opportunity for both faculty and students – if implemented with care. One fundamental challenge is to understand the degree to which a distributed design review is, or is not, similar to a traditional face-to-face studio review. For example, one may find fault in the apparent linearity of online presentations. Yet, business presentations and lectures are generally linear as are most, but not all, written work. The ability to move laterally between images and components can be solved with multiple instances of software running and even in *PowerPoint* with more effective use of the slide sorter screen to which critics may refer. While face-to-face reviews, and many distributed reviews, occur at the same time (synchronously), asynchronous web-based design reviews have also been gaining popularity. They provide a method to continue design work outside normal studio hours and at a distance, include a larger set of viewpoints, compensate for time-zone differences, and acknowledge varying connection speeds. Yet, one of the described case studies in this chapter points to a benefit of conducting synchronous design reviews across different time zones. Specifically, the use of multiple time zones may actually facilitate synchronous reviews.
An overlap of available time slots when working with critics of other time zones would occur when working in design studios that require consecutive large blocks of time (generally three to five hours). So, this process may actually expand the pool of available critics, all without significant time spent on travel to and from remote locations.

While conducting several distributed design review sessions it was deemed that the use of video was not essential to the progress of the review. Accordingly, to optimize the use of the Internet connection, a video connection was largely not used. In contrast, the use of audio was found to be essential, even critical. The complete loss of audio and even the degradation of audio severely hindered the progress of the review. Therefore, an optimum use of the Internet link requires the allocation of appropriate bandwidth to and a robust implementation of the voice channel. Given the direct manipulation capabilities of the SmartBoard screen and the fact that the projected image occupies the full screen, the computer interface virtually disappeared. The participants experienced only the full image and the voice of the reviewers rather than the computer hardware and the myriad of buttons and dialog boxes usually associated with a graphical user interface.

In conclusion, this experiment illustrated the advantages and limitations associated with general-purpose collaboration tools when adapted to a specific domain. As long as designers and architects continue to use and adapt general purpose tools, these limitations will continue to exist. As illustrated in the next section of this chapter, tools that are domain-specific can sometimes be adapted for use in another domain. Yet, such an adaptation is fraught with problems similar to – and sometimes more severe than – those reported above.

**Experiment 2: Distributed Design Reviews Using Modified Gaming Engines**

While the first experiment addressed the issue of adapting general-purpose tools to the domain-specific problem of distributed architectural design reviews, the second
experiment addresses the issue of adapting one domain-specific technology for use in another domain. Specifically, this experiment addresses the issue of modifying commercial gaming engines for use in distributed design reviews. While traditional 3D modeling and visualization systems such as 3D Studio MAX and *form•Z* offer increasingly convincing visual simulations, gaming engines are approaching the visual realism of such systems and are offering additional interactive features that are usually available only in more expensive immersive virtual reality systems. Additionally, the capability to have multiple individuals inhabit and navigate the space offers unique opportunities for collaboration as well as the investigation of human behavior. Participants with internet access can be invited to access a shared virtual environment. Collaboration among users can be further enhanced by combining immersive navigation with peer-to-peer instant messaging and/or adding a voice channel. Video games have piqued the interest of a new generation weaned on electronic media, including budding architects and designers. In much the same way that motion pictures with their elaborate, evocative lighting, sound design, and simulated motion through space are engaging experiences, video gaming provides an immersive experience, suspension of disbelief, and transport to a virtual and usually fantastic environment. Video games differ from the movies, however, in that they are interactive and interrogative, allowing a level of control over the participant’s presence in the simulated environment.

**Early Research**

An experiment was conducted in an effort to better understand the usefulness and limitations of gaming engines in architecture. In all, four game engines were examined (*Doom II, Quake I, Quake II, and Quake III*) for their capacity to represent both existing
spaces and conceptual spatial designs. Specific aspects of potential application to architectural modeling and presentation are outlined and summarized.

**DOOM II**

*Doom II* was released in 1994 by id Software as a first-person perspective shooter that offered four degrees of visual freedom – the software offered no ability to look up and down (see Figure 7.4 below). In 1995, to test whether the game engine was compatible with the depiction of traditional space, the creation of a game environment of the interiors of two connected buildings, were modeled. The spaces were chosen specifically for easy access, which facilitated continuous measurement for accurate documentation. Close proximity also allowed an ongoing comparison of spatial qualities between the actual space and the space depicted in the game environment. Students were enlisted as volunteers for this documentation phase.

Once level-editing software was successfully installed (not an easy task), the modeling phase was surprisingly straightforward, but at the same time very limiting. *Doom II* is actually a two-dimensional engine that simulates 3D by vertically projecting 2D texture information. Photographed images of local textures were cropped, resized to a maximum 128x128 pixels, and re-sampled into the 8-bit *Doom II* palette. This procrustean manipulation obviously stressed the original image a great deal and necessitated several cycles of judicious doctoring and field testing before the mapped surfaces rendered satisfactorily.

Because the *Doom II* engine uses a simplified lighting model and is not a polygon-based engine, the time required to compile editable maps to game files was very short (<5 minutes on a dual processor 90 Mhz Pentium). This was useful in that it facilitated the refinement process by offering almost immediate feedback. In comparison
to still-image rendering time requirements of the day, the *Doom II* engine offered more experientially gratifying results, despite overall deficiencies in visual richness.

Figure 7.4: Screen captures illustrating the *Doom II* environment.

Despite the formidable obstacles, the virtual place that was created with the *Doom II* engine captured some hitherto unreachable aspect of spatial representation. Many students who were invited to ‘play’ the level for extended periods reported fleeting moments of confusion and panic while walking the actual physical space, unsure if an experience in the virtual world could be expected in the physical world. This could perhaps be attributable to an interaction of space, function, and mnemonics.

**QUAKE I**

*Quake I*, backed by a true three-dimensional polygon-based game engine offering six degrees of visual freedom, was released in 1997 by id Software. While *Quake I* created an abundance of excitement in the gaming community for its radical new graphics engine and its polygon based characters, discernible shortcomings in its relevance to architectural depiction caused reservations about its potential usefulness. Screen resolution was poor, the palette was limited to 256 colors, a function to simulate swinging doors was glaringly absent. This diminished the possibility of gleaning any value from large scale testing. Nevertheless, two students who had had involvement in the *Doom II* project decided, in 1997, to make use of the *Quake I* engine in exploratory study of architectural design and graphics. The students produced projects that were
innovative in their representational scope, if somewhat inconclusive in architectural value.

**QUAKE II**

In 1998 id Software released a sequel to *Quake I* with *Quake II* (see Figure 7.5 below). The *Quake II* engine offered major improvements which included 3D graphic hardware support through the OpenGL API, better screen resolution, and more sophisticated actor behavior. Interest in testing the feasibility of this more mature game environment in an architecturally relevant project was abetted by access to the conceptual design process of a New York City residence. The designer of the project agreed to consult on the game environment development while concurrently creating still renderings of her design in *3D Studio 4*. Since the design had reached a developmental plateau before game modeling commenced, the work would involve only the transcribing of conceptual design data and would not be used for design development until after presentation to the client.

