DISTRIBUTED COGNITION IN GAME DEVELOPMENT: INFLUENCE OF SOCIO-TECHNICAL ENVIRONMENT ON COLLABORATION AND COMMUNICATION

A thesis submitted to the Faculty of Graduate Studies and Research in Partial Fulfillment of the requirements for the degree Masters of Arts

by

Minh Quang Tran

Department of Psychology
Carleton University

August 2008

©2008 Minh Quang Tran
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.
Abstract

This study examined collaboration and communication among a group of serious game developers. The goal was to understand how collaboration in multi-disciplinary teams is mediated by technology and the environment. We performed a cognitive ethnography based on the Distributed Cognition framework. Our findings are presented using the Distributed Cognition of Teamwork (DiCoT) analysis method, content analysis and as a model based on Grounded Theory analysis. This report includes work models and design rationale for improving the physical layout of the workspace, information flow and shared artefacts. We also present graphs describing the patterns and content of face-to-face communication within this team. Lastly, we developed a model of collaboration that includes social and technical factors affecting the collaborative process.
Acknowledgements

My supervisor, Dr. Biddle, for his guidance, insight, and generosity. My committee members, Dr. Brian Greenspan, Dr. Andrew Patrick, Dr. Janet Mantler, Dr. Tzvetanka Dobrev-Martinova and Dr. Bernadette Campbell, for their questions and criticism. All my professors, but especially Dr. Avi Parush, Dr. Gitte Lindgaard, Dr. Tansley and Dr. Thorngate, for their knowledge and advice. Thank you.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Serious Game Development</td>
<td>2</td>
</tr>
<tr>
<td>Successful Serious Games</td>
<td>3</td>
</tr>
<tr>
<td>Motivation for Research</td>
<td>5</td>
</tr>
<tr>
<td>Research Context</td>
<td>6</td>
</tr>
<tr>
<td>Software Development Process</td>
<td>7</td>
</tr>
<tr>
<td>Software Design Process</td>
<td>9</td>
</tr>
<tr>
<td>Multi-disciplinary Game Development</td>
<td>11</td>
</tr>
<tr>
<td>Computer Supported Team Work</td>
<td>12</td>
</tr>
<tr>
<td>Collaboration, Creativity and Teams</td>
<td>15</td>
</tr>
<tr>
<td>Teams as Information Processing Units</td>
<td>18</td>
</tr>
<tr>
<td>Conceptual Framework</td>
<td>20</td>
</tr>
<tr>
<td>Distributed Cognition</td>
<td>20</td>
</tr>
<tr>
<td>Research Questions</td>
<td>25</td>
</tr>
<tr>
<td>Methodology</td>
<td>26</td>
</tr>
<tr>
<td>Cognitive Ethnography</td>
<td>26</td>
</tr>
<tr>
<td>Data Collection Procedure</td>
<td>29</td>
</tr>
<tr>
<td>Data Analysis Procedure</td>
<td>32</td>
</tr>
<tr>
<td>Results</td>
<td>34</td>
</tr>
<tr>
<td>Distributed Cognition of Teamwork (DiCoT) Analysis</td>
<td>38</td>
</tr>
<tr>
<td>Physical Model</td>
<td>39</td>
</tr>
<tr>
<td>Information Flow Model</td>
<td>45</td>
</tr>
<tr>
<td>Artefact Model</td>
<td>63</td>
</tr>
<tr>
<td>Summary of Design Issues and Solutions</td>
<td>73</td>
</tr>
<tr>
<td>Recommendations and Design Alternatives</td>
<td>79</td>
</tr>
<tr>
<td>Content Analysis</td>
<td>89</td>
</tr>
<tr>
<td>Content Analysis Results</td>
<td>92</td>
</tr>
<tr>
<td>Content Analysis Summary</td>
<td>97</td>
</tr>
<tr>
<td>Grounded Theory</td>
<td>98</td>
</tr>
<tr>
<td>Model of Collaboration</td>
<td>99</td>
</tr>
<tr>
<td>Contributing Factors to Collaboration</td>
<td>100</td>
</tr>
<tr>
<td>Act of Collaborating</td>
<td>104</td>
</tr>
<tr>
<td>Creative Process</td>
<td>111</td>
</tr>
<tr>
<td>Grounded Theory Discussion</td>
<td>114</td>
</tr>
<tr>
<td>Main Findings and Discussion</td>
<td>119</td>
</tr>
<tr>
<td>Factors Mediating the Collaborative Process</td>
<td>119</td>
</tr>
<tr>
<td>Interdependency of Face-to-face and Electronic Communication</td>
<td>121</td>
</tr>
<tr>
<td>Lack of Technical Support for Team Situational Awareness</td>
<td>123</td>
</tr>
<tr>
<td>Recommendations for Building Collaborative Teams</td>
<td>124</td>
</tr>
<tr>
<td>Strengths and Limitations of Methodology</td>
<td>125</td>
</tr>
<tr>
<td>Conclusion</td>
<td>128</td>
</tr>
<tr>
<td>References</td>
<td>131</td>
</tr>
</tbody>
</table>
List of Tables

Table 1. Cooperative work technologies (Baecker, Grudin, Buxton & Greenberg, 1995) 14
Table 2. Comparison of individually and socially Distributed Cognition (Perry, 1998) ... 24
Table 3. Principles of traditional ethnography (Ball & Ormerod, 2000) ......................... 27
Table 4. Field study timeline and events ................................................................. 35
Table 5. Studio room description ........................................................................ 41
Table 6. Roles and their corresponding team members ........................................... 47
Table 7. Information Flow 2 description ................................................................. 48
Table 8. Information Flow 2: I - Finder to Reporter ............................................... 49
Table 9. Information Flow 2: II – Reporter to Assessor .......................................... 51
Table 10. Information Flow 2: III – Assessor to Fixer ............................................ 53
Table 11. Information Flow 2: IV – Fixer to Resource ............................................ 54
Table 12. Information Flow 2: V – Fixer to Assessor ............................................... 55
Table 13. Information Flow 2: VI – Assessor to Output ........................................ 56
Table 14. Information Flow 3 description ............................................................... 57
Table 15. Information Flow 3: I – Information is created ..................................... 58
Table 16. Information Flow 3: II – Information is filtered ..................................... 59
Table 17. Information Flow 3: III – Information is updated ................................. 61
Table 18. Information Flow 3: IV – Information is archived ............................... 62
Table 19. Relation of Information Flow 2 to status changes in Mantis Bug Tracker ...... 64
Table 20. Mantis Bug Tracker dashboard view description .................................... 66
Table 21. Mantis Bug Tracker profile view description .......................................... 68
Table 22. Description of whiteboard placement .................................................... 71
Table 23. Claims analysis of physical layout recommendations ........................... 81
Table 24. Claims analysis of recommended information flow .............................. 83
Table 25. Claims analysis of recommended artefact designs ............................... 85
Table 26. List of generated codes for content analysis .......................................... 91
Table 27. Recommendations for Collaborative Teams ......................................... 124
List of Figures

Figure 1. 'Darfur is Dying' game screenshots (Ruiz, York, Stein, Keating and Santiago, 2007) .................................................................................................................................3
Figure 2. Waterfall process model of software development life cycle (Royce, 1970) ..............8
Figure 3. Cognitive engineering model of software development (Yilmaz, 2006) ......................18
Figure 4. Distributed Cognition model (Preece, Rogers, Sharp, 2002) .................................22
Figure 5. Field study objectives and timeline chart ...............................................................32
Figure 6. Teams involved in game development process .......................................................37
Figure 7. Social and technical components involved in the development process ................38
Figure 8. Physical layout of Studio room .............................................................................40
Figure 9. Photographs of the Studio Room ...........................................................................40
Figure 10. Information Flow Model 1 ..................................................................................46
Figure 11. Information Flow Model 2 ..................................................................................47
Figure 12. Information Flow Model 3 ..................................................................................57
Figure 13. Mantis Bug Tracker screenshot 1 .........................................................................65
Figure 14. Mantis Bug Tracker screenshot 2 .........................................................................65
Figure 15. Mantis Bug Tracker screenshot 3 .........................................................................66
Figure 16. Whiteboards in Studio room ..............................................................................71
Figure 17. Current arrangement of agents in Studio team ....................................................80
Figure 18. Recommended arrangement of agents in Studio team .........................................81
Figure 19. Recommended Information Flow Model ..............................................................83
Figure 20. Recommended application to increase situational awareness .............................86
Figure 21. Percentage of communication with only Studio members (Internal) versus Studio members with other groups (External) ............................................................93
Figure 22. Percentage of communication relating to work versus non-work topics ..............93
Figure 23. Percentage of communication relating scheduling to work (Plans), game issues, game development tools (Tools) and scheduling tools (Meta-tools) .........................94
Figure 24. Percentage of communication relating to scheduling work (Plans), game issues, game development tools (Tools) and scheduling tools (Meta-tools) .........................94
Figure 25. Studio team social network .................................................................................95
Figure 26. Involvement of each Studio member .................................................................96
Figure 27. Humor in conversations – Day 8 .........................................................................97
Figure 28. Theoretical model for collaboration in game development teams .......................99
Figure 29. Game Abstraction Hierarchy ..............................................................................109
List of Appendices

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A</td>
<td>Ethics Package</td>
<td>139</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Examples of DiCoT models</td>
<td>147</td>
</tr>
<tr>
<td>Appendix C</td>
<td>Physical Layout Sketches</td>
<td>151</td>
</tr>
<tr>
<td>Appendix D</td>
<td>Grounded Theory coding examples on paper (hand-written)</td>
<td>153</td>
</tr>
<tr>
<td>Appendix E</td>
<td>Grounded Theory thematic scheme on wall</td>
<td>156</td>
</tr>
<tr>
<td>Appendix F</td>
<td>Grounded Theory text corpus sample</td>
<td>157</td>
</tr>
<tr>
<td>Appendix G</td>
<td>Official Job Descriptions</td>
<td>161</td>
</tr>
<tr>
<td>Appendix H</td>
<td>Social network graph and matrix</td>
<td>166</td>
</tr>
<tr>
<td>Appendix I</td>
<td>Humor and conversation charts</td>
<td>167</td>
</tr>
</tbody>
</table>
Introduction

The purpose of this study is to understand how social and technical factors in a software game development environment influence collaboration and communication. The development environment is a socio-technical system (Whitworth, 2006). It includes individuals, artefacts and culture. Collaboration and communication are processes that occur within the environment. By understanding how it affects these processes, we can begin to design better environments for game developers. Better development environments are believed to lead to better game products. Good development environments are those that support cognition at the individual and team level. Thus, the goal of this study is to understand how individual and team cognition is supported by the environment.

The theoretical framework we use is Distributed Cognition (Hutchins, 1995). Distributed Cognition emphasizes cognition in socio-technical systems. This framework will allow us to examine the human and technical components of the development environment by centering the study around a unit of analysis common in both. This unit of analysis is information or more specifically, information use. We look at social interactions and artefact-mediated activities that occur during game development. Two immediate aspects of the environment to be looked at are the multi-disciplinary nature of the team and artefact-mediated communication. Effective collaboration in multi-disciplinary teams requires a common understanding of the goals and shared mental model of the problem space (King & Majchrzak, 1996). This common understanding is
much harder to achieve in diverse teams. Technology plays a large role by allowing new forums and opportunities for communication to build the understanding of the product and each other.

**Serious Game Development**

There has been interest in computer games for a long time because of their ability to capture and engage players (Loftus & Loftus, 1993), but more recent studies of games show much greater potential. Software games made purely for entertainment are argued to be rich learning environments (Gee, 2004) and social spaces (Yee, 2007; Steinkuehler & Williams, 2006). The potential value of games beyond entertainment has lead to the development of *serious* games. Serious games keep the essential elements of games, but their primary purpose is not entertainment. These games incorporate educational elements to teach concepts and train skills that will be useful outside of the gaming context. Other names for serious games include educational games (edutainment) or persuasive games. However, attempts to mix learning and fun have only been moderately successful (Neal, Perez & Miller, 2005). The difficulty with integrating pedagogical goals into software gaming products is partly due to lack of understanding about how each domain complements each other (Dickey, 2006). The different domain expertise required to develop serious games include software engineering, instructional design, game design, interaction design, writing, and subject matter experts. How well the knowledge and skills from these disciplines can be integrated will ultimately determine the success of serious game products.
Successful Serious Games

In this section we provide an example of a serious game and explain some principles of serious game design. Successful serious games seamlessly integrate fun and learning. An example is the game titled "Darfur is Dying" (Figure 1). The game was created to raise awareness about the struggle of refugees in Darfur. It was chosen as the winner of a national serious game design competition and has received many endorsements (Darfur Is Dying, 2008). The game is available online and has been played by over 800,000 people. The game concept was created by students at the University of Southern California in collaboration with humanitarian workers from the Darfur region and then later developed with an industry partner. The gameplay is a mix of action and strategy, but set in a real context. The game requires players survive the Darfur region by avoiding all of the dangers. To succeed, the player must learn about managing resources in a refugee camp and escaping militia. The real world objective of the game was to raise awareness about issues in Darfur.

Figure 1. 'Darfur is Dying' game screenshots (Ruiz, York, Stein, Keating and Santiago, 2007).
All serious games are built upon basic principles of game design. Adams and Rollings (2007) formally define a game as “a [participatory] activity, conducted in the context of a pretended reality, in which participants try to achieve at least one arbitrary, nontrivial goal by acting in accordance with rules” (p. 5). Within this definition are four elements of all games: interactivity, pretending, goals, and rules. Serious games should be designed with careful attention to these aspects. The rules set the parameters for possible actions by the players. The goal is the objective the player tries to achieve in the game. Pretending gives significance to the actions in the context of the game world. Lastly, games must be interactive.

According to Crawford (2003), interactivity is the defining characteristic of software games. He argues software game designers should focus on this aspect the most. It is what Bates (2004) refers to as the “moment-to-moment [player] experience”. Interactivity is also a key focus of usability research (Norman, 1988). The interactivity is a major determinant of user satisfaction and learning; the player's should always have a “constant stream of interesting choices that have significant outcomes” (Bates, 2004, p. 20). This is how games engage players; it urges them to act. Crawford (2003) equates computer interactivity to a “good conversational partner” (p. 77). The computer listens well, accepting and acknowledging all player input. It thinks well, creating interesting scenarios and reacting appropriately to the player's inputs. Also, it speaks well, giving the player timely feedback that is easily understood. Thus, the game experience is like being engaged in a good conversation where you forget about everything else and the "real
world disappears” (Bates, 2004, p. 21). It is this level of engagement that educators hope to capture from game design and translate into an educational experience (Dickey, 2006).

Learning relevant skills and knowledge is the second goal of serious game design. Players are supposed to acquire skills and knowledge that are relevant in real-world contexts. One way is to integrate work and play such that they are the same activity, such as scoring points for successfully managing water supply in a camp. Another way is to take advantage of the state players are in when playing and present relevant knowledge alongside the game experience. There are believed to be benefits to knowledge acquisition and retention when in a state of contentment. However, the evidence that serious games are significantly better than other learning methods is not strong (Wong, Shen, Nocera, Carriazo, Tang, Bugga, Narayanan, Wang & Ritterfeld, 2007). This may be due to lack of knowledge about how to best integrate game design with pedagogical methods. We hope to learn more about how they are currently being integrated by studying the serious game development process as it occurs in a real setting.