The transcribing of the residence into a digital format would present formidable challenges for any 3D modeling package. Within the residence was a mezzanine creating a double-height space over the main living area. The centralizing design element was an open riser stair crafted of diamond-plate steel in a housed stringer with cable railings above. Most floors were exposed concrete. There was extensive use of glass on the front façade to give views of the city and grey-smoked glass panel partitions were used throughout the interior to express ambiguity of spatial confinement and to offer views of the stair from otherwise enclosed spaces. Among other challenging elements were a fifth-floor terrace and a kitchen ceiling expressed in sheet copper and configured as triangular in plan and curved in section. Overall, the heavy use of transparency would ultimately manifest itself in the contrast between visual and physical accessibility with particular
regard to circulatory paths and rich materiality. This and the urban siting, with its attendant sights and sounds, made the project a good fit for the perceived capabilities of the *Quake II* engine.

![Figure 7.5: Screen captures illustrating the *Quake II* environment.](image)

To reinforce the idea that the building was sited in a busy urban environment, ‘objective’ sounds were integrated into the spatial experience. On the street level, the participant would hear a city soundtrack, replete with the sounds of crowd chatter, footsteps, passing automobiles, and the occasional honking horn. Upon entering the building, the sound would attenuate as the participant gained distance from the street. The street sounds could still be faintly heard if the participant moved close to the windows of the front façade, but would become less discernible as one moved to the upper floors. On the fifth floor terrace, the street sounds were barely audible, now replaced by the sound of light wind and occasional bird chirps.

The clients of the project, a real estate developer and a banker, were presented with the final results as a supplement to a traditional design proposal. The designer opened with a presentation of traditional plans, sections and elevations before moving to her rendered images, then demonstrated the game environment. Both clients demonstrated enthusiasm for the game presentation, stating that it had clarified many personally important issues, particularly in regard to material choices, spatial adjacencies,
and visual accessibility. None of these issues were obscured in the traditional presentation; on the contrary, the designer had conveyed these ideas quite lucidly.

In conclusion, the game model had acted to substantiate the same information that was given via traditional methods, which included both orthographic line drawings and detailed high-resolution renderings. It is not known how the clients would have reacted to the game presentation alone. Although the scope of these findings is limited, it is probably safe to conclude that most laypeople will find value in interactive virtual presentations because of the diminished abstraction and the experiential nature of the depiction. Feasibility, however, is a yet unresolved question which must be subject to cost/value analysis.

The documentation of the Manhattan dwelling was extremely useful with regards to the clear, accessible representation of a complex conceptual design solution. Complemented by traditional tools, the game walk-through added an experiential dimension to the presentation and gave the clients valued confidence in the elapsed process, the current results, and the potential of continued design development.

Another documented experience with the Quake II engine is offered by Richens and Trinder, who used it advantageously in the design development of the new Computer Laboratory at the University of Cambridge (Richens and Trinder 1999).

Coursework

An elective course using gaming engines was developed and first offered at the New Jersey School of Architecture in spring 2002, when the gaming engines and ancillary editing tools were judged to have reached an acceptable level of graphics capability and technical maturity. The purpose of the course was to introduce design students to the power of the game engines within an architectural context. While the structure of the course was based on the digital design studio paradigm, the additional
capabilities of the game engine were used to exploit the potential for participant immersion, spatial interaction, and multi-user design collaboration.

The graphics engine chosen for use in the course was a version of the *Quake I* engine heavily modified by Valve Software for use in the game *Half-Life*. The original engine was modified to remove irrelevant gaming-specific features. Through a series of assignments within an academic course, students in the school of architecture were asked to iteratively use and test this prototype for the collaborative exploration of a designed environment. Starting with theoretical readings in virtual reality, the students were asked to design and create spatial environments using the 3D graphical editor *QuArK* (a freeware game level editor not to be confused with *QuarkXPress*) (see Figures 7.6 and 7.7 below). Students made their environments available for others to navigate in real-time and offer comments. Design reviews were conducted in which critics were asked to enter the designed environments, explore at will and interact with the student as well as others present in the same virtual spaces (see Figure 7.8 below).

Three major projects were required to complete the course. The first, an art gallery design proposal, accustomed the students to the interface and forced them to concentrate on circulation, lighting, and custom texture libraries. In the second project, students used the scripting functions of the game application to create an event (an animation of a construction detail within a virtual space). This required careful consideration of how an event should be announced to, and triggered by, those navigating a virtual space. The final project was to express the cumulative knowledge of the semester in a programmed design challenge, the relocation and redesign of the Great Falls Cultural Center in Paterson, New Jersey.
Figure 7.6: A Screen capture illustrating the 3D world and material editors.

Figure 7.7: Screen captures illustrating advanced lighting and texture mapping.
Observations

The following is a summary of current gaming technology features, observed during research outlined by the mentioned projects. They are intended to be specific to potential usefulness in architectural applications. Listings have been broadly categorized in terms of perceived advantages and limitations, despite the understanding that a polar distinction is inappropriate in many cases.

Advantages

*Fully Interrogative.* Like the well known animated architectural walk-through, the game interface represents space from the visual vantage point of one moving through an environment on foot. Unlike most animated walk-throughs, however, the direction, the
speed, and the sequence of movement is not forcibly prescribed or scripted. This effectively eliminates the passive nature of the animated walk-through and allows participants to completely and thoroughly interrogate the rendered environment. Creating virtual spaces that are fully interrogative allows, for example, the discovery and possible subsequent enhancement of underutilized parts of buildings. Furthermore, by observing the directional tendencies of participants, as well as their propensity to linger or aggregate in particular spaces, architects can tailor designs changes responsively to the more efficient use of all spatial resources.

*Tactile Solidity.* Solids in game environments assert their solidity through collision-detection. Gaming engines allow for proper navigation through spaces bounded by solid elements such as walls as well as realistic navigation of staircases and ramps. Unlike traditional animations where the viewer can go through walls and fly through spaces, a gaming engine can be configured to disallow unrealistic navigation. Many game engines allow the spatial designer to account for friction and slippage of specific materials under foot to help reinforce the sense of an interactive presence.

*Integrated Audio.* The nature of sound and its relationship to visual space is another important part of game environments. Sounds can be triggered by specific events in the virtual game world, complementing the visual stimuli. For example, the impact sound of the footsteps of the viewer or other characters can correspond to the material that is being treaded upon, appropriate sound can issue forth from moving water or air, doors can squeak or slam, mechanical systems can hum, and crowds can roar – often with controllable levels of volume, reverberation, and attenuation. The widespread use of digital stereophonic sound, 360-degree panning, and doppler shifting in many games helps reinforce the nature of space by allowing noise sources to express their locations relative to a player.