**Motivation for Research**

Serious games are a promising application of computer technology. In this study, we are interested in the development process that leads to the creation of successful serious games. The analysis will focus on software development in its natural context. We believe the quality of serious games can be improved through better understanding of the problems that arise during the design and development phase. Gulliksen (2006) agrees that developer issues have been largely neglected by computer science researchers.
The serious games development environment is viewed as a complex socio-technical system (Whitworth, 2006). The people who design and develop the games are important, but so is the environment within which they work. The socio-technical perspective emphasizes the importance of contextual factors that affect the design and development process. The socio-technical perspective includes people (individual skill, cognition, behaviour), technology (cognitive artefacts, work processes, organizational structures) and social relations (communication, collaboration). It stresses the “reciprocal interrelationship between humans and machines” and posits that both the “technical and social conditions of work [be shaped] in such a way that efficiency and humanity would not contradict each other” (Rodolph, 1999, p. 59). This is consistent with the goal of human-computer interaction research, to enhance the performance of human activity through designing technical systems that accommodate the strengths and weaknesses of people and computer systems (Wickens, Lee, Liu & Becker, 2004).

Research Context

The research context for this study is a multi-disciplinary software development environment. In this section we explain some of the trends and ideas that will frame the study of the development environment. Two major trends in software development have been the increased diversity of expertise in software development teams and increased complexity of the software products. In the early days of computing, software and hardware were very closely coupled. Writing software required highly detailed knowledge of the hardware domain. This meant that software development teams
consisted mainly (if not exclusively) of software and computer engineers (Barthelmes & Anderson, 2000). Eventually, uses for software spread to outside of the engineering domain. More people, whose main expertise was not engineering, became interested in having software applications for their work. At the same time, hardware and software development tools became more powerful and easy to use. This allowed the creation of larger and more complex software products. Large and complex software products necessarily required larger development teams. Larger development teams required management and control. Producing software products became as much about managing work processes and social structures as it was about producing computer code.

Software Development Process

As software development teams became larger and more diverse, problems of communication and collaboration became prevalent. This made large-scale software development more "chaotic" (Barthelmes & Anderson, 2000, p. 8) and resulted in difficulty with predicting completion time and budgeting costs (Kremerer, 1987). Thus, engineers and managers saw the need to pay closer attention to the development process. The result was prescriptive models for software development called software development life cycle models. Following these models would help ensure successful development of software that met the user requirements and standards. Life cycle models serve as a basis for "planning, organizing, staffing, coordinating, budgeting, and directing software development activities" (Scacchi, 2001, p. 1). One life cycle model that was
widely used and is still the basis for many life cycle models today is the waterfall model, shown in Figure 2.

Figure 2. Waterfall process model of software development life cycle (Royce, 1970).

The waterfall model is based on stepwise refinement starting from system requirements and ending with an operational software product. It is a process of "progressive refinement and enhancement of high-level system specifications into source code components" (Scacchi, 2001, p. 6). This model of development requires highly centralized planning, control and coordination. A problem with this model was the long feedback cycle from software requirements to testing. The requirements of large software products change continuously (Lehman, 1985), sometimes outpacing the speed of design and coding. Although this model does include iterations and incremental builds as a key part of the development, the iteration cycles were not fast enough. To keep up with the pace of changing requirements, testable code had to be generated faster. Thus, the incremental builds needed to be more frequent. The long and elaborate design phase typical of the traditional waterfall model became less practical.
The traditional waterfall fell out of favour for more agile development process that have shorter feedback cycles between requirements, design, coding and testing. Another factor that forced changes to the traditional waterfall model was the need for user involvement. User involvement in the development process led to better products for the end-users and is more cost effective, particularly in larger projects (Maintei & Teory, 1988). Following the latest trends, the serious games development process is expected to have short iteration cycles and high user involvement.

*Software Design Process*

Almost all aspects of software development involve some form design work. This has led some researchers to characterize the entire development process as a design process (Ye, 2006; Barthelmes & Anderson, 2000). Ye (2006) defines design as “an integrated response to a whole series of often conflicting issues through integrating and combining a limited set of basic building blocks” (p. 16). The definition given by Carstensen and Schmidt (1998) is design “consists in exploring and identifying the interactions between conflicting requirements so as to be able to decide on an acceptable compromise” (p. 2). Generally, it is agreed that a design process involves a series of decisions made to achieve an overall objective. However, none of the decisions can be planned completely ahead a time and the final design outcome cannot be specified exactly until it is finished. The corollary of development being a design process is that it is difficult to predict and control. This is even truer when dealing with creative development teams, where software design involves “complex human processes of
cognition, communication and negotiation (Floyd, 1991, p. 24, emphasis added).

Design is a highly contextualized and opportunistic activity that depends on the resources in the environment (Suchman, 1987; Guindon, 1990). Designing requires making choices based on the current situation and resources. The design product is “dynamically determined by the knowledge that the designer has and the back-talk of the information presented in the workspace” (Ye, 2006, p. 16). The information presented in the workspace is critically important to the final state of the product just as much as the skill of the individual or design team. Information is present in the form of representations encoded on artefacts and can also be gained by communicating with other people.

Design artefacts are created throughout the design process. Design artefacts “embody the knowledge about the product being designed” (Keil-Slawik, 1991, p. 168). They are the interim designs, prototypes, and models of the final product. A unique characteristic of software development is the design artefact and the product are both essentially the same material. They are both “symbolic representations” (Simon, 1996). The material structure of the design artefact and product is irrelevant. This means designers have much more freedom to use different types of physical artefacts as design artefacts. Furthermore, our brains are also “machines” that represent, manipulate and store symbols (Simon, 1996). Therefore, they also hold design artefacts. Within the development environment, design artefacts are the thoughts inside people’s heads, the code inside the computer program, the hand-writing on paper and the scribbles on the
whiteboards. All of these, and many other, artefacts are expected to be used by the developers to guide their decisions as the product evolves. Thus, given the multitude of ways information (as design artefacts) are created, used and stored, we will have to examine a wide variety of tools and keep an open eye for what relevant factors are constraining or contributing to the creative work.

*Multi-disciplinary Game Development*

Serious games development has additional challenges because of its multi-disciplinary nature. Development of serious games requires integrating concepts and work practices of different professional domains. A similar issue was examined by Chamberlain, Sharp and Maiden (2006) in their study of agile development teams integrating user centered design (UCD) practices. They observed how professionals reconciled difference between UCD and agile development. The main differences between UCD and agile development are the documentation requirements and deliverable timelines. UCD requires documentation because it values empirical and technical reporting. In contrast, agile development prefers not to rely too much on documentation because it slows down the development process. UCD requires more time doing up-front design and planning to understanding user needs. Agile developers prefer to spend their time getting the program code right. Although these differences are not contradictory, how they complement each other is not always clear and can result in conflict when not integrated properly. In the Chamberlain, Sharp and Maiden (2006) study, integrating UCD and agile resulted in logistical problems and strained relationships. In the end, their
conclusions is successful integration of these two traditions requires commitment from professions of both disciplines to be willing to contribute their expertise, understand each others needs and resolve their "power struggles" (Chamberlain, Sharp & Maiden, 2006, p. 152). We expect professionals in other multi-disciplinary teams, such as game development, to have to deal with similar problems.

In serious games development, there are several different professional domains that must work together. Typically, there are programmers, graphic artists, writers, subject matter experts, game designers and instructional designers that collaborate to create a serious game. Some teams may also have usability experts, user interface designers and system engineers. These different professional domains come from different backgrounds and have different expectations about work. They also produce different components in a game. These components can be viewed as the different scripts that go into a game, such as story elements, game play rules and game code (Aldred, Biddle, Eaket, Greenspan, Mastey, Tran & Whitson, 2007). Successful integration of these scripts requires successful collaboration between the different professions. We examine how collaboration occurs and what of the environment contribute to a positive outcome.

Computer Supported Team Work

Software development is typically a team activity requiring the coordination and collaboration of many different people. The essential elements of teams are perception of a group, interdependence, role differentiation and shared purpose (Wickens, Lee, Liu & Becker, 2004). However, apart from those common characteristics, teams vary greatly in
structure, relation of members and output. Teams can be classified by various factors. Sundstrom, De Meuse, and Futrell (1990) classify teams by their role differentiation and external synchronization. Role differentiation is the extent to which the work and contribution of each individual in the team differs. External synchronization refers to level of control external entities have on the work and productivity of the whole team. It can also refer to the amount of external feedback that the team receives, which allow them to know if they are doing their jobs correctly.

Different types of teams have different needs. For example, teams with high differentiation benefit more from cross-training. Cross-training gives individuals a chance to learn about the other jobs. This increases the knowledge of each member and creates a shared mental model for the team (Volpe, 1993). A more recent study by LaToza, Venolia and DeLine (2006) also found benefits of having a shared mental model, but show that shared mental models can be created with less formal methods than training. The less formal methods are mainly information repositories, such as whiteboards and documentation. Those are also valuable for learning about each other’s work and keeping up-to-date on the team’s progress. Computer technology is also a valuable tool for cross-training and communication.

Computer supported cooperative work (CSCW) is a field of study concerned with the use and design of computer systems to support collaboration and coordination between individuals and groups (Carstensen & Schmidt, 1999). The research has been fruitful in providing models of cooperative work (Andreissen, 2002), research
methodologies for studying cooperative work (Dourish & Button, 1998) and design rationale for groupware (Grudin, 1988). A basic understanding of CSCW will help frame the observations and analysis of this study.

Baecker, Grudin, Buxton and Greenberg (1995) present a way to classify CSCW. They note that cooperative work can be organized by time and place (Table 1). Different types of collaborative activity are satisfied by different types of technology. Many of these technologies are common now in game development environments.

Table 1.

Cooperative work technologies (Baecker, Grudin, Buxton & Greenberg, 1995)

<table>
<thead>
<tr>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same</td>
<td>Public computer displays</td>
</tr>
<tr>
<td></td>
<td>Shared desktop</td>
</tr>
<tr>
<td></td>
<td>Wall displays</td>
</tr>
<tr>
<td></td>
<td>Instant Messaging</td>
</tr>
<tr>
<td>Different</td>
<td>“War” rooms</td>
</tr>
<tr>
<td></td>
<td>Email</td>
</tr>
<tr>
<td></td>
<td>Group displays</td>
</tr>
<tr>
<td></td>
<td>Bulletin boards</td>
</tr>
</tbody>
</table>

An advantage of computer supported work is that they do not have to be confined by location or time (Ishii and Miyah, 1991). CSCW addresses collaborative needs by externalizing knowledge, facilitating communication and creating a shared virtual workspace. Thus, the CSCW technology plays an important role in the development process.
Collaboration, Creativity and Teams

Two relevant concepts from organizational psychology are teams and teamwork. Muchinsky (2003) defines a team as "a social aggregation in which a limited number of individuals interact on a regular basis to accomplish a set of shared objectives for which they have mutual responsibility". Collaboration is the term we use to identify the process of interaction to achieve the objective and is used interchangeably with teamwork. However, our study will not look strictly at teamwork. Thus, factors such as team leadership and team structure are important, but not central to the study. Collaboration is considered to be a more broad and inclusive term to take into account interaction between people who are not part of the same team or group. Also, we will look at collaboration that is mediated by computers, where technological factors may be more prevalent than team factors.

Many factors lead to successful collaboration. Martin-Rodriguez, Beaulieu, D'Amour & Ferrada-Videla (2005) found a broad set in their review of the literature on collaboration. They categorized the factors into three levels: interpersonal, organizational and system level. They refer to system as environmental factors beyond the control of any one organization. At the interpersonal level, friendship, trust and willingness to collaborate are key factors. At the organization level, organizational culture, resources and structure are important. At the system level, professional norms, power and education are identified as the factors. Furthermore, two case studies by Rossen, Bartlett & Herrick (2008) emphasis the need for good communication to exist between collaborating parties.
Consistent with these findings, Pietroburgo & Bush (2008) have their own set of five elements required for successful collaboration; they are management and organizational support, positive relationships and history between parties, empathy and understanding of the other party, a common purpose and lastly, they identify respect as an essential element.

Computer-mediated collaboration (CMC) presents its own additional challenges. One of these being computer use itself. There is the hurdle of usefulness and self-efficacy (Padilla-Melendez, Garrido-Moreno & Auila-Obra, 2008). The collaborators have to know how to use the system and see the use of it as important to getting the work done. These factors influence their attitude and intention to use the system for collaboration. Providing collaborative software does not guarantee greater collaboration either. Cho and Lee (2008) found that computer-mediated collaboration did not increase collaboration significantly. The flow of information and number of collaborating partners were limited to what was already happening in face-to-face situations. The lesson from this is the issue of collaboration needs to be examined from a broader context. Introducing CMC software cannot solve problems of collaboration that are interpersonal or cultural. However, advantages of successful CMC is that it provides another method of interacting that can increase social engagement within the group (Shen, Hiltz & Bieber, 2007). A CMC environment can also provide real-time feedback to collaborators about who is contributing, which helps in collaborative problem solving (Jermann and Dillenbourg, 2008).
Collaboration is also considered an essential element for fostering creativity within organizations (Amabile & Gryskiewicz, 1987). We expect the serious game development process to include a high level of collaboration resulting in a creative output. Ideally, the game product will implement novel ideas and methods for gameplay and teaching. In addition to having a collaborative work process, other factors that encourage creativity in organizations are goal clarity, trust, constructive tension, challenge, freedom, and management support (Nemiro, 2007; Andriopoulos, 2001). Andriopoulos (2001) would add to the list some personal factors such as, a participative leadership style, and knowledgeable and motivated employees. In describing the ideal worker for creative teams, he says they should be “people with broader interests, who are eager to learn and prepared to take some risks”.

One more factor relating to CMC and teams is the use of multiple types of media to communicate. Lipnack and Stamps (2000) coined the term “boundary-crossing interactions” to refer to interactions through a computer mediated space. They believe connections made through electronic media, as opposed to face-to-face, builds a different type of trust among team members. This type of trust is unique because it is not based on authority (Lipnack & Stamps, 2000) or social identity (Tanis & Postmes, 2008). Although boundary-crossing interactions was coined for virtual teams who have no face-to-face interactions, additional bonds from the use of CMC may develop in co-located teams working in virtual environments.
Teams as Information Processing Units

Yilmaz (2006) proposed a model of the work process in terms of cognitive engineering. She proposes a computational model that has the three stages of information processing: input, processing, and output (Figure 3). Armour (2003) agrees with this conceptualization, noting that software development work is a “knowledge acquisition activity that involves the transformation of user needs into software products that realizes the requirements elicited from these needs”.

![Figure 3. Cognitive engineering model of software development (Yilmaz, 2006)](image)

Yilmaz (2006) also makes other notable observations about development work. She views the development process as “human-centered” and stresses a model of the software development process has to include a large human component. This is made up of motives, individual skill, individual knowledge and social arrangements. “What people actually do, how they communicate and collaborate, how they solve problems, resolve conflicts, and learn behaviours matters in the outcome” (Yilmaz, 2006). Another observation they make about the software development processes are they are managed
and coordinated to achieve an increasing or stable level of efficiency. Thus, the software
development process tends to improve over time. Software development processes are
also goal-directed and adaptive. The goal of the software development process is to make
some type of software product. To reach the goal, the system is required to adapt to
several types of changes. The changes can be employee turnover, requirements changes
or changes in work procedure. The development process is characterized as an
information processing system, but the processes is performed and maintained by people
within the system. In other words, it cannot be considered as merely an automatic or
mechanical process.