Game environments also provide rich opportunities to test and use nonobjective sound. Nonobjective sounds may be used alone or as a supplement to reality-based or
objective sounds. They can be used as audile enhancements to spatial experiences in attempts to create and nurture specific emotional dispositions in participants. While the technique is rarely used in architecture (Helmut Jahn’s United Airlines Terminal at O’Hare Airport is a notable exception), it is used heavily in the film industry. In similar ways it can be used in game environments to underscore, for example, grand entries, spatial contrasts, anticipatory movement, etc. How successfully non-objective sound works in virtual architectural depictions is largely dependent on the skill of the spatial ‘director’ in responding to the social and physiological makeup of the audience. There are also serious ethical issues to consider as the use of nonobjective sound may be considered by some to be abjectly superfluous and emotionally underhanded.

**Multiple Simultaneous Users.** Most computer games are designed to be networkable, or multi-user. To the spatial designer, this means that designed game environments can be simultaneously experienced by multiple participants (the current limit is 64 connections on some higher-end games). This is, of course, of relevance to the collaborative component of architectural design. Colleagues, clients, and consultants can all meet together within a space, virtually, regardless of whether the individuals are in the same room or across the globe. In fact, the immersive nature of the game interface seems to enrich the collective comprehension and appreciation of a space even when participants in a critical evaluation are physically co-located, insofar as each critic becomes an active inhabitant. For more distant collaborations, most games feature an intercommunications function that takes place within the game interface using internet chat-like typed dialogue or voice channels.

As a spatial designer invites other users into a virtual world during design development, their input is likely to affect the design process more forcefully than traditional collaborations. Spaces designed for public use, for example, can be experienced by other design professionals acting as surrogate users. Opinions about perceived successes and failures in the design can be clearly communicated because the
game interface allows the articulation of the total spatial experience, rather than a perception of the cumulative effect of spatial elements. The tendency for some designers to embrace design solutions because of some personal or idiosyncratic ideal will likely be diminished within the facilitated consensual framework. In many ways this reinforces the paradigm of the studio peer critique; nonetheless the potential impact on individual creativity should be duly considered.

**Inclusion of Avatars.** A functionally and philosophically significant feature experienced while engaging in multi-user game environments is the presence of moving human-like characters. Considering that most architecture is designed for human use, representing spaces without the human presence for which they were intended may be something of a deception. In order to properly represent an architectural space meant for humans, designers must accept that humans are design entities that impact space as powerfully as walls and columns.

Understandably, humans are not walls or columns – they sit, they stand, they move from place to place, they change their clothing, they interact, and they express free will. Perhaps the very complexity of this ‘design element’ accounts for its omission from most architectural depictions.

Networked implementations allowing multi-participant interactions provide an enriched simulation of the human/spatial dynamic. Multiplayer entities in games are dynamic, purposed characters that can be very much the avatar of their embodied user. Participants can choose how their virtual character appears to other players by choosing from libraries or by creating custom models; this can help afford immediate visual recognition of participants in virtual space. To varying degrees game characters can gesture or otherwise interact with other participants.

*Simulation of Movement.* Game environments are predicated on the fact that things move. Whether it is the movement of ourselves through space or the movement of other characters, the dynamic relationship of occupants to spatial boundaries helps us to
understand the ergonomic characteristics of a space. Movement also introduces a valuable temporal quality to spatial depictions, demonstrating the time it would take to move across a single space or among multiple spaces in particular regard to obstacles to free movement, line-of-sight versus available path of movement, and separation of visual and tactile accessibility. This feature can simulate wait times for elevators, predict delivery times for goods, and test effectiveness of locations for emergency means of egress. The movement of inanimate object through space also helps to gauge the changing characteristic of a space, e.g., a closed door containing space versus an opened door admitting space.

Simulation of Site. Since game environments are interrogative, bitmapped views of the surrounding site must be dynamically adaptable to the viewer’s constantly changing position. To address this, environments modeled for games with views of the surrounding world use panoramic texture-mapping to simulate distant views. This technique is somewhat difficult to accomplish without proper photographic equipment and obviously does not allow participants to interact with the objects pictured on the panorama bitmaps.

Interaction with Objects. The competition to enrich game play has led to some innovative strides in the gaming industry with regard to player interaction with inanimate objects set in the virtual world. In early games, most inanimate objects in space did not respond to user interaction. Because most inanimate scene objects (walls, furniture, appliances) were part of a pre-computed visible set of polygons in the binary space partition data tree, performance goals dictated that they remain static. Eventually, doors and windows began to open and shut, lights responded to switches, and elevators could be operated. Each successive version of these games brings some new level of interactivity with scene objects. At this writing, there are movable objects (e.g., furnishings), breakable objects (e.g., glass panes), and operable objects (e.g., video monitors, soda machines). While there are clearly shortcomings in the physical behavior
and visual quality of affected objects, these will only improve with time. What may be potentially more useful to spatial designers is Woodbury’s idea of ‘designerly virtual reality’ in regard to inanimate objects. In such a world, even ‘non-movable’ entities would be modifiable during the spatial experience. For example, walls could be moved or reconfigured, ceiling heights could be adjusted, openings could be shifted or resized, etc.

Limitations

Comprehensiveness. Giving outsiders an interrogative ‘run of the place’ is essentially relegating control over what is viewed and how it is viewed, an important aspect of traditional architectural presentation. As such, neglect and omission become glaringly obvious. ‘Minor’ spaces and details may receive as much scrutiny as major ones. As abstraction diminishes so will the benefits of its economy. Details will invite more details.

Comprehensiveness can of course be considered an advantage. Differing approaches to design development will find differing value in a representational paradigm that demands completeness. Nevertheless, this technology will likely force a reevaluation of the acceptable balance between abstraction and reality.

Complexity. Game environment creation software currently lacks turnkey functionality and professional polish. Setup can be frustrating and lines of support are informal at best. There are still very few industry standards or conventions regarding features and user interface. Geometric data exchange between game applications and industry-standard CAD and modeling software is spotty at best. Familiarity with traditional CAD and modeling software is often more a hindrance than a help.

However, developers of high-end modeling software are beginning to take notice of the needs of gaming content creators. Polygon-friendly data exchange formats are beginning to appear in high-end modeling software that allow the exchange of geometry
with game level-editing tools. In fact, Discreet’s *3D Studio Max* is heavily used to create
game content, such as landscape elements, vehicles, and furniture. Character Studio has
become an indispensable tool for designing avatars and the parameters of their behavior.

*Geometry Limitations*. One of the most difficult aspects of working within the
current state of game modeling is geometry limitations. Objects in any possible scene
must be rendered in real time. Binary space partition data trees can be used to construct a
predetermination of the visibility and draw-order of objects from any given viewpoint,
but the dynamic depiction of the world must occur instantaneously. Most game
environments necessarily avoid excessive geometric detail opting instead for textural
detail, with mixed results.

On the positive side are the continuing advances in graphics hardware and
software Application Programming Interfaces (APIs), i.e., *OpenGL* and *DirectX*. Furthermore, manufacturers of consumer level graphics processing units (GPU) are
responding to the gaming industry’s appetite for rich, complex graphics with powerful
and sophisticated hardware solutions that optimally handle off-loaded graphics operations
that were previously accomplished through software and CPU, i.e., lighting, shading,
shadows, mapping, fog, anti-aliasing, etc. Game designers are in turn leveraging this
newfound computational potency to create more detailed and expressive virtual
environments.

*Lack of Standardization*. As with any fledgling technology, there exist a variety of
similar and competing products. As the push to create better game engines ensues, the
advances are certainly an overall benefit but the volatility of the industry is frustrating to
those using game-editing software. While there is some level of compatibility of game
geometry files among game engines, the finer points of the virtual depictions (textures,
behaviors, sounds) do not translate very well among the various game content creation
packages.
Recommendations

While many of the features of gaming technology were designed for the needs of interactive entertainment rather than for the needs of architecture, there is clearly potential for usefulness within the profession. It should be noted that many digital tools now widely used by architects were conceived for other disciplines (in fact, advertising literature for Discreet’s 3D Studio Max and Adobe Photoshop make very little mention of a usefulness to architects). Nevertheless, any tool that demonstrates utility should be examined. Wide adoption, if it occurs, will most certainly be followed by the discovery and development of a more specific tool set.

Spatial designers should explore the advances in gaming technology as a means of furthering their own goals. The tools have reached a level of maturity that allows their use in informal design development and, with the proper modifications, formal presentation. Rather than let this valuable graphic tool continue to fledge only within the gaming community, the spatial design profession should extract the architecturally relevant components of the technology to develop and nurture a specific tool set.

Architects should seek to develop new abstractions for use in the virtual world that will supplement the information availed by increasingly accurate depictions of reality. Using the experience and knowledge gained through the use of traditional conventions, architects can complement the weaknesses of the emerging game systems by judiciously borrowing from established tools.

Slavish attention to the details of reality should be pursued carefully. The cost of this in both computational resources and man-hours must be justified by project goal requirements. Rather, synergies with more traditional methods of depiction should be established and standardized.
Educators should exploit the game engines both for the analytical and representational values as well for the apparent affinity for this informational interface displayed by a new generation of students.

**Discussion**

The technological advances in computing during the last fifteen years have precipitated a fundamental reevaluation of the tools that define architecture as a profession. The maturation of graphic engines that use binary space partition tree data structures is clearly a watershed in the present digital graphics revolution. Like cinema before it, this technology will inexorably insert itself into public consciousness.

Much of the scholarly discourse on the nature of architecture and virtual reality submitted over the past three decades opines that the technology needs further development before issuance of formal conclusions. This technology threshold has been recently breached from outside of the architecture profession. The computer gaming industry has developed tools that allow architects access to mature and useful technology that applies directly to their spatial, pedagogical, and physiological sensibilities. The trajectory that these tools will take as they continue to advance into the realm of public acceptance should be directed by those who best understand the nature of space, both real and virtual.
Notes to Chapter VII

1 To explore collaboration software that functions on several hardware platform, a third distributed design review experiment was initiated using webex.com’s solution, but was quickly abandoned when the VoIP channel failed. Additionally, the host institution where the experiments were being conducted withdrew its webex.com license and thus the service was no longer readily available for further testing.

2 At the New Jersey Institute of Technology, where the author accepted a position as an assistant professor, he was introduced to the work of Professor Glenn Goldman and Michael Hoon that involved the modification of gaming engines to provide a real-time immersive environment for collaboration in architectural design (Goldman and Hoon 1994). Through casual discussions regarding the pros and cons of real-time, first person games as appropriate tools for design collaboration, the author, Professor Goldman and Mike Hoon decided to co-author a paper on the topic and submit it for publication at the CAADRIA conference – which was accepted (Hoon, Jabi, and Goldman 2003). It is important to note here that the work on gaming engines and the implementation described here should be largely credited to Mike Hoon. The author’s contribution is in the study of the technology, the framing of its advantages and disadvantages and the investigation of its relationship to and potential in synchronous distributed design reviews.
CHAPTER VIII
WEB-BASED SYSTEMS FOR COLLABORATION

General purpose tools and tools designed for a specific domain, as illustrated in the previous chapter, limit their users when adapted for different domain-specific problems. Since the architectural market, and especially the sector that would use collaboration tools, is relatively small compared to the overall market for these products, it is unrealistic to expect commercial companies to develop domain-specific adaptations of their software. A more realistic approach is either to use software that allows modification through scripting, plug-ins and other methods or to develop custom-designed solutions. Regardless of the approach, this dissertation makes a case for the need for domain-specific tools that, while re-using general collaboration techniques, provide domain-specific functionality. This chapter reports on two such custom-designed collaborative systems. By sheer coincidence, a need arose for these systems and thus they were designed by the author knowing full well that their design is useful for the dissertation research. While these systems have been applied in the general field of architecture, they are not directly concerned with architectural design. Yet, the case being made in this chapter is that regardless of the domain, certain collaborative techniques and functionalities are universal. Thus, the reader is advised to ignore the applied domain and instead compare the system architectures presented here to the one presented in association with the WebOutliner system (see Chapter VI) as well as with the requirements and theoretical framework outlined in Chapters III and IV. In fact, many of the objects and methods developed in WebOutliner are re-used for the systems described in this chapter. In addition, the two systems address issues not previously addressed by
WebOutliner yet were proposed as requirements in Chapters III and IV. For example, as discussed below, one of the systems addresses the issues of competition and evaluation as essential components of collaboration while the second system addresses the issue of hierarchy. Furthermore, the two systems discussed here have been deployed in a real-world situation with hundreds of users. The lessons learned by real-world deployment substantially add to the validity of the proposed approach.

**CAMEO-Review**

The first system is named *CAMEO*, which stands for Computer-Aided Membership and Event Organization. *CAMEO* is an object-oriented system that enables an academic/scientific organization to administer its members and events. *CAMEO* was developed as an implementation of a set of conceptual guidelines developed through a historical survey of computer-supported cooperative work (Jabi 2003). *CAMEO* is composed of three main applications: 1) a membership management system, 2) a conference registration system, and 3) a double-blind review system. All three systems extensively share reusable components that were developed using Apple’s *WebObjects* – a web-based three-tier, object-oriented system that enables the creation of interactive web-based applications (see Chapter VI).

The *CAMEO-Review* system allows authors to register for paper submission. The author(s) fill an online form that gets saved in a database and the software generates a unique ID and e-mails it to the author(s). The software instructs the authors to use that code in naming their submission. This ensures the confidentiality of their submission. While the system could in theory allow file uploads, it currently does not make use of that capability. Instead, it instructs the authors to send a *PDF* file of their submission via e-mail. The *PDF* file is then saved on a web server.
Concurrently, the *CAMEO-Review* system administrator collects the list of reviewers, enters them into the database and specifies a login and random password for each. Additionally, once the papers have been submitted, they are assigned to the different reviewers through the establishment of a relation in a relational database model. When reviewers log in, they see a list of papers that have been assigned to them and an online evaluation form for each paper that includes a numerical scoring system, private comments to the Technical Chair and public comments that are sent to the author(s). The *CAMEO-Review* software collects all information and averages the score of different reviewers. It then sorts the papers and presents all information, through a private administrative interface, to the Technical Chair for evaluation. Final notification is conducted via e-mail.