In our research context, we identify the serious game development process as a
special type of software development process. We expect the process to share similarities
of software development, particularly in having the same phases of the development life
cycle. However, the phases should be shorter resembling agile and iterative methods. We
also identify the team as a multi-disciplinary team working in a technical environment,
and engaged in a design activity. Thus, we expect the team to exhibit many of the factors
common to successful collaborative and creative teams. Finally, we approach the study
from a cognitive frame of reference with a focus on information use and specifically, the
input, process and output stages. A theoretical framework appropriate for this study is
Distributed Cognition (Hutchins, 1995; Perry, 2003). We discuss this framework in the
next section.
The theoretical framework for this study is Distributed Cognition (Hutchins, 1995). It borrows ideas and vocabulary from research on cognition. However, it differs from the traditional viewpoints by stating cognitive processes are present in the interactions of people and artefacts, rather than merely being something that happens inside individuals as a brain activity. Thus, the framework fits well within socio-technical systems thinking, which emphasizes the interaction between social and technological factors. Distributed Cognition claims informational representations and transformations that occur in socio-technical systems are functionally equivalent to those representations and transformation that occur in the mind (Hutchins, 1995; Perry, 2003). The software development environment is viewed as a “functional system consisting of people as well as cultural artefacts, designed and evolved to accomplish [goals]” (Artman & Waern, 1999). Using a cognitive lens, we focus on the information processing characteristics of the system and the “cognitive phenomena” as they occur between individuals and artefacts (Sharp, Robinson, Segal & Furniss, 2006).

The theory of Distributed Cognition extends the boundaries of cognitive processes outward to include socio-technical systems (Hollan, Hutchins, Kirsh, 2000; Perry, 2003). The theory states that cognitive processes occur when individuals interact with other people, resources and things in their environment. More importantly, the cognitive processes actually occur outside the head. Hutchins refers to this as cognition in the wild.
(Hutchins, 1995). Neisser defined cognition as "all of the processes by which the sensory input is transformed, reduced, stored, recovered, and used" (as cited in Perry, 1999, p. 88). Perry (1999) also adds it is the "abstract machinery of problem solving and the organization of knowledge about the problem domain" (p. 88). These definitions of cognition are based on the function of cognition. They suggest it is the mechanical properties of cognition that are important and not the material properties. Typically, the study of cognition is assumed to be the study human cognition or cognition in other animals. However, this does not need to be the case. We can find cognition in many more places when we make no assumptions about what can participate in the cognitive process (Hutchins, 1995). The criteria for cognition should be the "functional relationships of elements that participate together in the process" (Hollan, et al., p. 175). Four elements that are common to all cognitive systems are: sensors that take input into the system, action generators that produce the output, information processing units that transforms representations to convert input to output, and memory to store representations. Figure 4 shows how the elements are present in physical environments.
Figure 4. Distributed Cognition model (Preece, Rogers, Sharp, 2002).

In socio-technical systems where there are people communicating and working with external artefacts, we can find Distributed Cognition. The entire system is a cognitive entity because it is involved in "creation, transformation and propagation of representational states" (Hutchins, 1995, p. 49). When teams coordinate actions to get work done, information is spread and modified throughout the team and across external artefacts. The simplest cognitive processes that socio-technical systems are capable of are representation and transfer of information, but can be more complex to include computation and problem solving.

Within socio-technical systems, we differentiate between two types of Distributed Cognition. Distributed Cognition can occur at the individual level and at the social level. Individually Distributed Cognition (Perry, 1999) involves one person working with external devices and using them to support their own work. The external devices are used
to augment the individual’s cognitive ability by offloading some of the mental activity by sharing the cognitive load (Hermann, 2005). This is sometimes referred to as external cognition (Norman, 1994; Sharp, Rogers & Preece, 2002). External devices that aid cognition are tools for representing and manipulating symbols (Simon, 1996; Green & Blackwell, 1998). Simple tools, such as paper, are able to present and store information. Advanced tools, such as computers, can do computation and reasoning. Hermann (2005) refers to complex cognitive tools as “cognitive prostheses” (p. 13) or more generally cognitive technologies. The tools are designed as an artificial extension for our brains. The most advanced cognitive technologies are able to reason and solve complex problems (sometimes even better than humans). However, simple cognitive technologies can be just as useful as advanced technologies (Nardi, 1993). This is an essential point when analyzing and designing cognitive tools. The usefulness of the technology is not based on the technological capacity, but rather on how well it supports the task that the human is performing. Tools involved in individually Distributed Cognition are most effective when they complement the human activity, rather than replace it.

Cognition is socially distributed when a group of people use technology or each other as a resource to help them think. Through interacting with the environment, a shared external space is created where the cognitive processes occurs. Technology can be used either as a facilitator for the cognitive work or as a medium for the work (Cartensen & Schmidt, 1999). In systems where cognition is distributed socially, there is no central agent that controls the entire system. Cognitive processes occur in parallel and
autonomously. The individual people in the system do not even have to be aware of the overall goal of the system. They only need to be aware of their own work and their own output. However, to get their own work done, they need to interact with other agents in the system. The patterns of interactions that emerge are important characteristics of the cognitive system as it affects information flow.

Table 2 shows the differences between individually and socially Distributed Cognition and how researchers would approach each for their study (Perry, 1998).

Table 2.

*Comparison of individually and socially Distributed Cognition (Perry, 1998)*

<table>
<thead>
<tr>
<th>Features</th>
<th>Distributed Cognition Type</th>
<th>Individual</th>
<th>Social</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td></td>
<td>Centralized in the individual's mind</td>
<td>Emergent, arising out of the interaction of multiple actors</td>
</tr>
<tr>
<td>Tool use</td>
<td></td>
<td>Artefacts are used as cognitive resources</td>
<td>Artefacts are cognitive resources as well as mechanisms for coordinating social interactions</td>
</tr>
<tr>
<td>Cognitive approach</td>
<td></td>
<td>Serial cognitive process</td>
<td>Parallel, distributed process</td>
</tr>
</tbody>
</table>
Investigation method | Quantitative / Experimental or functional analysis | Qualitative or interpretative analysis
--- | --- | ---
Focus | Show where representation resides and where rules or processes can be externalized | Show the coordination of collaborative activities through an examination of the representations passing between actors.

**Research Questions**

The purpose of this study is to understand aspects of the serious game development environment that influence collaboration and communication. We are interested in how the environment is used to represent and transform the game through its development. Distributed Cognition is the theoretical framework used for this study. The focus is on describing the information processing characteristics of the socio-technical system. The following research questions will be used to guide the study:

- How does the environment affect collaboration?
- How is information shared?
Cognitive ethnography is a variation of ethnography that emphasizes how information is constructed and used (Williams, 2006). Information, knowledge and meaning are constructed through human activity and is embedded in language, social structures and artefacts. Ethnography is a method to learn about these activities by examining the activities in-situ, as well as looking at the language, social structures and artefacts associated with the activity. Examining how information is processed in an environment is the same thing as examining culture (Holland et al., 2000):

"Culture emerges out of the activity of human agents in their historical contexts, as mental, material and social structure interacts, and on the other hand, that culture in the form of a history of material artefacts and social practices, shapes cognitive processes, particularly cognitive processes that are distributed over agents, artefacts and environment." (p. 178)

Traditional ethnographic methods require intense and long study of the system. Some principles of ethnography are listed in Table 3 (Ball & Ormerod, 2000).
Table 3.

*Principles of traditional ethnography (Ball & Ormerod, 2000)*

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Situatedness</td>
<td>The data is collected by an observer who participates in the work or becomes a member of the community that they study.</td>
</tr>
<tr>
<td>Richness</td>
<td>The data is collected from a diverse number of sources and captures a large range of activity.</td>
</tr>
<tr>
<td>Participant autonomy</td>
<td>The researcher has no strict protocol for doing the ethnography. Instead, the researcher is expected to adapt and modify his actions to fit into the situation.</td>
</tr>
<tr>
<td>Openness</td>
<td>The researcher remains open to the possibility of unpredicted events or issues that may arise leading to a shift in the focus of the study.</td>
</tr>
<tr>
<td>Reflexivity</td>
<td>By virtue of being there, the researcher changes the behaviour that they seek to study. Rather than trying to eliminate this effect, they try to understand it.</td>
</tr>
<tr>
<td>Personalization</td>
<td>The researcher is allowed some degree of subjectivity and the report may be thematic or dramatic (Creswell, 1998).</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Self-reflection</td>
<td>The researcher is conscious and accepts that their past experiences and biases will affect the way that perceive events.</td>
</tr>
<tr>
<td>Intensity</td>
<td>The researcher strives to become a genuine member of the group being studied. This may require the study timeline span over years.</td>
</tr>
<tr>
<td>Historicism</td>
<td>The researcher always considers the context and cultural norms from an evolutionary and historical perspective.</td>
</tr>
</tbody>
</table>

For this study, some principles of traditional ethnography were omitted. Due to the pace of technological change and the objective to provide practical design rationale, this ethnography was shorter and more focused. The principles of intensity, personalization and openness are not essential in cognitive ethnography (Ball & Ormerod, 2000). It is difficult to justify performing full-length ethnographies for HCI research since time is constrained. One of the benefits of HCI research is its cost-effectiveness (Mantei & Teorey, 1988). In order to be cost-effective and relevant to developers, HCI research must provide answers within very short time frames. Another
requirement of HCI research is the observations and conclusions are verifiable. Less value is placed on personalization or retelling of the researcher’s experience (Ball & Ormerod, 2002). HCI research also addresses specific design problems. Thus, the ethnographic data has to be purposefully collected to answer pre-determined questions. Two methods based on ethnography that have been used in HCI research are rapid ethnography (Preece, Sharp & Rogers, 2002) and Contextual Design (Holtzblatt & Beyer, 2002). For this study, cognitive ethnography was used because it is was developed specifically to study Distributed Cognition and “information use and its propagation through the cognitive system” (Sharp, Robinson, Segal & Furniss, 2006, p. 2).

Hutchins is credited with being the first to use cognitive ethnography in his study of ship navigation (Hutchins, 1995). It has since been applied by other researchers to study other situations including pair programming (Flor & Hutchins, 1991), office work (Neilson & Bodker, 2004) and medical dispatch (Furniss & Blandford, 2006). We use it here to study game development.

**Data Collection Procedure**

This section details the data collection and analysis strategy. The field study was conducted over 10 working days (2 weeks). This was followed by an analysis phase and additional semi-structured interviews over the next 8 weeks. Below are the stages of the research project.

*Review existing data.* A year earlier, a Contextual Design study was performed at the same company by the same researcher and a colleague (Kauhanen & Tran, 2007).
The information gathered included details about the team, workflow and their use of software products. That information provided direction for how this study could be conducted. Based on the results of the previous study, we were also able to generated a few initial questions and decide where to focus our observations.

**Become familiar with environment.** A first visit was made to the company. The objective was to become familiar with the environment and set up the observation equipment. We found a suitable location within the company that gave us easy access to the participants while not being obstructive. The first visit was also used to collect consent from the participants and inform them of the study (Appendix A).

**Conduct field observations.** Once all the consent and logistics were settled, the field study began. The observation sessions lasted 2 weeks. Each day, the researcher spent 3 to 5 hours at the company. Most of the time was spent observing what was happening in the main work area, but we also accompanied the participants on their lunch breaks. The lunch breaks were an excellent opportunity to learn about the culture of the team, especially the social relationships. As a courtesy, no note taking or audio recording was performed during breaks.

The objective of the field observations was to gather enough information to create three models of the work environment (Figure 5). Examples of the models from Furniss and Blandford (2006) can be found in Appendix B. The first objective was to create a physical model. Digital photos were taken of the room, desks and artefacts. Sketches were drawn and prominent artefacts were noted (Appendix C). The objective then shifted
towards learning about the people and their work processes. The physical model still remained important since work processes included interacting with the physical environment. The last objective was to understanding information flow and information use across people and artefacts.

We collected data with the use of digital audio recordings, digital photographs, observation notes and interview notes. To get observation notes, we sat at a desk in a central area of the main room where the participants worked. It was important not to interfere with any of the interactions or interrupt the work. We wanted the interactions to be natural and accurate. It was also an issue of courtesy to allow the participants to work without being interrupted since their work requires a high level of concentration. When the opportunity was available, questions were asked informally about their work. The informal interviews typically only lasted a few minutes and were done to learn about a task that was recently completed. Towards the end of the field study, semi-structured interviews were scheduled with selected individuals to clarify remaining issues about the artefacts, information flow and physical layout. The semi-structured interviews were conducted near the end of the field observations. The semi-structured interviews focused on checking the accuracy of the preliminary models and expanding on any issues the participants felt should be included in the analysis.
Prepare the data. After the field observations and data collection, all the notes, photos, and audio recordings were compiled and organized. The audio recordings were transcribed into a text format to be imported into qualitative data analysis software tools. The transcription was done using Transcriber 1.5.1, an open source transcription software tool (Transcriber, 2008). Transcription of the audio provided us with a text corpus from which we could perform content analysis. The data was prepared by the researcher.

Data Analysis Procedure

Data analysis. We used three methods to analyze the data; Distributed Cognition of Teamwork (DiCoT), content analysis and affinity diagramming based on Grounded Theory. DiCoT was created by Furniss & Blandford (2006) as a method to analyze socio-technical systems. It includes analysis of the physical environment, artefacts and work procedures. It relies on the researcher making sense of the observations and interview notes, then using principles and heuristics based on Distributed Cognition and user-centered design to create work models and recommendations. The result is a description of the work environment, highlighting issues and solutions to improve communication and collaboration.
The second analysis was a content analysis. This was done on the entire text corpus generated from the transcriptions. It involved classifying what was said, by whom and for what reason. The text was coded in HyperResearch (ResearchWare, 2008). Additional analysis was then performed using 'R', a programming language and environment for statistical computing (R-Project, 2008). The objective was to find patterns of communication that would be relevant to the discussion of collaboration and game development.

The third analysis method was based on Grounded Theory techniques. This method required manipulation of the textual data by performing a series of "coding" exercises (Creswell, 1998). The goal was to build a theory about the work domain, but keeping communication and collaboration as the central question. The content analysis exercise started by finding recurring patterns of action or speech when there was collaboration (open coding). Then we came up with ideas about how the patterns relate to each other (axial coding). Finally, an attempt was made to consolidate the ideas and find a theory that answered the question of how collaboration occurred in the environment (selective coding).
Results

This section provides the general results from the observations. Specific results from DiCoT, content analysis and Grounded Theory are presented in the section following this one. The location of the observations was a game development company in Ottawa, Ontario. For purposes of confidentiality, the name of the company and employees will not be disclosed. The company employs between 25-35 people. The study focused on the work of six employees, collectively referred to as the Studio team. The Studio team is responsible for all of the game design, some of the game development and most of the game testing. The game development platform is Abode Flash (Adobe, 2008). One of the advantages of Flash is the rapid application development because of the lightweight design of the technology and sophistication of the accompanying development tools. Using Flash also allows the game to be delivered and played over the Internet. The Studio team typically produces games in 8 to 16 weeks. Development time of games depends on several factors including complexity, client management and concurrent projects.

We observed the ending of the production phase and some of the testing phase. The main task of the team during the two week study was to polish and finalize the game for launch. This required making many small scale changes to the game and fixing any remaining issues (or 'game bugs'). Most of the game design and game programming had been completed at this point. Characteristic of this phase is a high volume of work as the deadline for releasing the game is near. It is usually a more stressful time for the team
because of the deadline. They were working on completing two games concurrently. The games will be referred to as Game1 and Game2. Game1 is for training employees about workplace safety. Game2 is for training employees on quality management. Table 4 shows the tasks of the Studio team and data collection methods.

Table 4.