The object architecture of *CAMEO-Review* includes representations for a Paper, an Evaluation, and a Reviewer. Author information is removed before the system is opened to reviewers. Papers are assigned to multiple reviewers and reviewers can review multiple papers. This is accomplished through a many-to-many relationship (see Figure 8.1 below). The *CAMEO-Review* system, while simple in its architecture and implementation, provides some significant insights into the design of collaborative systems. Below is a summary of the most pertinent issues. Those are elaborated on at the end of the chapter.

- Ubiquity of the Web: The World-Wide Web has become the de-facto platform for collaboration. *CAMEO-Review* relies on the familiarity and ubiquity of web browsers and the standardization of HTTP and IP protocols to ensure the acceptance.
- Competition as Collaboration: The common goal of *CAMEO-Review* is achieved through a process of competition and selection.
- Social Unawareness: *CAMEO-Review* depends on social unawareness (the process of a scientific double-blind review) to accomplish its goal. Any
breakdown in this unawareness not only hinders this goal, but renders it invalid.

- **Hierarchy**: The object model of *CAMEO-Review* clearly delineates a hierarchical structure which is often missing from collaborative systems.

- **Synchronicity vs. Asynchronicity**: While *CAMEO-Review* is largely asynchronous it does deal with time as a component of the system with submission deadlines and review periods in which the system is open to reviewers for logging in and retrieving papers for review.

![Database Schema](image)

Figure 8.1: *CAMEO-Review* database schema.

**Implementation**

*CAMEO-Review* has been successfully deployed for the *ACADIA* 2001 and 2003 conferences and is being deployed again for the *ACADIA* 2004 conference. In both instances, the review system aided the submission and selection of hundreds of papers. Anecdotal evidence suggests that the system was viewed very positively by those who used it. Reviewers, in particular, enjoyed being able to download the PDF papers at their
leisure and fill out the form for each paper when convenient (see Figure 8.2 below). The system allowed them to log out and return to the system to complete their review.

Figure 8.2: Reviewers’ view.

ViSTA

The second system is named ViSTA which stands for Virtual Slide Tray Archive. ViSTA is a digital asset management and display system designed for education. A digital asset is any computer-based artifact. Examples of digital assets include images, video, audio files, word processing documents, etc. ViSTA’s implementation is currently concerned with digital images, but built-in mechanisms are in place to enable ViSTA to manage other types of assets. The ViSTA system enables instructors to search a database of digital assets, select the ones they want and save them in virtual slide trays. They can then sort the images in those virtual trays and project them using a networked computer and projector for in-class presentations as well as allow registered students to view them at will from any internet-connected computer. Students register for courses in the ViSTA system through a special code issued by the instructor. Once registered, their ViSTA homepage automatically displays the courses they are registered for and the associated
trays for that particular course. Students can also create and modify their own personal trays organized in any fashion they want. These trays can be used for reference, studying for an exam, or for presentation purposes. The ViSTA system helps administrators and faculty manage the digital collection, the courses, the trays, and the user accounts.

The ViSTA object model and database schema includes representations of: 1) Users (both faculty and students), 2) Courses, 3) Trays, and 4) Slides (see Figure 8.3 below). Users and Courses have a many-to-many relationship. That is, users can be enrolled in multiple classes and, vice versa, courses can have multiple students registered for them. Users and Courses can own multiple personal trays (one-to-many). A tray, on the other hand, can belong to a user, a course or both. ViSTA allows a tray to belong to both a user and an instructor to allow the publication and sharing of personal trays as course trays. Once an instructor deletes a tray from a course (as at the end of the semester), the tray will remain available in a personal collection of slides. Finally, a tray can have a right and a left slide and a Slide can be inserted in multiple trays (the main benefit of using digitized virtual slides as opposed to physical ones).

**Implementation**

The ViSTA system contains multiple screens. The first screen asks for a user login and password for security purposes (see Figure 8.4 below). Once logged in, the user is taken to a home page (see Figure 8.5 below). At that page, a user, if registered for a class, can see the course trays. If not, the system allows the user to register for classes as well as modify profile information at the Preferences page (see Figure 8.6 below). Even if not registered for a course, a user can search for images and create personal trays. The slide search page allows the user to search for slides using keywords and once retrieved to add them to the selected tray (see Figure 8.7 below). The task of sorting the slides can be
achieved in the Slide Sorter page (see Figure 8.8 below). At that page a set of interactive buttons allow a user to:

1. Insert new empty tray slots.
2. Specify a single or a double (left and right) format for the display of images.
3. Move, copy, paste and delete slides and tray slots.
4. Specify additional parameters such as desired image size.

Once a tray is sorted, a user would then click on the Slide Show button to view the slideshow (see Figure 8.9 below). Normal slideshow buttons are included on that page. Finally, ViSTA includes a page for sending feedback comments to the administrators as well as an online help system.

ViSTA has been tested and implemented in a real-world situation at the School of Architecture, New Jersey Institute of Technology. Specifically, it has been used for the ARCH252: History of Architecture II course with an average enrollment of 100 students. ViSTA thus far has been successfully deployed twice in the Spring semesters of 2003 and 2004.
Figure 8.3: ViSTA object model and database schema.

Figure 8.4: ViSTA login screen.
Figure 8.5: ViSTA main page.

Figure 8.6: ViSTA preferences page.
Figure 8.7: ViSTA slide search page.

Figure 8.8: ViSTA slide sorter page.
Discussion

As mentioned above, ViSTA and CAMEO-Review are two collaborative systems that use and expand components built in WebOutliner. Specifically, ViSTA’s Course, User, Tray, and Slide objects are derived directly from WebOutliner’s hierarchical node implementation. Course objects know what and how many trays they contain and they know who and how many students are enrolled in them. Trays, on the other hand, know what and how many slides are stored in them and in what location they reside. The slide sorter functionality in ViSTA is derived directly from the one implemented in WebOutliner with modification to allow left/right repositioning and remove the unneeded hierarchy of nodes and sub-nodes. CAMEO uses the same technique to assign papers to reviewers. A Reviewer object knows what papers are assigned to it. A Paper object knows what reviewers it has been assigned to. A Results object queries, aggregates and averages the scores stored in evaluation objects in very much the same manner as a WebOutliner node would query and aggregate the total areas stored in its sub-nodes.
Yet, ViSTA and CAMEO-Review address additional collaboration issues that WebOutliner did not address. The object models of CAMEO-Review and ViSTA specify a strict hierarchy that was missing in WebOutliner. Users of those systems are assigned roles and subsequently access privileges. Hierarchy has been found through real-world deployment of CAMEO-Review and ViSTA to be essential – even critical – to the their proper functioning. The case study and theoretical framework discussed in Chapters III and IV clearly indicated the need to address competition and the evaluation of proposals as essential elements of collaboration. While included in the design of WebOutliner, the actual implementation did not address the evaluation of competing proposals. CAMEO-Review presented an opportunity to study this issue. One can say that the common aim of potential contributors, reviewers, and administrators of CAMEO-Review is to collaboratively create and share a body of knowledge. Yet, systems such as CAMEO-Review use the scholarly standard of competition and anonymous evaluation to achieve that goal. Furthermore, CAMEO-Review adheres to the theoretical process diagram included in Chapter IV in that a solution consists of an assembly of sub-solutions as well as the elimination of failed proposals.