*Field study timeline and events*

<table>
<thead>
<tr>
<th>Days</th>
<th>Studio Team Main Task</th>
<th>Data collected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 4</td>
<td>Resolve issues in Game1</td>
<td>• Audio recordings</td>
</tr>
<tr>
<td></td>
<td>(prepare for game release)</td>
<td>• Ethnographic observations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Photos and video recording taken of physical environment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Interviews with studio team</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Collected official job descriptions</td>
</tr>
<tr>
<td>5 - 9</td>
<td>Re-design Game2 (new iteration)</td>
<td>• Ethnographic observations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Audio recordings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Interviews with studio team and other employees</td>
</tr>
</tbody>
</table>

The Studio team has six members. Their official job descriptions can be found in Appendix G. The Studio members are:

- Lead Graphic Artist (LAR)
Figure 6 shows the where Studio team is situated within the game development process. The Studio team is the main team that does game design and game development. Some of the game design can be done in collaboration with the Research & Development team. The Studio team also tests games, but that responsibility is shared with the Engineering team and the clients. The clients, Studio team and Business Development team work together to create the game content (game text and learning material). As shown in Figure 6, a typical game starts as a request from a client. The Research & Development team will then consider the requirements. If the game can be created with existing technology and content, then the Studio team can start designing the game immediately. Otherwise, the Research & Development team will start prototyping new technology with the Studio team. The Studio team proceeds to create the artwork and other game content. They work with the Engineering team to get the game up and running. The Engineering team helps with the 'back-end' of the game (i.e. delivery and platform). Throughout game development, Marketing & Business Development will communicate with the Studio team any additional client requests and provide the clients with interim designs for evaluation. The final game is then released to the client when the
company and client are satisfied with the product.

Figure 6. Teams involved in game development process.

This study only describes the work of the Studio team. Using the Distributed Cognition framework, we looked at the social and technical components of the team environment. The team environment (or socio-technical system) includes “the people, the processes and all objects in the environment used by the people to complete the processes” (Harvey, Robertson & Edwards, 2004). Figure 7 highlights some components in the environment. It includes the human agents, electronic files, computer software, the World Wide Web, storage networks, whiteboards and users. The human agents are the Interface Designer (GUI), Studio Manager (MAN), Usability Expert (UE), Lead Artist (LAR), Artist (ART) and Game Programmer (GP). Of relevance to the Distributed Cognition framework is to consider how each component contributes to the cognitive functioning of the environment. Each component can be shown to have one or more of
these functions; sensing ability, information storage capacity, decision making ability or an output mechanism. These are basic functions of a cognitive system. Thus, the whole team environment, inclusive of people and artefacts, is said to be a cognitive entity to the extent that it can demonstrate those functions.

![Figure 7](image)

*Figure 7. Social and technical components involved in the development process.*

**Distributed Cognition of Teamwork (DiCoT) Analysis**

This section is the results of the Distributed Cognition of Teamwork (DiCoT) analysis. It includes three diagrams and descriptions of the socio-technical environment. Towards the end, we include recommendations for redesign that could increase collaboration and communication. The method borrows many ideas from Contextual Design (Beyer & Holtzblatt, 1998). In particularly, it emphasizes the use of diagrams, rather than text, to present complex ideas. The resulting diagrams are then used as a tool
for discussing issues and trade-offs in the design. The goal of the analysis is to describe factors in the environment that “contribute positively and negatively to the performance of the system” (Furniss & Blandford, 2004). The descriptions that follow include a Physical Model, three Information Flow Models and three Artefact Models.

**Physical Model**

The first model is the Physical Model. It is an abstraction of the physical space in the Studio room. Relevant to collaboration and communication are proximity of the agents to one another, placement of shared artefacts and barriers to communication. The physical layout can be analyzed at different levels of granularity. For example, we can create the layout of the entire company. This would show how the functioning of the Studio team, as a whole, is affected by their placement in relation to other teams. A Physical Model could also show individual workstations. This would show how individual work is affected by the arrangement of the artefacts around their desk. We have chosen to create a mid-level model of the Studio room to show the interaction between agents (Figure 8). Figure 9 are photographs of the Studio room. Table 5 describes how the layout affects Distributed Cognition.
Figure 8. Physical layout of Studio room.

Figure 9. Photographs of the Studio Room.
Table 5.

| Summary | The Studio room contains two large desks and one small table. Each desk is divided into four work stations to accommodate up to four employees. At each desk, there is one partition that doubles as a whiteboard. The partition elevates about 20 inches off the desk. This essentially separates the desk in half, leaving two workers on each side of the partition. The partition is high enough to hide the work area of the workers on the other side, but low enough that workers can maintain eye contact when talking. A table is set up in a corner where short meetings can occur. Formal meetings are usually scheduled in a separate meeting room. The table also holds the printer. The printer can be used by other employees who are not part of the Studio team. Five whiteboards are mounted on the walls to provide a place to share writing and drawings. The Studio room layout is deliberately designed as a collaborative space. It is meant to provide easy access to other people and shared artefacts. |
| Details | Communication (Access to Agents) The entire Studio team sits close together with no walls separating any of them. This arrangement places virtually no barrier |
on verbal communication. Every worker is within listening distance and no shouting is required to get each other's attention. The Studio is also small enough that every worker is only a few steps away from each other. This makes walking to another workstation easy when working in pairs is required.

The arrangement of the workstations also complement the duties performed by the different members. Those with similar duties or complementary roles tend to sit closer together. For example, the Lead Artist and Artist sit next to each other. The Interface Designer and the Usability Expert also sit side by side. The Game Programmer is close to the Studio Manager. Essentially, those who communicate frequently sit closer to each other.

Access to Artefacts

The most prominent shared artefact in the Studio are the whiteboards. The whiteboard walls are hung on each wall in the Studio room. There are also smaller whiteboard partitions at each desk. The whiteboard partitions can be accessed while sitting down, but may require leaning forward a little. A more comfortable position to use the small whiteboards is to stand at the edge of the desk. Visibility of the whiteboards at the desk are limited to those sitting next to it. However, the whiteboard walls are visible to everyone.
Another prominent shared physical artefact is the progress report pasted on the wall, which is mainly used by the Studio manager to keep track of development milestones. It is a collection of spreadsheets that note progress in game development.

The physical layout supports cognition in terms of sensing, understanding and planning. Sensing is maximized by the open layout. There are no walls or other barriers that restrict sight or sound. The team is aware of anyone approaching and wanting to gain their attention. The whiteboards supports understanding and planning. They are excellent for recording, leaving messages and creating sketches. The team can easily make notes to share and develop their ideas collaboratively. Generating ideas and plans together facilitate the creation of a shared mental model. Ideas potentially spread within the group much more quickly because of these artefacts.

Further Notes
- The environment provides a minimal level of privacy. It is easy to see if someone is working or not.
- Only the Game Programmer, Usability Expert and Studio Manager have telephones. This suggests external communication is only essential to these three roles.
- The whiteboards at the desks are just high enough so that the
contents of the desk on the other side is hidden. This gives each person a slight bit of privacy.

- Headphones are occasionally used while working. The reasons are usually to either minimize distraction or reduce boredom.

- The whiteboards do not get erased often, allowing them to accumulate a lot of different drawings and writing from different phases of development. This gives the room a "lived in" effect giving passerby's the sense that work was done here. It may also act as cues for the team reminding them of decisions made at certain points of the development.

- The Usability Expert sits in the middle of all team members. This reinforces the idea of usability as central to the Studio team's work. Furthermore, the Usability Expert sits in-between the Interface Designer and Game Programmer. Thus, conversations between the Interface Designer and Game Programmer literally pass through the Usability Expert. This layout enforces the idea that interface design decisions go through usability assessments before it is implemented in the game.

- The Studio manager is frequently absent from his desk due to
other responsibilities within the company. This often leaves
the game programmer as the sole worker at the second desk.

- This model does not take into account what is afforded by the
electronic network. In some cases, the electronic network is
superior for collaboration and communication. This is
discussed in later sections.

- The company owns two printers. One of them is located in
the Studio room. This increases traffic into the Studio room.
It is a potential source of distraction. However, it also creates
opportunity for informal chatting with other employees.
These can sometimes lead to fruitful discussions, such as
when the Client Liaison Manager starts ad hoc discussions
the Studio team about client issues.

Issues
- The studio team is in relative isolation from the rest of the
company. No other teams are visible from their location. In
contrast, Research & Development, Marketing & Business
Development and Engineering are all within sight of each
other.

Information Flow Model

The Information Flow Model describes the workflow for resolving game issues
with a focus on the movement of information. Three Information Flow Models were created. The first model is a high level Input-Process-Output overview. The second model is a look at the communication channels in the processing stage. The third model is an adaptation of the second model, but focuses on how the information is used.

The first Information Flow Model is shown in Figure 10. The input are reports about game issues. These reports come from many source and includes sources external to the company. The report will trigger the processing stage. The processing stage involves solving game issues. The output is an updated game, minus the reported issue. Throughout the process, information is being created, manipulated and stored in a database.

![Figure 10. Information Flow Model 1](image)

The second model focuses on the processing stage (Figure 11). Each box represents a role in the information flow. The arrows represent the connections and information flow between roles. This is also referred to as the communication channel. Tables 7 to 13 explain this flow in detail. Roles do not have a one-to-one correspondence to Studio team members. Any member of the Studio team may assume one or more roles during the resolution process. However, certain roles are restricted to certain team
members (Table 6). This flow model is not based on formal procedures in the company. They were created based on the observations and interviews conducted with members of the Studio team.

Figure 11. Information Flow Model 2

Table 6.

Roles and their corresponding team members

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Team member</th>
<th>Role Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finder</td>
<td>All</td>
<td>Find issues with the game</td>
</tr>
<tr>
<td>Reporter</td>
<td>All</td>
<td>Create record of issue in database</td>
</tr>
<tr>
<td>Assessor</td>
<td>Studio Manager</td>
<td>Determine if issues are real and categorize them</td>
</tr>
<tr>
<td>Fixer</td>
<td>All</td>
<td>Troubleshoot and solve issues</td>
</tr>
<tr>
<td>Resource</td>
<td>All</td>
<td>Help troubleshoot and solve issues</td>
</tr>
</tbody>
</table>
Table 7.

*Information Flow 2 description*

<table>
<thead>
<tr>
<th>Process</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>I – Finder to Reporter</td>
<td>Issues are found in the game by the Finder. This gets the attention of the Reporter.</td>
</tr>
<tr>
<td>II - Reporter to Assessor</td>
<td>The Reporter notifies the Assessor an issue has been found. This is done by creating a record in the database</td>
</tr>
<tr>
<td>III - Assessor to Fixer</td>
<td>Once notified of the issue, the Assessor attempts to duplicate the issue and categorizes it. It will then be assigned to an appropriate Fixer.</td>
</tr>
<tr>
<td>IV - Fixer to Resource</td>
<td>If necessary, the Fixers will recruit the help of a Resource. Resources can clarify problems or help with the actual troubleshooting.</td>
</tr>
<tr>
<td>V - Fixer back to Assessor</td>
<td>After the issue is resolved, the record is updated in the database by the Fixer. The Assessor is notified of the update.</td>
</tr>
</tbody>
</table>
VI - Assessor to Output

The Assessor will test the issue to confirm if it is resolved. The final update is made to the database which archives the issue.

Table 8.

Information Flow 2: 1 - Finder to Reporter

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
<tr>
<td>Issues are found in the game by the Finder. This gets the attention of the Reporter.</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>The Finder's role is to test the game. When issues are found through testing, the Reporter is notified. The reporter will enter the information about the game issue in a database. There are four types of testing; usability testing, subject matter expert (SME) testing, targeted internal testing and untargeted internal testing. In usability testing, the Finder is usually someone who is paid to play the game and not an employee of the company. They are recruited through various means and are selected based on demographics. The Finder tests the game through 'free-play'. They play the game from start to finish at their own pace. In some cases, they may be asked to play specific scenarios or perform specific tasks within the game. When the Finder has</td>
</tr>
</tbody>
</table>
trouble progressing through the game or specified tasks, then the problem is noted as a game issue.

Subject Matter Expert (SME) testing is another way to find issues. In this case, the Finder role is assumed by a client. SMEs usually belong to the client organization and is not a member of the Studio team. Their normal job is to provide the learning material and game content. They test the games to ensure the content was added into the game properly. Thus, SME testing usually focuses on finding game content issues.

Targeted internal testing is a third way for finding issues. The Finder role is assumed by a Quality Assurance Technician (QAT). The QAT is a member of the Engineering team. The QAT performs expert walkthroughs and test plans. In expert walkthroughs, the QAT acts as a game player and plays the game to find usability issues. The QAT also performs test plans. Test plans are a more systematic way to test the game and are meant to test every possible action in the game. This is useful for finding issues that would rarely come up in normal gameplay. It also tests the robustness of the game code.

Untargeted internal testing is ad hoc testing by the Studio team. The Finder role can be assumed by any Studio team
member. As components are added to the game during development, they are tested on the spot. Problems can be dealt with immediately or later on. Problems that don't get solved immediately are entered into a database as an issue for future resolution.

Further Notes

- The database that stores the game issue information is called Mantis Bug Tracker (2008). The database is only accessible with a user name and password. Generally, access is only given to employees.

Issues

- Minor game issues sometimes do not make it into the database because it is inconvenient to document small changes.

Table 9.

*Information Flow 2: II – Reporter to Assessor*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Details

Once the Reporter is aware of an issue, they enter the details into the Mantis Bug Tracker (MBT). Details include status, name of reporter, severity, priority, date, person responsible to resolve issue and description. The Reporter will categorize the game issue. This category helps determine who will be responsible for resolving it. There are two broad categories; game client issues and game content issues. Game client issues deal with the game technology, such as platform and compatibility issues. These issues are handled by the Engineering team. Game content issues are handled by the Studio team. The role of the Assessor is to verify game issues and assign them to a Fixer within his team.

Further Notes

• Issues can be discussed face-to-face or online. Information from face-to-face communication usually does not get documented. Therefore, that knowledge remains 'in-the head' of the team and not stored in any external artefact, such as the database.

Issues

• Issues can be mis-categorized, which directs it to the wrong team within the company. This causes delays in the resolution process.
Table 10.

**Information Flow 2: III – Assessor to Fixer**

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>
| **Further Notes** | - Some issues require collaboration between Studio team members.  
  - Fixers determine the time when issues are resolved based on their own schedule. The order in which issues are resolved does not necessarily correspond to the priority or severity assigned by the Assessor. |
Issues

- The database does not track the progress made by Fixers on particular game issues.
- The Assessor cannot easily predict when issues will be resolved.
- Issues can be mis-assigned, causing delays.

Table 11.

*Information Flow 2: IV – Fixer to Resource*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
<tr>
<td>If necessary, the Fixers will recruit the help of a Resource. Resources can clarify problems or help with the actual troubleshooting.</td>
</tr>
<tr>
<td>Details</td>
</tr>
<tr>
<td>Issues are usually small enough that a single Fixer can resolve it alone. However, some issues may exceed the expertise or knowledge of one Fixer. When this happens, the Fixer asks for the help of a Resource. The role of the Resource can be assumed by anyone from the Studio team (or even anyone outside the Studio team).</td>
</tr>
</tbody>
</table>
Further Notes • The communication channel between Fixers and Resources is mostly informal. An analysis of this communication content is provided in the next section.

Issues • Since the communication is informal, only a fraction of the conversations between Fixers and Resources are ever documented.

Table 12.

*Information Flow 2: V – Fixer to Assessor*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>

Further Notes None

Issues None
Table 13.
*Information Flow 2: VI – Assessor to Output*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
<tr>
<td><strong>Further Notes</strong></td>
</tr>
<tr>
<td><strong>Issues</strong></td>
</tr>
</tbody>
</table>

The Information Flow Model 3 (Figure 12) shows how information is used at the different stages of the information flow. The previous model described the communication channel. This model describes how the information is being “filtered, transformed and buffered” (Furniss & Blandford, 2004) along the communication channel. Tables 14 to 18 describe the model in detail.
Figure 12. Information Flow Model 3

Table 14.