While online collaborative systems rely on social awareness to enable interaction (e.g. chat rooms), CAMEO-Review emphasizes the need for social unawareness to ensure privacy and honesty in the evaluation process.

In a similar manner, ViSTA provides for personal trays that are not shared, but that can be published as part of a shared course. A private workspace is available to users based on their login credentials. Through testing, the need for a private workspace became quickly apparent. Private workspaces were found to be needed not only for private individual work, but as essential components of the collaborative process. Specifically, in order to project slides during an exam to test the students’ knowledge, exam trays needed to be kept private till the moment they are projected and shared. Obviously, it would be impractical to assemble a tray during the exam. Assembling a
shared tray ahead of time would obviously mean the students get advance notice of the exam material. Thus, it became evident that there is a need to create trays that exist in a private workspace and that can be shared at a moment’s notice and quickly unshared and moved back to the private workspace. This corroborates the findings in the case study (see Chapter II) where the ability to move artifacts from private to public settings was found to be an essential component of a collaborative process.
CHAPTER IX
CONCLUSION

Summary of Findings

The review of the literature since the 1960s revealed that early themes set by the pioneers of the field of computer-supported collaboration have influenced the thinking of researchers to this day. The need for synchronous collaborative systems, intercommunication, a common protocol of interaction, and direct manipulation of graphics are still important issues in contemporary research and writing. However, early research tended to be utopian and take a simplistic view of design. Furthermore, it tended to overemphasize the need for synchronous support given new technological advances brought forth by faster and networked computers. The frameworks provided by pioneers such as Engelbart, Coons and Sutherland are useful yet incomplete representations of collaborative design. One of the first contributions of this dissertation was to further develop and complete the conceptual framework advanced by earlier researchers in order to inform future researchers more fully of issues that need to be addressed and provide templates for possible methods of addressing them. As result, this dissertation took an alternative approach that favors: 1) user and task analysis rather than the advancement of a normative theory, 3) an emphasis on artifacts as the enablers of collaboration, 3) small, focused, modular and object-oriented solutions rather than large and integrative systems, 4) Custom-designed, domain-specific solutions rather than the adaptation of general purpose tools, and 5) the generalization of collaboration patterns rather than the generalization of collaborative activity as a whole. This was accomplished through a
multi-faceted approach. Starting with the philosophical framework of phenomenology, a participant-observer case study was conducted that revealed the mechanisms of collaboration in the field of architecture and discovered the central role that artifacts play in them. Artifacts were found to be the main vehicles of collaboration and their intrinsic nature is a strong indication of the setting in which they were created and their purpose.

The case study facilitated the creation of a framework of representations and mechanisms that can be applied in a computer-supported collaborative system. Several aspects of the framework – consisting of participants, tasks, proposals and artifacts – were analyzed through the creation of computer-based prototypes. These prototypes allowed for a kind of insight into the needed representations, relationships and processes that would not have been possible with commercial systems. The system architecture and algorithms of commercial systems is usually a trade secret. Thus, an analysis of such a system will be limited to its user interface and operation. Such an analysis can be easily faulty. As an example, a cursory review of SYCODE and WebOutliner may lead to the conclusion that the former is a shared whiteboard while the latter is a shared outliner. While superficially this is true, a deeper investigation of their system architecture would reveal that SYCODE shared not strokes, but deeper data structures regarding rooms and spaces in a building. The SYCODE experiment proved that a common protocol of interaction, regardless of implementation, can enable collaboration. An analysis of WebOutliner’s system architecture, on the other hand, would reveal the intelligence put into each outline element and the object-to-object messaging needed to make it function as both an asynchronous and synchronous collaboration tool. The WebOutliner experiment corroborates the claim that artifacts are the main vehicles of collaboration. A synchronous collaboration session in WebOutliner is not scheduled, but is activated seamlessly when two or more participants are viewing the same artifact. Thus, artifacts become spatial locaters – such as rooms – that collect participants and allows them to interact.
ViSTA’s system architecture builds on the ideas established in the case study, the theoretical framework, SYCODE, and WebOutliner. Objects in ViSTA, as in WebOutliner are hierarchical and have built-in methods that allow them to manage their own collections, allow and disallow access, and move themselves between private and public realms. Such features were found relevant in the case study and thus it is of no disadvantage that ViSTA’s application domain is not in architectural design. The issues and mechanisms of collaboration remain constant and thus can be transferred from one domain to another.

CAMEO’s system architecture builds on a different, but complimentary set of ideas established in the first few chapters. Mainly, CAMEO-Review implements the proposal object and allows an analysis of competition as a form of collaboration. The case study clearly indicated the need for a collaborative system that includes the ability to maintain and evaluate multiple proposals that respond to a particular task. CAMEO-Review’s evaluation system adheres to the double-blind scientific standard such that issues of hierarchy, social unawareness and competition were brought forth as important requirements for collaborative systems. As in the case of ViSTA, while the application of CAMEO-Review is not in architectural design, its system architecture is based on and closely allied with the findings of the case study, the theoretical framework, and the experiments with SYCODE, WebOutliner, and ViSTA.

This dissertation makes a case for domain-specific tools that respond directly to the requirements of the application domain. While much of the logic and mechanisms of collaboration can be transferred among systems – as we have seen in WebOutliner, ViSTA and CAMEO-Review, many of the features must be tailored to the applied domain. Thus, this dissertation makes a case against general-purpose tools as insufficient for the needs of domain-specific tasks. In order to support this claim, two experiments were reported on in the area of synchronous design reviews.
The first experiment used an immersive interactive board coupled with a commercial general-purpose collaboration and instant messaging tool. The setup was used to conduct distributed design reviews. The experiment illustrated that while effective distributed design reviews are possible with the implemented setup, many problems exist. The fundamental problem is still that of lack of bandwidth. However, the experiment also indicated that a lack of domain-specific tools hindered the design review process. In many instances, design review processes and expectations needed to be adjusted to accommodate software processes that were not tailored to the task at hand (e.g. *PowerPoint* for design presentations, lack of multi-participant annotation input).

The second experiment reported on the modification of gaming engines for use as multi-participant, real-time, immersive design review systems. The experiment made a clear case for the difficulties of adapting tools that were not meant for the task at hand. Many of the limitations reported on in this experiment do not derive from a technological source, but from the fact that features and processes were not implemented since they were of no use to the original application domain. Yet, as evidenced in *WebOutliner*, *ViSTA* and *CAMEO-Review*, the underlying technology and system architecture can be re-used, adapted and extended to fit new needs. An object-oriented architecture as the one used in these systems allows for encapsulation and inheritance processes that are responsive to change and adaptation.

**Misconceptions**

The literature in computer supported collaborative design contains many misconceptions about both design and computer-supported cooperative systems. Some of these misconceptions trace back to work done in the 1960s. Unfortunately, many faulty ideas and misconceptions about the nature of the problem and the needed solution were carried forward to the present. While it is perhaps unwise to paint the fields of CSCD and
CSCW with a broad brush, it is useful to point out these common misconceptions plaguing the field and subsequently the design of corresponding computer aids. In this section, some of the most common misconceptions are explained ways of avoiding them are discussed.

**Cooperative design is one thing**

The first misconception is that cooperative design is a single activity that needs to be supported using a single system or solution. Design cannot be easily defined or reduced. It is certainly more complex, subtle, and multi-faceted than initially thought. Integrated systems that claim to support the full spectrum of collaborative design usually fail due to both their complexity as well as inability to deal with certain aspects of design (Grudin 1988). They tend to force a particular view of design that participants may or may not agree with. Once a participant feels that a system is demanding a particular method to be followed, that user will reject the system and subsequently allow it to fail.

**The more the better**

A second misconception is that the problems of cooperative design are solved by providing more information to everyone. The unfortunate side-effect of this approach has been information overload. The abundance of information has obscured relevant information rather than make it more readily accessible. As a result software developers have been introducing systems that emphasize information filtering. *WebOutliner* is a system that enables information filtering by zooming in on a sub-node and making it the root of the hierarchy. As such, a user can work on that node and its descendants regardless of other nodes in the hierarchy.
We are all team players

A third misconception is that the term cooperation is synonymous with collaboration, teamwork, and group effort. That is, those participants in a cooperative design environment are always engaged in a positive and helpful mode of co-work. However, internal design reviews in practice and academia, architect-client meetings, and architect-consultant meetings are usually a mix of collaborative and adversarial work. Systems should not only support conflict resolution, but perhaps encourage conflict to bring out the desired solution. Gaming theory has realized the potential of conflict, yet rarely is that applied in a CSCD system.

Two heads are better than one

A fourth misconception is that group work is always preferable to individual work and that synchronous collaborative work is always beneficial. If traditional modes of co-design are used as an example, the fact can be easily derived that drawing on the same piece of paper at the same time by multiple users is certainly not a useful method of co-designing. Field research indicates that group discussions work best when individuals bring significant work done previously in a private context to the meeting (Jabi, 1996a).

The open floor plan

A fifth misconception is that by bringing everyone to the same shared workspace cooperation will take place naturally. What researchers sometime forget is that team members are not all in the same hierarchical position. In almost any cooperative system, there will be a need to include concepts of privacy, access privileges, and group hierarchy.
Competition as collaboration

What is unique about systems such as CAMEO-Review is that they reverse the traditional notion of collaborative systems. One can say that the common aim of potential contributors, reviewers, and administrators is to create and share a body of knowledge. Scholarly standards specify that this goal must be achieved not through an open and inclusive process, but rather through a closed process of competition, selection, and finally confluence. While collaborative solution is sometimes thought of as the sum total of all the efforts of contributors to it, systems such as CAMEO-Review emphasize the fact that a solution usually includes a process of elimination through competition.

Social unawareness

As commonly seen, online collaborative systems rely on social awareness to enable interaction. Chat rooms (both textual and multi-dimensional) are prime examples where knowing who else is in the room is crucial to the collaborative experience. Scientific standards, however, require that the system is blind: 1) The authors do not know who else is submitting papers, 2) the authors do not know who will review their papers, 3) the reviewers do not know the identity of the authors 4) the reviewers do not know who else is reviewing the same papers they are reviewing and 5) the authors are not aware of candid comments regarding their paper that the reviewers send to the technical chair. While traditional collaborative systems rely on social awareness to enable interaction, submission and review platforms as presented in this chapter depend on social unawareness to accomplish their goal. Any breakdown in this unawareness not only hinders this goal, but renders it invalid. CAMEO-Review’s success and popularity among its users derives directly from its emphasis on strict scholarly double-blind review standards.
Hierarchy

The object model of CAMEO-Review and ViSTA clearly delineates a hierarchical structure which is often missing from collaborative systems. Both systems identify user types and present information and a level of access appropriate to that type. Both systems allow participants to peruse shared workspaces as well as create and control access to private workspaces. Such hierarchy is essential for the proper functioning of any groupware system.

Synchronicity vs. asynchronicity

The WebOutliner system provided a good example of how a web-based system can seamlessly allow the users to move between asynchronous and synchronous modes of work. Rather than manually inviting other users to join a synchronous collaborative session, WebOutliner uses an artifact-based approach that brings users together based on what artifacts they are interested in viewing. That is, users viewing the same artifact are automatically participants in a synchronous session and can immediately see who else is interested in the same artifact. CAMEO-Review does not exercise the issue of synchronous vs. asynchronous collaboration -- It is strictly asynchronous. However, CAMEO-Review does have to deal with time as a component of the system with submission deadlines and review periods in which the system is open to reviewers for logging in and retrieving papers for review. ViSTA on the other hand blurs the boundaries between synchronous and asynchronous systems. While used at the same location, ViSTA is a synchronous system – The presenter and the audience view the content at the same time and at the same location. When used outside the classroom, ViSTA behaves as an asynchronous distributed system – users can browse the content at different times from different locations.
Ubiquity of the World-Wide Web

The World-Wide Web has become the de-facto platform for collaboration. In contrast to earlier collaborative systems that created their own proprietary collaboration protocols, current systems rely on the familiarity and ubiquity of web browsers and the standardization of HTTP and IP protocols to ensure the acceptance of their software. In general, even simple and low-cost Web-based solutions have proven successful and easy to use in terms of flexible handling of the information and by this providing content for special interest groups. The world-wide web provides a consistent communication layer on top of which application servers must still be deployed to manage the collaborative process as we have seen in the case of WebOutliner, ViSTA and CAMEO-Review.

Conclusions and Recommendations

This dissertation was authored in order to ground a research agenda on collaborative design in the theoretical, empirical, and inter-disciplinary work that has been generated so far. Given this baseline of knowledge, it is hoped that researchers and designers of future systems will be able to extract useful guidelines and avoid some of the pitfalls they might otherwise face. Yet, it can also serve as the theoretical underpinning for an environment for computer-supported collaborative design. Thus, beginning with a brief discussion of a theoretical and methodological approach to the task of creating such a computer system, this section will conclude with a reference to a series of associated requirements and guidelines.

This dissertation advocates a hybrid approach to addressing the issues of computer-supported collaborative design. As indicated in the previous chapters, theories of design and group dynamics, phenomenological observations, empirical data, and knowledge generated in disciplines outside architecture can combine to fully inform efforts to construct supports for collaborative design. By adopting varied perspectives and
methodologies, it is hoped that relevant critical issues will present themselves in ways otherwise not available.