Information Flow 3 description

<table>
<thead>
<tr>
<th>Information Life Cycle</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>I – Created</td>
<td>Information about game issues are generated by Finders. Reporters also generate some information by expanding on what the Finder's have found. Reporters can also elicit information from Finder's that otherwise would not be included in the issue report.</td>
</tr>
<tr>
<td>II – Filtered</td>
<td>The Assessor filters the information before it moves further into the processing stages. This is also a decision point to determine where the information will go next.</td>
</tr>
<tr>
<td>III – Updated</td>
<td>As issues are resolved, the information is updated in the database to reflect its current status. When information is not updated, the information flow slows down.</td>
</tr>
</tbody>
</table>
IV - Archived

Game issue information is archived in the computer system indefinitely once the issue is resolved and tested.

Table 15.

*Information Flow 3: I – Information is created*

| Analysis | 
| --- | --- |
| **Summary** | Information about game issues are generated by Finders. Reporters also generate some information by expanding on what the Finder’s have found. Reporters can also elicit information from Finder’s that otherwise would not be included in the issue report. |
| **Details** | Generating game issues is important for the development of the game because finding and eliminating game issues makes the game product better. Information about issues are generated by testing or playing the game as described in Table 10. Finders communicate issues to Reporters. Reporters usually expand the information and then enter it into the database. New information in the database starts the information flow (and issue resolution process). |
Further Notes

- Generating issues can be planned or unplanned. The majority of information currently comes from planned methods, such as usability testing, SME testing and targeted internal testing.

- Issues are usually found in bunches. This means at times there will be a long list of issues in the database. This creation stage can also be considered the buffer stage since information waits in a queue until action is taken upon it. This can take up to 6 weeks. Information in the queue after 6 weeks is discarded.

Issues

- Finders usually require the assistance of Reporters to get information into the database.

- All reports are generated manually.

Table 16.

*Information Flow 3: II—Information is filtered.*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Details

Information about issues are entered in the database. The Assessor is notified once this is done by the computer system. The Assessor then has to look at the information and decide if it is a real issue. Some issues are too minor to spend the resources required to fix them. Other issues are too complex and would better be handled through a more elaborate design process (rather than with the issue resolution process described here). The Assessor also decides who will fix the issue. Additional information about the issue is added by the Assessor, such as placing it into categories. This is also a decision hub in the information flow. The Assessor decides if the issue is worth fixing. If it is, then he decides which Fixer will get the information.

Further Notes

• Issues can be assigned or re-assigned face-to-face. In that case, it is important to update the database so that the information remains accurate.

Issues

• Filtering the information to the wrong place will cause delays in getting issues resolved.
Table 17.

*Information Flow 3: III – Information is updated*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
<tr>
<td><strong>Further Notes</strong></td>
</tr>
</tbody>
</table>
unresolved issues after 6 weeks is that if after 6 weeks it is unresolved, it probably is not important enough to resolve anyways.

Issues

- It is difficult to predict when issues will be resolved. It is not based exclusively on severity or priority of the issue. It is left to the Fixer's discretion how they use that information.
- The resolution of one issue may require more than one Fixer. However, issues can only be assigned to one person at a time.

Table 18.

Information Flow 3: IV – Information is archived

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
<tr>
<td>Details</td>
</tr>
</tbody>
</table>
Artefact Model

The Artefact Model looks at how artefacts are used to support and enhance information use. We discuss how the design of the artefacts affect collaboration and communication. We have chosen the three artefacts we believe have the greatest influence in the team environment. They are the:

1. Mantis Bug Tracker (MBT)
2. Whiteboards
3. The Electronic Network

1. Mantis Bug Tracker. The Mantis Bug Tracker (MBT) is a web-based bug tracking system (Mantis Bug Tracker, 2008). It stores information about issues in an online database. Users enter information about game issues (bugs) into the database. The information can then be accessed and modified by other users of the system. The main advantage of the system is its ability to document, sort and display game issues in one central place. Information is displayed as a dashboard (compiled lists) or as individual profiles. Figure 13, 14 and 15 show screenshots of the MBT. Access to the MBT is protected by username and password. Generally, access is not given to anyone outside the company. Even within the company, only those involved in issue resolution would need access. The MBT is used from the beginning to the end of the information flow /
resolution process. Table 19 shows how information in the MBT is updated to match the stages of the information flow. The MBT is also used to notify team members what issues they need to be working on. Studio team members usually check the MBT to find out their tasks. They can also find out other team member's tasks. Thus, maintaining up-to-date and accurate information in the MBT is important because it is used to schedule work and track game progress.

Table 19.

*Relation of Information Flow 2 to status changes in Mantis Bug Tracker*

<table>
<thead>
<tr>
<th>Information Flow</th>
<th>Corresponding status changes in MBT</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Finder to Reporter</td>
<td>New</td>
</tr>
<tr>
<td>II Reporter to Assessor</td>
<td>New, Feedback, Acknowledged</td>
</tr>
<tr>
<td>III Assessor to Fixer</td>
<td>Confirmed, Assigned</td>
</tr>
<tr>
<td>IV Fixer to Resource</td>
<td>Assigned</td>
</tr>
<tr>
<td>V Fixer back to Assessor</td>
<td>Resolved</td>
</tr>
<tr>
<td>VI Assessor back to Reporter</td>
<td>Closed</td>
</tr>
</tbody>
</table>

Below are three screenshots of the Mantis Bug Tracker. The first screenshot is the dashboard view showing grouped issues. The second screenshot is also a dashboard view, but has all issues listed in a single column. The third screenshot is a profile view showing information for only individual issues. Table 20 describes the two dashboard views. Table 21 describes the profile view.
Figure 13. Mantis Bug Tracker screenshot 1

Figure 14. Mantis Bug Tracker screenshot 2
Table 20.

Mantis Bug Tracker dashboard view description

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>
starting with the most recently modified entry.

The alternate dashboard view is to have the issues displayed in a single column (Figure 14). Each issue is color coded and can be clicked on to bring up the profile view (Figure 15), which displays detailed information for a single issue.

Further Notes

- The views can be tailored for specific work roles. For example, an Assessor could set the system to only display 'unassigned' and 'resolved' issues since the Assessor role is only interested in these two types. Reporters would filter their dashboard to view only issues they entered. Fixers can view issues that are only assigned for them.

- Issues are also color coded by status. The status of an issue goes from new (peach), feedback (pink), acknowledged (orange), confirmed (yellow), assigned (blue), resolved (green) to closed (grey). The legend is shown at the bottom of Figure 13. The color codes are a visual indication of progress. If the issue list is mostly green, then it means a lot of issues are resolved. On the other hand, if there is a lot of peach, then there are a lot of new issues that need to be resolved.

- To minimize the length of the list, the user can specify filters that will hide some issues. However, the filter option is only
available in the alternate dashboard view.

<table>
<thead>
<tr>
<th>Issues</th>
<th>• The MBT only does a fair job of allowing customization of the dashboard views. For example, there is no way to use filters for the first dashboard view.</th>
</tr>
</thead>
</table>

Table 21.

*Mantis Bug Tracker profile view description*

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td>The profile view shows one issue at a time, but provides all the details associated with that issue (Figure 15).</td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
<tr>
<td>Basic information that is shown on each profile include:</td>
</tr>
<tr>
<td>• ID, a unique identifier</td>
</tr>
<tr>
<td>• Category, the aspect of the game that the issue relates to such as content, functionality and network</td>
</tr>
<tr>
<td>• Severity, the criticality of the issue</td>
</tr>
<tr>
<td>• Reproducibility, how easily is the issue reproduced</td>
</tr>
<tr>
<td>• Date Submitted, when was it first reported</td>
</tr>
<tr>
<td>• Last Updated, when was it updated in the ITS</td>
</tr>
<tr>
<td>• Reporter, who reported the issue</td>
</tr>
<tr>
<td>• Assigned To, who is working on resolving the issue</td>
</tr>
</tbody>
</table>
Further Notes

- Priority, how urgently does the issue need to be resolved
- Status, which stage is the issue currently in

The basic information also includes a summary, long description and attached files. The profile view also includes a discussion thread and history of status changes.

Issues

- The discussion threads are very brief and sometimes there are none. Since team members are physically present, it may be easier to talk face-to-face rather than use the discussion threads.

- Severity and priority are not explicitly used by Fixers when they decide which issue to resolve first. This makes it hard for others to predict when issues will be resolved. The order they resolve issues is determined by the Fixer's schedule. They try to maximize efficiency and convenience. An example would be if they just completed a task and there are 10 minutes left until lunch. They may spend the 10 minutes working on a low priority issue that is fixable within that time frame instead of starting an issue that is higher priority but would take more time.
2. *Whiteboards.* Whiteboards are writable and erasable surfaces in the physical space of the Studio room. They are placed strategically throughout the room for visibility and convenience. They are used by the Studio team to sketch game scenarios, mock up interfaces and schedule work. Table 22 provides a description of their use. They are a dominant feature of the physical space, both at the room level and desk level. Figure 16 shows where the whiteboards are located in the Studio room. The information on the whiteboards are useful mainly in the short-term, but information tends to stay on long-term. During the 2 week study, the contents of the whiteboard were not erased even though the contents of the whiteboards were not needed for that stage of development. It seems the whiteboards are only erased when space is needed to draw new things. Otherwise, the whiteboards are left *dirty.*
Figure 16. Whiteboards in Studio room.

Table 22.

**Description of whiteboard placement.**

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>

There are nine whiteboards in the Studio room. Five are wall-mounted whiteboards. Four are double sided desk partitions.
Details

One whiteboard contained a drawing of an entire game scenario. Another whiteboard contained information about the game development life cycle process. A third whiteboard had basic game information, such as title and client. The fourth whiteboard contained only doodles. There are also sketches and scribbles on most of the whiteboards that do not seem to fit in with the rest of the content.

The wall mounted boards tend to have information that are of higher level concepts, such as game information and developmental progress. The smaller whiteboard partitions had sketches of wireframes, prototypes and work plans.

Further Notes

- There are no formal rules for writing on the whiteboard. Any member can write anything. There are also no rules for keeping the whiteboards clean. Content is deleted only when whiteboard space is needed.

Issues

- The frequency of use of the whiteboard change depending on the phase in development. During the production phase, whiteboards use was infrequent.

3. The Electronic Network. The electronic network refers to both the Internet and Intranet. The network contains information repositories, shared storage for electronic files
and group applications. The Internet is a world wide network whereas the Intranet is the company's private network. Information from the Internet is useful for gathering knowledge, such as programming techniques or getting ideas about how game objects are drawn. Information on the Intranet is specific to the company, such as work procedures and game documents privy only to Studio team members. The main resources from the Intranet are the file server and Mantis Bug Tracker. Another part of the network is the game portal. It delivers games to the clients through a secure network. Employees also have access to the game portal for testing purposes. The value of the network artefact is extremely high. However, this point is usually overlooked because the network is so pervasive. Without it, sharing information would be very difficult and cumbersome. More importantly, without the network, there is likely no game product because the game is created, sold and delivered via the network. Thus, it is essential to the work of the Studio team that the network is always available.

Summary of Design Issues and Solutions

In this section we summarize four issues relating to information use. The four issues are situational awareness, work queuing, role specialization and network dependence. We review these factors in turn and explain how they are affected by the physical layout, information flow and artefacts.

Situational awareness refers to the awareness each team member has about what is happening in their environment. Mainly, it involves knowing about what others are doing. In collaborative systems, it is important to be aware of what other individuals are
doing because this factors into how you plan your own work. In turn, others must be aware of what you are doing. Thus, there have to be mechanisms through which individuals can keep each other informed of their tasks and progression on those tasks. When team members share this understanding, they work together more effectively.

The physical layout of the room supports situational awareness well. With the open concept and low barrier to face-to-face communication, it is easy to determine what others are doing. The physical layout also makes it easy to broadcast work. It is common to hear the team talk aloud as they work, announce when they complete a task or share anecdotes of past experiences. The open space also makes it easy to ask questions and get answers.

One area where situational awareness can be improved is the sharing of screen information. Some work requires team members share a single monitor. Usually, one person will call the other to their desk for help. Being a small room, everyone is just a few steps away. However, there are collaborative tools that let people share their desktop display over a network. Some of these options could be explored as an alternative to physically sharing a single monitor. On the other hand, sitting next to each other can provide other benefits, such as face-to-face communication and not having to worry about poor technology impeding the work.

A good information flow channel also helps situational awareness by reducing what each person needs to know. Each agent in the information flow channel only needs to know about the person they are getting information from and the person they are
giving information to. However, in this particular case, the information flow is short enough that it is not a large burden for the team to be aware of the entire process. In some cases, they may prefer it. The information flow is sometimes not followed as described in our model. It is not uncommon for the Fixer to get issues directly from the Reporter. In that case, the Assessor is left out of the information flow. The risk is that the Assessor may decide that the issue was not worth fixing in the first place, but the team will only know this after time and money was wasted fixing it. Another situation where the information flow channel can breakdown is when the Fixer finds an issue and does not report it. Instead, the Fixer resolves the issue without notifying anyone else. When this happens, there is usually no documentation of the work done. The team is not aware of any game changes until after they are made.

A breakdown in the information flow channels reduces situation awareness of the team, but tends to increase the speed of getting things done. At the same time, it introduces the risk of doing things that are not worth the effort. The effectiveness this process depends on balancing the adherence to information flow channels (i.e. communication protocol) with ad hoc decision making.

The artefacts are also essential tools for supporting situation awareness. The MBT allows team members to track the number of issues and who is supposed to be working on them. Determining how long it would take to resolve an issue is harder to determine. The whiteboards support situational awareness by encouraging sharing information via public information displays. When someone is writing on the whiteboard, everyone can
see what they write. Unfortunately, we were not able to observe enough use of the whiteboard to make sound comments regarding their use, but there is literature to support the usefulness of whiteboards in software development (Cherubini, Venolia, DeLine, & Ko, 2007). We believe they may be used more at other stages of development, particularly the design phase when more brainstorming occurs. The network is another tool that increases situational awareness. It allows team members to communicate asynchronously. Also, any collaborative software would likely rely on having a network. Thus, it is essential that there is a fast and stable electronic network in place so that these tools can be accessed.

*Work queuing* is about prioritizing tasks based on the work and requirements of others. It means timing your work so that the necessary pieces (inputs) are available when you want to start on your task. Alternatively, it means having your work completed (output) so that others can use your work to start on their tasks. This issue is important because having effective work queues means less waiting to start on tasks. The multi-disciplinary nature of the team adds another dimension to this problem because it is difficult to predict when others will have their output ready. A game programmer cannot be expected to know how long it takes to generate usability reports or an interface designer would not know how long it takes to draw a game object.

The physical layout does not directly address work queuing. However, it affects situational awareness. Situational awareness can help with creating better work queues because effective work queues relies on knowing how far others are progressing on their
tasks. Increased situational awareness makes it easier to synchronize tasks. The open physical space helps by making other's work visible and makes it easy to ask for updates about progress verbally. The physical layout also lets everyone see who is presently working or if they are absent.

The information flow makes clear the inputs and outputs required by each actor in the work process. It also makes it clear how each agent contributes to the work process. Using information flow models, we can identify areas where one agent may be getting too much information and unable to deal with it in time, thus creating a bottleneck. The result is an inefficient work queue because time is wasted waiting on work to be done.

The MBT, whiteboard and network artefacts indirectly support better work queues by increasing situational awareness. The network is arguably the most essential artefact for this since software can be installed on it designed to address the work queue issue directly. In larger teams or for more complex work processes, software support might be essential. However, the information flow of the Studio team is short and they communicate well enough that additional software may not offer substantial benefits.