*Design is complex.* Thus, rather than engaging in the futile effort of depicting it with simple and linear processes, one should embrace its subtleties, serendipities, and complexities and use them to one’s advantage by designing for breakdown and the unexpected (Winograd and Flores 1987; Ehn 1988). Furthermore, design is collaborative. Traditional views of the architect as lone hero and primary creator of a work of architecture have been replaced, due to a paradigm shift, with a recognition of collaboration not only as a means to respond to the increasing complexity of architectural tasks, but also as a means to enhance the discipline as a whole. It, then, becomes apparent that future CAD systems must be able to address various issues of collaboration because the inclusion of various agents (client, contractor, consultant, user, ...) into the process as well as such capabilities as computerization, standardization, and integration is not simply a practical solution, but a way of assuring architectural quality.

*Design is making.* As such it requires special emphasis on the action of making as well as the thing being made, i.e. the artifact. Computerized tasks should fall from the reflective world of present-at-hand to an unreasoned and spontaneous readiness-to-hand. As a prerequisite, however, these tasks should be absorbed into the background of and become indistinguishable from the designer’s everydayness, tradition and praxis. Virtual artifacts, on the other hand, need to be brought closer to the designer’s world of concerns so that they are endowed with such attributes as objectiveness and identity not dissimilar to those found in physical artifacts. That is, there exists a need to maintain a seamless continuity in the representation and treatment of the physical and the virtual. An ability for un-impeded capture and incorporation of physical design sketches into the set of private and shared virtual artifacts, is certainly one way CAD systems could achieve the afore-mentioned goals.
Metaphors of design, and especially those of design worlds and settings, are essential for the development of shared virtual workspaces and drawing surfaces. While these would serve as the holding environments for design artifacts and histories, they can play an even more important role as centers of gathering and interaction, thus fostering communication and collaboration. For that to happen, these virtual settings should be able to convey different modes of communication, such as talking, drawing, gesturing, writing, and model building. By being linked to worldwide multi-media information networks, they could provide access to a wealth of ideas and resources that will considerably broaden the designer’s horizon.

A computer system, to ultimately benefit the design enterprise, must be deployed judiciously, put to use at the earliest possible stages of the process, and be able to follow its developments. Thus, beginning from technology support for client briefing, computer systems should be able to assist designers in creating, representing, modifying and reasoning about one of the first, but most basic documents associated with a particular project: the building program.

Given the dynamic and social nature of collaborative design, computer systems should also posses ways of defining social and hierarchical roles. Current efforts in conceptual modeling of buildings, reasoning about user intent and the use of object-oriented approaches will greatly facilitate this process.

As indicated in earlier sections, the developments of architectural ideas, which designers often view as the essence of architectural creation, significantly depend on the production of multiple, ambiguous, incomplete and abstracted graphic representations. Computer systems should take advantage of the virtual artifacts’ unique ability to dynamically represent themselves in different ways by customizing the amount and resolution of information they present to the user. Such mechanisms, however, should avoid the brittleness usually associated with current systems by allowing for a variation in the granularity of control and the implemented protocols of interaction. Britteness
should also be avoided by ensuring a degree of tolerance for incomplete or inaccurate data. Ideally, if a computer system’s performance is to degrade it should do so gracefully – that is, at a rate proportional to the amount of missing or erroneous data.

**Going Forward: Collaboratories, Virtual Design Studios and Access Grid Nodes**

The central academic setting in a department of architecture is the design studio. As opposed to a classroom, the design studio serves as a semester-long home for the student at the department. In a studio, the student is assigned to an individual workspace with a desk, some storage space, and more recently a network and power receptacle for use with his/her personal computer. Depending on the requirements of the studio instructor and the nature of the assigned project, the student can work alone or collaborate with other students. With the maturity of Computer-Aided Design software and the advent of the Internet, more and more schools of architecture are investigating the potential for collaborative digital design studios with other institutions. Several attempts have been made to varying degrees of success.

All the while, globalization trends are resulting in architectural and engineering firms that have offices and consultants around the world. The severity of communication and coordination problems has forced these companies to look for innovative solutions using computer-assisted technologies and the Internet. As a result, the building industry is demanding that graduates be not only highly skilled in Computer-Aided Design technologies, but also possess highly developed interpersonal skills with the ability to engage in computer-mediated collaborations. The idea of a Collaborative Digital Design Studio has come about in order to help solve these severe problems and respond to the marketplace’s demands.

Given this urgent need, many departments are pushing ahead with installing these digital studios without careful study of the needs and effects of this intervention. Given
limited financial and temporal resources, the digital studio sometimes constitutes little more than a room with computers in it. Fortunately, schools of architecture have available to them a good foundation of research and trials conducted in other fields of expertise such as engineering, information technology and business. In particular, several research centers have been creating interconnections using high-speed networks and collaborative software.

Research centers that use high-speed networks to collaborate with other similar centers have come to be termed collaboratories – collaborative laboratories (Olson and Finholt 1998). Rather than concentrating on one large collaborative system, collaboratories tend to use an assemblage of software that supports various activities (Agarwal, Olson and Olson, 2002).

Several collaboratories are built according to guidelines set by the Access Grid Project. According to their web site (www.accessgrid.org) the Access Grid (AG) is: “…an ensemble of resources including multimedia large-format displays, presentation and interactive environments, and interfaces to Grid middleware and to visualization environments.” AG is used for distributed meetings, lectures, and training seminars. It is not, however, well-suited for individual peer-to-peer communication. AG is now available at over 150 institutions worldwide that have installed specifically designed nodes or spaces with the appropriate audiovisual and networking equipment to provide a compelling immersive experience. Yet, AG node setups lack the direct screen interaction and manipulation that researchers in architectural education have found necessary for virtual distributed design studios.

Agarwal, Olson and Olson describe in their paper how minor inconsistencies in the setup can lead to collaboration breakdowns. Furthermore, the experiment described in Chapter VII illustrates how bandwidth problems adversely affected communication. Thus, what is needed is an environment that supports high-bandwidth and meets the social and ergonomic expectations of its users.
While the engineering and manufacturing industry has made significant advances in computer-assisted collaborative systems and environments, schools of architecture are still grappling with the technology and its possible effects on the educational environment. Furthermore, the academic setting is somewhat different from the commercial workplace and requires a re-adaptation of the technology to satisfy pedagogic instead of commercial purposes. Thus, there is still a pressing need to investigate the requirements for an effective collaborative digital studio and specify a set of design guidelines that can be replicated. Future work should investigate and specify the physical, technological, and logistical frameworks needed for an effective environment for collaboration in architecture. This can be effectively achieved through the design of a prototypical collaborative digital design environment that can be tested and offered to other institutions to learn from and adapt to their particular needs. Given their visual and spatial training, architectural designers and researchers are uniquely suited to investigate the requirements for such an environment. Further, faculty who are teaching with these new technologies and those who are familiar with design pedagogy have a unique insight into the logistical problems and opportunities offered by this new setting.
BIBLIOGRAPHY