*Role specialization* refers to the difference in ability and contribution of each team member of a team. Teams with high role specialization are suited for work that is “driven by externally imposed pacing and work cycles that are often brief and take place under new or changing conditions” (Wickens, Lee, Liu & Becker, 2004). An important finding regarding these types of teams is that team performance is affected by how well the members are aware of each other's roles and responsibilities (Salas et al., 1999).
The physical layout does not allow people to easily identify specialized roles. It is an egalitarian or communal set-up where everyone is in the same general space and has access to the same things. The room is shared and almost nothing is kept privately. There are no walls separating anyone. There are also no signs indicating roles or status. There is one exception one to this. Different team members have different work tools at their desks. The Artists use electronic drawing pads. The Game Programmer and Usability Expert have telephones. Each of the workstations are also decorated with trinkets and ornaments to express individual style. However, the decorations are not regarded as status symbols, nor do they provide strong clues of roles. Thus, for a new comer, it may take a little longer for them to identify the different roles. However, the Studio team is small enough that this should not be a problem.

The MBT is the most relevant tool for role specialization. Different roles use different features and extract different information from the MBT. The studio manager is interested in getting information about all issues in the game to determine how many more need to be resolved. The other team members may only be interested in the issues they can fix or are assigned. The artists are interested in issues relating to graphics. The game programmer is interested in issues concerning game implementation. The interface designer is interested in interface issues. The usability expert is interested in user experience issues. Features in the MBT should be made specific to the specialized roles. The usability expert would most likely need tools to help with data entry whereas the manager might rely on filter and display options more. An analysis of the requirements
for each job role could help generate ideas for improving the MBT.

Network dependence refers to the work interruptions when a connection to the network cannot be established. It is an extremely rare occurrence. When it does occur, it can usually be diagnosed and fixed within days. However, despite the low frequency, we believe the severity of the issue warrants giving it attention nonetheless. It is important to design an environment and work process that is resilient to network failure. During the two weeks of the study, we did witness the network fail in the Studio team. They could not get access to the Internet due to a router malfunction. While this did not significantly delay game production, it was very disruptive to the work flow. There was a clear sense of frustration when communication and access to the network was denied. One thing that the Studio could do is develop alternative work processes when the network is unavailable for long periods.

Recommendations and Design Alternatives

In this section we present design alternatives to the Physical Layout, Information Flow and Artefact models. The re-designs are based on applying general usability heuristics to the models we created. The first design change addresses the physical layout. The main objective of this re-design is to reduce encumbrance when moving around and reduce distraction when communicating in the physical space. This re-design is partially based on a link analysis exercise. Parush (2006) provides some ideas on how link analysis can be applied to re-designing work spaces. First, we identify all the agents in the system. Then we quantify their relationships through metrics or a matrix. The metrics
show the strength of the relationships based on frequency and importance of interaction between agents. After identifying the relationships and their strengths, they are overlayed on the physical model as lines of varying color or width. The agents are rearranged 'by-hand' until an ideal placement is found. The placement of the agents should allow closer proximity between actors who have a strong relationship. There should also be less crossing of linked paths. In Figure 17, we can see the link between the Interface Designer and Game Programmer crosses the Usability Expert. Figure 18 shows the re-design, eliminating the crosslink. It also creates a space where it is easier for everyone to access the Game Programmer and Studio Manager, who both have central roles in development. Table 23 summarizes the advantages and disadvantages of the re-design.

*Figure 17. Current arrangement of agents in Studio team.*
Table 23.

Claims analysis of physical layout recommendations

<table>
<thead>
<tr>
<th>Design Feature</th>
<th>Hypothesized Pros (+) and Cons (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminate partition between Artist (ART) and Lead Artist (LAR)</td>
<td>+ Facilitate collaborative work between ART and LAR by providing easier face-to-face communication and a shared physical desktop</td>
</tr>
<tr>
<td></td>
<td>- Less privacy for ART and LAR</td>
</tr>
</tbody>
</table>

Figure 18. Recommended arrangement of agents in Studio team
Move Interface Designer (GUI)  
+ GUI and Game Programmer (GP) do not have to talk 'past' the Usability Expert (UE)
+ UE still has access to conversations between GUI and GP
+ Easier communication between UE and GP because partition is removed
- Less privacy for GUI

Larger space in middle of room  
+ More space to assemble around workstations for working in pairs, particularly around GP workstation
+ More room to walk between workstations
+ Partition whiteboards are viewable by more people; thus, they can be shared by more people.
- Loss of personal space

Square workstations and linear arrangement

The second re-design deals with the Information Flow Model. Currently, information flows sequentially. Potential bottlenecks are at the Assessor and Fixer. One solution is to create an additional decision point to reduce the volume of information passing through the Assessor (Figure 19). The new decision point could be at the Reporter. The Reporter would be given the ability to assign issues directly to Fixers. Only
issues that have been pre-defined and categorized (i.e. Pre-Assessed) would skip the Assessor. Generally, these would be common issues that re-occur in many games and whose fixes are simple. Table 24 contains additional recommendations.

![Figure 19. Recommended Information Flow Model.](image)

Table 24.

**Claims analysis of recommended information flow**

<table>
<thead>
<tr>
<th>Design Feature</th>
<th>Hypothesized Pros (+) and Cons (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Give Reporter ability to assign some issues directly</td>
<td>+ Reduce volume of issues Assessor deals with + Issues get to Fixers sooner - Increases chance of mis-assigning issues</td>
</tr>
<tr>
<td>All unresolved issues go to Fixer and Resource</td>
<td>+ Resource can decide to fix issues if they are capable instead of waiting for Fixer to request help - Resource may get too much irrelevant information</td>
</tr>
</tbody>
</table>
Some issues can be automatically generated and input into the MBT, such as error logs. Information may be less meaningful if computer generated.

There are also some re-designs for the artefacts. These are summarized in Table 25. Regarding the MBT, one solution for increasing situational awareness is to provide a way to indicate in the system which issues are currently being worked on. One way to do this would be a small application that sits on the desktop. Each user would input a task and estimate the completion time (Figure 20). The tasks can be populated from various sources; one of them being MBT database. The MBT would collect information from the application and provide one screen showing what each team member is working on and their estimated completion time. Other modifications to the MBT would be designing the interface to suit the informational needs of each role within the team.

The functionality of the whiteboards can also be extended by digitizing their contents and connecting them to the network. This would allow the contents of the whiteboard to be stored long-term. It would also facilitate sharing since they would be viewable from the desktop. This is not a new idea and is already a commercial product (Smartboard Technologies, 2008). Smartboard Technologies (2008) also allows projecting your desktop onto the whiteboards, which could help when working in pairs or more. However, this solution is expensive, requires software and the return-on-
investment is not clear.

Table 25.

Claims analysis of recommended artefact designs

<table>
<thead>
<tr>
<th>Design Feature</th>
<th>Hypothesized Pros (+) and Cons (-)</th>
</tr>
</thead>
</table>
| Allow desktop to be projected onto walls| + Reduce need to share monitors when working together  
 + Easier to present work  
 - More distracting to others not involved in collaboration |
| Save whiteboard contents electronically | + Electronic information is easier to move, share and store  
 - Electronic mediums tend to be more restrictive  
 - Software may be prone to error |
| Software tool allowing update and notification of work status | + Allow others to know what work is currently being worked on and when it will be completed (Figure X)  
 - May threaten autonomy |
Figure 20. Recommended application to increase situational awareness.

DiCoT Summary

The DiCoT analysis highlights how the design of the physical layout, information flow and artefacts affect collaboration and communication. The physical layout encourages collaboration with an open space design. Each team member has quick access to other members physically, verbally and visually. They all sit in the same room with no walls separating them. Members easily see and hear where everyone else is and what they are doing. The open spaces also give them easy access to shared artefacts such as the whiteboards and lets them communicate when using the artefacts.

The information flow ensures there is a clear communication channel and that members can identify their role within the work process. The information flow supports collaboration by eliminating confusion. Team members know what they are responsible for and who they need to interact with. With the Studio team, the information flow has not yet been formalized as a protocol. This means adherence to the information flow is
sometimes lax. The team is given freedom to adjust their work procedures based on the situation, such as when the manager is absent. Nonetheless, a normal information flow does exist and provides the common ground for situating ones place in the work flow.

The artefacts in the environment also support collaboration and communication. The main ones are the MBT, whiteboards and network. The MBT lets the team track and document game issues. Having updated records of game issues makes it easier for the team to coordinate their work because they know what has already been done and what needs to be done next. The dashboard view provides an organized and condense list of relevant game issues allowing team members to quickly gauge how much work is left to do. The MBT is also good for acquiring knowledge about how to resolve game issues since the MBT keeps records about all past issues. Another artefact for collaboration are the whiteboards. These offer places for team members to share their ideas informally. Since the whiteboards are meant to hold transitory information, team members do not have to worry about being neat and formal. This encourages them to jot any ideas, even if the ideas are not fully formed. Having it written and available for others to reflect on help the ideas develop collaboratively. There is also the electronic network that connects all of the team members. It allows them to communicate and share their work. The Internet is used to get ideas and information via the World Wide Web. By staying connected to a larger community, the team can assimilate and adapt new knowledge from outside the company to improve their own work.
This DiCoT analysis provides recommendations to improve the work environment. We identified four salient factors that affect Distributed Cognition: situational awareness, work queues, role specialization and network dependence. A re-design, based on usability heuristics, for the physical layout, information flow and two artefacts were suggested to address these factors.

The advantage of using DiCoT as a basis for re-design is its broad perspective on task performance and usability. Traditional HCI approaches may focus too much on the single-user single-device interaction. DiCoT takes into account multiple-users and multiple devices. Furthermore, it does not limit itself to desktop computers and graphical interfaces. The analysis includes any cognitive artefact; acknowledging that people use all sorts of tools to help them perform mental tasks and they can use them in unexpected or unintended ways. People are very good at using their environment and adapting to whatever is available. Thus, DiCoT encourages field study type research where the actual use of products is observed in the natural setting.

In concluding the DiCoT analysis, we believe the Studio team and their environment are well designed for collaborative work. From the analysis, we identified issues surrounding the collaborative work. The re-designs that we recommend address the issues and should facilitate more collaboration. Next, we shift the analysis towards the human communication aspect of the socio-technical system. The goal is to draw insights for improving collaboration based on patterns of communication.
Content Analysis

This analysis seeks to answer questions about how collaboration occurs by studying the information content and patterns of face-to-face communication. It was done using the conversation data. We looked the content of all audio recordings during the two week study to determine what was being said, by whom and when. The data set consists of audio recordings of all conversations by the Studio team while we were present. The data was collected using a digital audio recorder that was placed in the center of the Studio room. The data set includes 30 hours of audio. It was recorded during the morning and afternoon between 10am and 5pm. It picked up all conversations within the Studio room and include conversations of anyone coming into the Studio room to talk with the Studio team. Thus, it was not limited to just chatter among Studio team members. The conversations also cover a wide range of topics and are not just work related. By recording and analyzing all conversations, we hoped to provide a more comprehensive understanding of the work environment.

The analysis was performed with two people to reduce bias that may happen with just one coder. One person was the researcher who performed the ethnography and is the author of this paper. The second person is a computer science graduate student who was familiar with the company and their processes. We chose to include someone who had a basic understanding of the work environment. Otherwise, it would have been too difficult to interpret the meaning of the text without some prior knowledge of serious game development practices.
The first step to analyzing the audio was to convert the audio format into a text format. The conversion of audio-to-text was done by ear. This was done with the help of Transcriber 1.51 (2008), a software program designed specifically to help with audio-to-text transcriptions. The data was then imported into HyperResearch (2008), a qualitative data analysis tool. HyperResearch was used for tagging and coding segments of the data. The final codes were derived after many iterations. We had four brainstorming sessions and three coding sessions. The brainstorming sessions involved listening to the recordings and deriving categories that described what was being said. This was an inductive exercise. New categories were created whenever necessary. At the end of each brainstorming session, the categories were reviewed and consolidated. The categories tried to capture information such as intent, agents, tone, timing and subject. At the end of the last brainstorming session, we agreed on a small set of categories. The categories are listed in Table 25. The categories were then used to tag every segment of the data set. The tagging (or coding) was done using HyperResearch (2008). The HyperResearch and Transcriber files were then modified and fed into R (2008), a statistical program, for further analysis. Custom scripts were written in R to aggregate and graph the results. Some graphs were created with a spreadsheet program. The results are presented in the next section.
Table 25.

*List of generated codes for content analysis*

<table>
<thead>
<tr>
<th>Codes</th>
<th>Related descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work</td>
<td>Work related conversations</td>
</tr>
<tr>
<td>Planning</td>
<td>Scheduling tasks, coordinating tasks</td>
</tr>
<tr>
<td>subject-game</td>
<td>Specific game issues, game objects</td>
</tr>
<tr>
<td>subject-tool</td>
<td>About the development tools</td>
</tr>
<tr>
<td>subject-metatool</td>
<td>About the tools for coordinating work</td>
</tr>
<tr>
<td>Non-Work</td>
<td>Non-work related conversations</td>
</tr>
<tr>
<td>subject-technology</td>
<td>Computers, computer peripherals not for work</td>
</tr>
<tr>
<td>subject-popculture</td>
<td>Music, movies, celebrities</td>
</tr>
<tr>
<td>subject-personal</td>
<td>Relationships, family, health, financial, rent</td>
</tr>
<tr>
<td>subject-ending work</td>
<td>Scheduling breaks, discussing taking time off work</td>
</tr>
<tr>
<td>subject-gossip</td>
<td>Gossip about other people</td>
</tr>
<tr>
<td>subject-leisure sports, non-work life</td>
<td>About weekend activities</td>
</tr>
<tr>
<td>subject-food</td>
<td>Food, scheduling lunch activities</td>
</tr>
<tr>
<td>subject-weather</td>
<td>Comments on the weather</td>
</tr>
<tr>
<td>Outside</td>
<td>Conversation involving Studio team and</td>
</tr>
</tbody>
</table>
other groups

Inside Conversations involving only Studio team members

Humor Humorous remarks or attempt at humor

Other Codes created, but not included in analysis; Closedloop, Openloop, Broadcasts, Artefacts

Content Analysis Results

The data was graphed by character count and by conversation. Graphs generated from character count are referred to as communication data. Graphs generated from conversations are referred to as conversation data. A conversation is considered a sequence of chatting preceded by 3 minutes of silence. Conversations lasted from 5 seconds to 15 minutes. In total, there were 154 conversations. The decision to use time to separate conversations rather than subject was for practical reasons. By not using subject, we did not have to worry about conversations that carried over multiple chat sessions and we did not have to worry about overlapping conversations. However, a result of using time to separate conversations is that a single conversation could include many topics. For example, the team could be talking about a particular game issue, then immediately talk about getting lunch without pausing. Thus, game issues and lunch would be considered topics within the same conversation. The graphs using communication data are shown first.

The Studio team communicates internally among themselves and also with other
groups in the company. We found nine percent of the communication involved people from outside the Studio team (Figure 21).

Next, we looked at the subjects discussed; 75% of all communication was work-related (Figure 22). When talking about work, 64% was about specific game components, followed by 23% about scheduling work (Figure 23). Only a minor amount of time was spent talking about the development tools and planning tools. This implies that the top two reasons for communicating is to (a) find out what's going on in a game or how to fix something in the game, and (b) coordinating work tasks.

Figure 21. Percentage of communication with only Studio members (Internal) versus Studio members with other groups (External).

Figure 22. Percentage of communication relating to work versus non-work topics.
25% the communication was non-work related. Of that percentage, 27% was about food. Personal matters, gossip and technology were talked about 18%, 17% and 14% of the time. The non-work communication can give us clues about the culture of the group. It shows where their common interests lie and what they are comfortable talking about as a group.

A social network of the studio team was created based on conversations. It shows the frequency of talk between team members. The thick lines indicate high frequency. For complete numbers and the social network matrix, see Appendix H. Figure 25 shows the Game Programmer (GP) and Interface Designer (GUI) talk most frequently to each other.
This is followed by GP and Lead Artist (LAR), then GP and Usability Expert (UE). The thing to note here is that the lines are thickest going towards the GP. This means the GP is getting the most attention from the team in terms of face-to-face communication. This graph reinforces the idea of the central role of the Game Programmer.

Figure 25. Studio team social network
A similar result is drawn from Figure 26. It shows the Game Programmer (GP) is involved in almost all the conversations (83%). The Interface Designer (GUI) and Lead Artist (LAR) are the next most active speakers at 49% and 47%. They are involved in almost half of all conversations in the Studio room. The remaining speakers are Usability Expert (UE) (38%), Artist (ART) (35%) and Studio Manager (MAN) (19%). The low involvement of the Studio Manager is partly due to his absence for the last four days of the study.

Another aspect of the communication was the humor content. There was clearly an effort by the Studio to make jokes and keep the conversations lighthearted. This could be attributed to the personalities within the Studio team and how well they get along. Humor was found in 12% of the communications. Figure 27 shows graph of humor occurrences on Day 8 of the study. Each of the vertical lines represent an instance where humor occurs. The black dots represent 5 minute conversation chunks. Dots appearing high in the graph signify conversations chunks where work-related content was
discussed. The higher the dot, the more work related content discussed. What this graph shows is that many of the conversation include an element of humor, even when the conversations are about work. Appendix I includes other humor graphs for the rest of the data. They show similar patterns where humor is used commonly when discussing work and non-work related subjects.

**Figure 27.** Humor in conversations – Day 8

**Content Analysis Summary**

The main points from the content analysis are there are frequent conversations, the Game Programmer receives the most attention, and work related talk is typically for learning about the game and scheduling work. This is also a friendly group who share things in common and are able to have small talk and joke to each other. Humor commonly appears in the conversations. The number of conversations this team has plus the content of what they talk about suggests face-to-face communication is integral to the collaborative process. However, based on this data alone, we cannot make any
conclusions about the significance of these findings. Without any comparable data, we assume that the communication is effective and helps collaboration.

This analysis is useful in a couple of ways. The first is that it provides some insight on what is needed by to the development team, especially relating to work. Their main topic of conservation are game issues, but second to that is work scheduling. Unfortunately, there is no software tool that allows them to coordinate their work hour-by-hour. There may be an opportunity here to introduce a tool (software or otherwise) that would allow team members to coordinate work more easily. The second way that this analysis is useful is by showing how face-to-face communication can be quantified. In doing so, we can begin to operationalize collaboration. Audio recordings, analyzed in this way, provides us with a means to measure the effectiveness of face-to-face communication and compare it against other outcomes.

*Grounded Theory*

The third and final analysis is based on Grounded Theory. Grounded Theory was developed by Strauss and Glaser (1967) as a way to generate theories out of a body of qualitative data. In this study, the technique was performed to generate themes and create connections between the themes to provide a model of the collaborative process. All the coding and organizing of themes was done without software. Although we have previously generated codes from the content analysis, they were not used to in the Grounded Theory analysis. We started with no codes and generated the new themes with a fresh perspective. These themes were then consolidated and arranged as an affinity
diagram (Beyer & Holtzblatt, 1998). Samples of documents produced during the process can be seen in Appendix D and Appendix E. At the end of the analysis, we created a model that represented our understanding of how all the themes were tied together.

In the next section, we present our model first to help the reader understand and frame the rest of the details. The *italics* represent the themes that were identified. To keep the section brief, we provided references for most of the themes, rather than cite passages directly. The numbers in brackets refer to line numbers in the data file where the theme occurs. The body of data used in the Grounded Theory analysis only includes work related communication. A sample of the text data can be found in Appendix F.

**Model of Collaboration**

Our model of collaboration among serious game developers is shown in Figure 28. Each box represents a collection of themes that were identified through the analysis.

*Figure 28. Theoretical model for collaboration in game development teams.*
The first column of themes are the contributing factors to the collaborative process. The elements of the collaborative act itself, are noted in the second column. Finally, the collaboration leads to the collaborative process noted in the third column.

**Contributing Factors to Collaboration**

The main contributing factors to collaboration are the *physical space*, the *social space*, the *team mental model* and *organizational goals*.

*Physical Space.* The layout of the room and availability of shared artefacts can be designed to facilitate and encourage collaboration. An *open layout* that does not use walls or cubicles to isolate team members encourages group conversations and informal cross-training. Group conversations are when more than two people are involved in a conversation. This allows additional viewpoints to enter the discussion and creates a greater diversity of opinions and ideas (280, 394, 835, 899, 2535). It also means design decisions have greater support because consensus is reached between more people. Informal cross-training occurs when members broadcast information about their experiences or progress for everyone to hear. In the *open layout*, no conversation is completely private. Thus, team members are always aware of the issues being discussed regarding the game product. Even though it may not be directly related to their immediate task, the information can be useful to gauging overall team progress. They are also informed about what other people are doing resulting in increased awareness about others team member’s work.
We also found the tendency to share experiences and expertise very high in among team members (24, 183, 547, 1420, 2574). This may be due to the increased opportunities afforded by the physical layout. The team does not have to plan meetings and schedule appropriate times to debrief. Everyone is always present and available to talk. Learning about experiences from others exposes each member to the different aspects of the game development process. One result is a team that is more empathetic to different disciplinary perspectives.

Another advantage of presence in a shared physical space is the ease of coordinating work (125, 475, 1313, 1810, 1904, 1929, 2183). Coordination is an important function because team performance is affected by how easily each member is able to synchronize their work with each other. For example, the work of the programmer requires the completion of work from the artists and GUI designer. Also, the usability expert relies on the programmer to finish his work so that she can test new versions of the game with users. With every team member physically present, communication and feedback can occur instantly (68, 168, 551). This allows them to assign tasks and get progress updates immediately. Being physically present also allows them to work-in-pairs (211, 273, 784, 1101, 2447). Working in pairs increases the quality of the work, encourages mentorship and increase communication. These advantages are well noted in the software development literature [14]. In this study, we notice mentorship was especially prevalent (211, 1929).

Social Space. The social environment is friendly and team members are
committed to learning from each other. Certain work roles also require two people work
together to complete a task. We call this collaborative partnerships. The game
programmer has the most collaborative partnerships because everyone needs to work
with the programmer to get their contribution put into the game. He is the only team
member responsible for compiling the game. Everyone else works on component pieces,
such as graphics, text or interface widgets, but the programmer is the only one who works
with all components. Accordingly, this requires that the game programmer collaborate
with most other team members at some point during development. Thus, the game
programmer also has to be the person who is the most sociable. This is consistent with
the finding from the content analysis and Figure 25 from the previous section. The game
programmer communicates most frequently and is involved in over 80% of the
conversations.

Personalities also play a role in determining the social environment. Personality
factors are the individual differences of team members that include attitudes, motivations
and knowledge. They determine how well team members get a long and enjoys working
together. Certain personality characteristics affect collaboration positively, such as
openness to learning, negotiation skills, willingness to help others and a sense of humor.

These positive personality characteristics facilitate the social interactions. The
result is a social space that is friendly and committed to learning and self-improvement.
Humor was perhaps the most noticeable characteristic exhibited by all team members. It
plays a role in facilitating communication and keeping the mood upbeat. Humor has the
ability to reduce tension and builds friendship (106). In this friendly environment, team members are willing to admitting fault and help one another (4, 394). As a result, the team is effective at identifying problems and improving both the game product and the development process. They adapt well and exhibit features of a self-organizing system. This is a team that does not have to be monitored or given strict policies in order to be productive.

*Team Mental Model.* The team mental model refers to the shared knowledge between the members and also the knowledge individuals have of each other. The latter refers to *team situational awareness.* It is essential for collaboration. Each team member needs to be aware of what others are doing. Without situational awareness, it would be very difficult to plan and perform one's own work since there is a requirement to coordinate individual contributions.

In addition to awareness, there is also the issue of *shared understanding.* The shared understanding facilitates communication by providing a common background of experience and knowledge. It allows team members to work together without being explicit about everything. When there is a shared understanding, the team works more productively because they do not have to spend time explaining basic details or justifications. *Team mental models* are formed through communication and experience. The team we observed has a particularly healthy *team mental model* because they communicate so frequently, especially about their work.
Organizational goals refer to the direction and purpose dictated by the organization and its culture. It motivates the collaborative effort by giving the reason to work or the mission. In this case, it is to make a serious game product. The organization also determines the software development processes by prescribing to certain development models. The organization establishes procedures, work flow and provides the software and equipment. All these affect collaboration in some way. Organizational goals also relate to the leadership and management. Collaboration is much easier when it is supported by managers and is a part of normal operations.

Act of Collaborating

The act of collaboration is defined as the sharing of information between roles to gain knowledge about or modify the game product. Within that definition are three themes: roles, information sharing and game knowledge.

Individual Roles. Teams with high role diversity benefit more from collaboration. High role diversity is a result of having many specialists. Team members are experts in their domain, but could not perform the work of team members. Since the product requires the unique contribution from each member, the game product can never be completed without a full collaborative effort from all team members. Two other themes relating to roles is the role respect and role overlap.

The team members ensure roles are respected. Each member has a role and contributes something different to the game development project. When someone oversteps their role boundaries, it is noticed (494, 648, 1090). In this example, the
Usability Expert, in half-jest, is taking notice of Game Programmer's extra effort to ensure production stays on track:

    GP: Oh yeah, this [game component], what's your status on this [game component]? Did you do it or what? What are you working on?
    GUI: Not yet.

    UE: Anyone get the feeling that [GP] wants to be our manager?

Even with the high role differentiation, there are areas where the skill sets of the members overlap. Where the skills overlap, it can become unclear who should be doing the work. We call this role overlap (2303). This is particularly true between the Artist, GUI Designer and Game Programmer who are all knowledgeable in Flash animation. There is also role overlap of usability skills between the Game Programmer, GUI Designer and Usability Expert (1233, 2347). In this example, the game requires 3D fonts and the Game Programmer and GUI Designer are negotiating whose tasks it should be:

    GP: Well is it a font [or image]?
    GUI: Just arial [font], but I 3D'd it.
    GP: You 3D'd it? How did you 3D it? Is this something I can do?
    GUI: Not at all
    GP: You put a grey [shadow] behind a white [letter].
    GUI: No actually I didn't, but you could do that

Sharing Information. Moving a piece of information from one person to another is the most basic result of the collaborative act. This is just another way of saying
communication is essential for collaboration to occur. Broadly speaking, there are three types of face to face communication patterns that occur in this environment. There are private conversations, public conversations, and broadcasts.

Private conversations are meant as conversations for only two people. To have private conversations in the open room, team members simply move closer to one another. Often, this is done for the purpose of completing work together. We refer to this as working-in-pairs. Troubleshooting issues is easier when working-in-pairs, as seen in this example:

*GP:* Right ok so I was just fixing the ground here

*LA:* Oh that's my fault

*GP:* You know I think I did something wrong. Was the office ground always like that?

*LA:* What do you [see]?

*GP:* Did I screw something up? I made a mistake?

*LA:* I don't see the ground

*GP:* That is the ground right

*LA:* What was?

*GP:* This? [But] that's just grey

*LA:* That's just grey, ha ha, I see [what's wrong]

*GP:* (realizing the issue) That's fine

*LA:* That is what we are looking for

*Working-in-pairs* saves time and frustration from troubleshooting because two
people can pool their effort to solve a problem. It also eliminates the problems from working on two differently configured computers. Computers that are configured differently display the game product differently. This is especially a problem for color display. Having different configurations makes it harder to pinpoint errors, so working-in-pairs is a common way of collaborating.

Public conversations are conversations that involve more than two people. Since everyone is in the same physical space, even the private conversations can be heard. Occasionally, if a third person finds the private conversation interesting, they will join in, turning it into a group or public conversation. In this next example, the GUI Designer is troubleshooting privately with the Game Programmer, but the Usability Expert soon joins in to help. The example also illustrates the difficulty of troubleshooting with separate computers:

GUI: I have it open here; [game2]

GP: Alright I'm going [to check it] online right now

GUI: Just refresh [your browser]

GP: It's [correct], so why are people telling me the one online is wrong?

UE: Have they cleared their cache? Want me to check? What are we looking for?

GP: Maybe

GUI: I guess it is online then

GP: You guess?
GUI: Well I know that's the correct one up there.

UE: Is it something to do with credits?

GP: No it's the logo.

GUI: Maybe their computer is the wrong color.

GP: Is it? Go to that and this. It's the same color.

UE: Oh yeah now I see.

GP: Maybe you put dark outlines around it, that's why it looks different.

UE: Well maybe we should get rid of the dark outline then.

GUI: Can't, cause then you can't notice so much the [letters]

UE: So like is this how it is on your computer [GP]?

GP: Yeah? [UE] come here, look at this. This is the right blue for sure.

UE: Yeah it is.

Lastly, there are broadcast communications, which are open-ended conversations that do not require a response. Their purpose is to simply provide information to the group as a whole. They can be important messages such as the Program Manager announcing an development milestones or as benign as someone speaking to themselves as they work aloud. However, in both cases, information is transmitted and even in a small way, learning occurs.

Aside from face-to-face communication, another way that information is shared is via external artefacts. Specifically, we are referring to software and whiteboards. The software for communicating includes email, bug tracking software and file management
systems. Whiteboards are also placed throughout the room for the team for team
members to use. The whiteboards tend to contain notes and ideas that are sketched in
rough form. These artefacts support asynchronous communication and allow the
collaborative process to be documented. For these reasons, external artefacts provide a
valuable supplement to face-to-face communication.

Game Concept. The game concept is how the team thinks about the game and is
illustrated in Figure 28 as the game implementation hierarchy. The hierarchy shows
different abstraction levels of the game product. The team’s conception of the game
influences how they organize their work and contributions to the game product. It
determines their roles, responsibilities and is the common ground for discussing game
issues. Issues are normally discussed using these general concepts.

![Game Abstraction Hierarchy](image)

*Figure 29. Game Abstraction Hierarchy*

The game hierarchy has three layers. The highest layer of abstraction consists of
ludology and pedagogy. These two terms capture the high level goals of the game; the
game must be a playful experience and through play, the user must achieve specific
learning objectives. The Studio Manager is primarily responsible for the implementation
layer, but relies on advice and suggestions from the rest of the team.

The second layer of abstraction includes *interactions, scenarios, code structure*
and *mis-en-scene*. *Interactions* refer to how and which objects in the game are manipulated by the user (1490, 2461). *Scenarios* are the stories and setting of the game (643, 973). The *mis-en-scene* is the visual aspect of the game users are exposed to (2570). The *code structure* refers to the organization of the lowest level abstractions (1400).

At the lowest layer in the implementation hierarchy, we have the *XML, code, text, fonts, interface widgets, art and sound*. The *XML* is the file that contains the game data and *text* (691). *Code* is the collection of scripts that allow the game to respond to user actions. *Text* refers the actual words and phrases that are displayed in the game (622, 1066, 1534). The art includes all graphical elements. The *GUI* (interface) is the collection of objects that the user interacts with to navigate in the game world (575, 1031). There is also *sound*, which is added to the game to provide additional feedback and enhance the playing experience (506).

When working on their own tasks, the individual will spent most of their time working on components at the lowest level of abstraction. For example, the Game Programming will focus on building the game world by generating the right *code*, the Artist will draw *graphics* most of his time and the GUI Designer is usually creating *GUI* widgets. However, when team members collaborate, they communicate using concepts from the second layer of abstraction. This second layer of abstraction is important because it means team members can communicate without needing to know about the low-level implementation details. The team’s common understanding exists at more abstract level. In this example, three members are engaged in a discussion of the *mis-en-
scene. Each understands it differently in terms of lower level abstractions. The Game Programmer works on the mis-en-scene in terms of code and xml. The GUI Designer works on it in terms of GUI components. The Lead Artist's concern is with graphics. While they all have different low level concerns about the mis-en-scene, it does not affect their shared understanding of the problem as long as they discuss issues at a higher abstraction layer:

GP: The panel can't be on at all times without distracting people. Like if that's up, you can't play the game

GUI: Oh you mean the panel, if it's out

GP: There's a huge problem with that thing right now. You click on the [game object] and it takes you there, but you can't see the [game object anymore] because the panel is up.

LA: That is a big problem.

GP: Do you understand?

GUI: Can it move? Like not [game object], but can the [panel] shift over?

Creative Process

The game product is development incrementally. The game concept starts vague, but becomes increasingly complex and refined after each iteration cycle. Through iterative refinement, the game is transformed and built from small pieces and these pieces are integrated opportunistically (537, 584, 2351). That is to say, the components are integrated at the earliest convenience. The sooner components are placed in the game, the
sooner they can be evaluated and refined. This team can typically develop a serious game within two months. Thus, they are on an accelerated development cycle and employ development processes that allow for rapid design, development and integration. The game product is also under continuous evaluation throughout development via quality assurance testing (2119) and usability testing (1297, 1769, 2543). As a result of this rapid game development, there are sometimes unnoticed changes and team members can run into the problem of cascading dependencies. *Cascading dependencies* refer to having to make several changes in many places to achieve one effect. This is normally due to functions and objects are coupled (2481). In this example, the GUI Designer is surprised by additional work because of *cascading dependencies*:

* GUI: *You can do the [change] right? Just scooch the [object] down, 6 pixels, 6 notches, it looks fine still*

* GP: *And your telling me I can do what? Like that?*

* GUI: *No no This. Scooch it down 6 notches or 5, Whatever. What's that at the end? (notices a graphic out of place in the mis-en-scene)*

* GP: *Oh a number. Oh cause all the bars Now See See See. Everything. Ok we'll do it like that though (fixes once instance)*

* GUI: Cool*

* GP: *Yeah all those bars. Ripple effect man*

* GUI: *What?*

* GP: *When you move one thing, you got like 9 things to move*
The team also employs two tactics to support rapid iteration and evaluation cycles. One tactic is to give priority to the visible result. The game has-to-look-good. The value placed on the visible results is summarized by this quote from the Game Programmer when trying to place a graphic in the game; “I don't know anything about the scaling or the size. This is just what it looks like in the game” (2698). Modifications to the game product are evaluated by how it affects the visual end-product with less emphasis on efficiency of the implementation code (56, 349, 992, 2235, 2266).

A related tactic is do-it-and-see. This refers to the emphasis on modifying the game product and testing. A corollary of this tactic is the de-emphasis on documentation and design plans. The tactic pushes speed and early evaluation. Instead of waiting for all components to be completed before working on integration, the Game Programmer will integrate what is already available and use placeholder objects until the rest of the components arrive (1309). Generally, this means there is always a working version of the game product to test, even if it is in rough form. In this example, the Game Programmer and Lead Artist are discussing an early iteration of the game. The programmer uses the feedback to let him know he is on track:

GUI: I'm just wondering where these things go. Once I do the initial placements of these [other things].

LA: Yeah, sounds good
GUI: I'll look at them cause it's all like code and then you can make some suggestions on..

LA: Sounds good

GUI: On possible changes. Actually I should have the first thing here. Do you want to see this? The first screen?

LA: Oh yeah, I didn't realize it would be that fast, just give me a [second] here

GUI: I could definitely move some of these a bit. Yeah I think the [object] looks good, because instead of having them constantly come out, I put breaks here.

ST: Yeah yeah, nice.

GUI: Ah yeah so, I don't know about name placements if they should go somewhere.

LA: Umm for [that title] would it suck to put the title above the table?

GUI: [I'll] see if I have that much space.

LA: Well you know, it's a small illustration. I think it's pretty good.

Grounded Theory Discussion

This section gives our interpretation of the major factors that contributing positively to the collaborative process. There are role respect, iterative development and shared vision of the game product. As a result of these factors, we found there to be a collaborative spirit among team members. This is supported by a technical infrastructure
and informal attitude towards work that help with the free flow of information.

The first factor is respect for other’s roles and understanding how one’s own role contributes to the product. In the team we observed, there was a clear understanding of how each member contributes to the game product. Accordingly then, each focuses on ensuring their own contributions are of the highest quality. They do not waste effort questioning or criticizing the quality of other people’s work. Also, they do not degrade the work of others. Team members understand how their roles are complementary in the multi-disciplinary environment and thus concentrate on their own responsibilities. The team values collaboration because they understand each of them alone could not create the game product. They respect each other because they recognize the effort and unique contributions others bring. The team members also show interest in learning about other member’s work. There is an effort to learn about how other people do their work. They are curious and eager to acquire knowledge beyond what is required.

The second factor is the iterative development process because it demands frequent collaboration. The game product goes through changes almost daily. New artwork and graphical interface changes are added frequently. This requires collaboration between artists, the interface designer and game programmer. The game product is evaluated early in the development stages and then throughout all iterations until the game is released. Thus the Usability Expert is frequently communicating usability results with the rest of the team. Due to the short iterations, there is a high level of involvement. Team members are usually working on something that will go into the game product
relatively soon. Therefore, they need to coordinate with each other to make sure the deliverables can be completed on schedule. They also work together to integrate the recommendations or game components. The iterative process itself is therefore a main driver of the collaborative process because it demands for frequent person to person interaction.

The third factor is a *common vision*. It is permeated throughout the team. There is a *common understanding* about what the game product should be. Thus, team members may disagree about minor details regarding the implementation, but they all agree to the same overall purpose of the product. They have an understanding that the product should be fun, educational and easy to play. Being in agreement about the higher level abstract goals allows them to handle disputes and move forward with changes more easily. They know even though all their ideas are not implemented, the game product will still end up consistent with their vision because the intentions and motivations are the same from all members of the team. There are no ulterior or conflicting motives that team members need to worry about. For the most part, they share similar views on usability, graphical style, good gameplay and teaching methods within the game.

The respect, high level of individual involvement in development and common understanding of the game product gives this team a very strong *collaborative spirit*. By spirit, we mean they are motivated to and enjoy collaborating. Furthermore, this spirit is empowered by their environment, which offers an advanced technical infrastructure and a collegial atmosphere. This is a social environment that is friendly. All opinions are valued
equally and they share a similar level of responsibility in terms of contributions to the
game product. There is no sign of seniority or hierarchical structure. Several things
contribute to this collegial mood. The team are all roughly the same age and have the
same experience in game development. It is also a newer company so none of them have
been working at the company significantly longer than any other. Team members also get
along well in and outside the workplace. The relaxed social environment allows them to
speak freely and joke. They are allowed to be honest as well as have fun. They do not
have to be reserved or censor themselves for fear of any backlash. This makes it easy to
work together and express ideas because the team is not afraid of being judged or
criticized.

The technical infrastructure includes the computer network and software. Team
members have access to new versions of software and hardware for game development.
The work is not impeded by out-dated software or tools. Another feature of the technical
infrastructure is that it is not heavily regulated. The team is allowed to try new software
on their computer and use the network freely. They are not impeded to try new things by
any strict oversight procedures.

Finally, the informal nature of the technical and social environments is ideal for
fostering ideas and collaboration. The environment is relaxed and non-judgmental. Team
members are encouraged to have fun at work. At the same time, the team is aware of the
seriousness of the work. They share a vision for the game product, respect each other’s
roles and work together as much as required. Interestingly, the playful and business-like
aspects of this environment may mirror itself in the playful and serious sides of the game product.

We conclude the Grounded Theory by claiming that successful collaboration requires a team that respects each other's contributions, communicates frequently and shares a similar conceptual model of the product and goals. The iterative development process also encourages collaboration by forcing developers to constantly work together to refine and evaluate the product. The organization's role is to foster a collaborative spirit by protecting cultural norms and providing the physical tools that support collaboration.
Main Findings and Discussion

Our original research questions were:

- How does the environment affect collaboration?
- How is information shared?

Our findings that answer these questions are:

- The factors affecting collaboration are role respect, communication opportunities, common knowledge, the iterative development process and collaborative software tools.
- Information is communicated interdependently through face-to-face and electronic channels.

In addition, we found tools that support team situational awareness to be lacking. We explain each of these points in the following subsections. Then we provide general recommendations for supporting collaboration based on the findings.

Factors Mediating the Collaborative Process

Based on the conversation analysis and Grounded Theory, we identified five factors mediating the collaborative process. They are role respect, opportunity to communicate, common knowledge, iterative development process and collaborative software tools. We saw a high level of respect for each other's contributions and expertise. This supports Pietroburgo and Bush's (2008) point about respect in collaboration. They state "participants must accept each other as a representative of the same interests, regardless of [professional] membership". In the team we studied, the
team members respected each other's different approaches and contributions for solving the problems. Their respect for diverse viewpoints was evident by their interest in learning about each other's work, having frequent public conversations and being willing to share ideas.

Despite their different roles and disciplines, they were able to communicate well with one another because they share common knowledge and have the same vision for the final product. Good communication was also identified by Rossen, Bartlett & Herrick (2008) as important for successful collaboration. Each of the team members have a large breadth of knowledge and their skills overlap significantly. They share task-specific knowledge (Muchinsky, 2003) with regard to creating digital art, programming and understanding usability heuristics. This allows them to understand most of the issues that come up during game development and they are able to help each other troubleshoot problems even if it is outside their specific work domain. Another aspect that shows their shared knowledge is that they have the same concept in terms of vision for serious game products. They may disagree on implementation details or how to solve problems at the technical level, but at the design level, they agree with what makes a game fun, educational and usable. This is important since common purpose is also an essential element in collaboration (Pietroburgo & Bush, 2008).

An often overlooked aspect of physical environments is that it provides co-location. Everyone is in the same room and within sight. There are virtually no barriers to face-to-face communication. Barrier-free communication was also identified as important
in Andriopoulos's (2001) review of collaboration and creativity in organizations. As a result of barrier-free communication, we saw frequent ad hoc meetings and group conversations. The iterative development process also encourages frequent face-to-face communication. By virtue of the rapid development process and short design-build-evaluate-refine cycles, the team often needs to get together to discuss planning for the next iteration cycle. Finally, the collaborative tools also support communication by providing another medium for them to communicate ideas. The collaborative tools are particularly effective for asynchronous communication and formalizing notes.

Interdependency of Face-to-face and Electronic Communication

The second finding we want to highlight is the interdependency of face-to-face and electronic communication. Both are used extensively to share information and collaborate. There are three angles to describe this interdependency. In all cases, they have complementary functions. One angle is to view face-to-face and electronic communication as carrying exclusive information. We found that face-to-face communication contains information about daily planning and implementation details whereas the electronic channel, e.g. bug tracker, would have information about who is responsible for working on a problem. The electronic information carries different types of information. It would not contain information about when the problem will be solved or how to solve it. Those implementation details, about scheduling and how to fix the problem, are left to face-to-face communication. Face-to-face communication seemed to be preferred for explaining complex issues and showing how to do something.
Another perspective of the interdependency is viewing face-to-face communication as an envelope or addendum to the electronic message. This means that the core message is contained in the electronic channel, while the superfluous or temporal information is delivered face-to-face. An example of this is inquiring about emails using face-to-face communication. Rather than discussing the core issue itself, which is addressed in the email, the face-to-face communication is constrained to temporal details, such as asking if an email was received or sent. We also noticed this with the bug tracker; the bug tracker contains information about the actual issues and assignments, but face-to-face communication surrounding the bug tracker was for prompting others to check it. Face-to-face communication was used to direct attention to the message, whereas the electronic communication delivered the core message. Another aspect of the face-to-face communication is it contains more humor content. Thus, it can lighten the tone of any message.

Lastly, the electronic communication channel is also sometimes a recapitulation of the face-to-face conversations. As an example, after a group discussion, one of the first tasks is to record the agreements and conclusions into an electronic file. By recording the outcome of the conversations electronically, it becomes formal and more binding. It also serves as a reference and reminder about past discussions. Thus, electronic records are created to summarize face-to-face communications.

The team uses of both types of communication for complementary reasons. There are times when one is more appropriate than the other. There are other times when both
should be used. To maximize information sharing, thus support collaboration, the environment should allow for both face-to-face and electronic communication. We also believe that providing additional channels for communication has a positive effect on the trust and relations between members (Lipnack & Stamps, 2000).

*Lack of Technical Support for Team Situational Awareness*

The importance of team situational awareness emerged from the Distributed Cognition of Teamwork analysis. It is knowledge of what other's are doing and consequently, being able to predict when their work will be delivered so that you can plan your own. In a broader context, team situational awareness also refers to how much the team members know about each other; their strengths, weaknesses and tendencies (Muchinsky, 2003). Muchinsky (2003) also notes that knowledge of teammates helps the team “compensate for one another, predict each other's actions, and allocate resources according to member expertise”. This is especially important in multi-disciplinary teams where the expertise are more diverse and specialized. Currently, the team situational awareness comes from face-to-face communication and other physical cues. There is no software or external artefacts designed explicitly to support team situational awareness.

The bug tracking system or e-mail are potentially useful platforms to develop a team situational awareness application. One solution is to provide a tool for planning work hour-by-hour. The purpose of the tool would be to make it more transparent what other people are doing. Currently, team members can only know by asking one another or if someone verbally broadcasts their work progress. The tool would encourage more
broadcasting of work progress, but using the computer. The program would also have to be easily visible and updatable within each person's work environment.

**Recommendations for Building Collaborative Teams**

Our recommendations for building or improving collaboration in teams are derived specifically from our findings. They are summarized in Table 27. For a comprehensive list of recommendations for fostering collaboration, refer to Andriopoulos (2001), Martin-Rodriquez, Beaulieu, D'Amour & Ferrada-Videla (2005) and Nemiro (2007).

Table 27.

**Recommendations for Collaborative Teams**

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide opportunities for cross-training</td>
<td>When team members have a better understanding of other disciplines, they become more sympathetic to the problems faced by other people. This is likely to increase the respect they have for one another. Cross-training can occur informally by having team members mentor each other or formally, through training courses.</td>
</tr>
<tr>
<td>Disseminate information about the</td>
<td>Provide opportunity for team members to learn about what everyone else is doing. Encourage group participation.</td>
</tr>
</tbody>
</table>
group conversations and meetings so that there is a greater awareness about what others are doing. This includes information about general responsibilities, but also day-to-day activities.

Create shared physical work space Co-located teams have the advantage of face-to-face communication. Face-to-face communication complements electronic communication because it is immediate and sometimes better for explaining complex ideas. Face-to-face communication is also easier for building trust and friendships. Misunderstandings are less likely to occur or are easier to resolve face-to-face. Part of working in a shared physical environment also has other social advantages, such as being greeted in the morning and after work. We also believe spending time together on breaks and lunches is another advantage that is underestimated for team building.

Strengths and Limitations of Methodology

In this section we discuss the strengths and limitations of the theoretical framework and the results of our findings. The strength of Distributed Cognition is its
focus on information use in a broad context. This perspective permits the researcher to look at both how information is used by people and how it is used by software. Thus, it is ideal for studying human-computer interaction. This is in contrast to psychological theories that explain information processing as merely a brain activity or computer science research that focuses on improving information processing in computers. Distributed Cognition fits within the gap between psychological and computer science research. It studies a phenomenon of interest to both sides, but aims to understand and shape it in such a way that both human cognition and external information processing are harmonized. Despite its strong conceptual footing, there are very few methods for applying the framework. More work needs to be done to develop a rigorous methodology for data gathering and analysis. Currently, studies of Distributed Cognition have been carried out mainly through ethnographies and other qualitative studies. This is partially due to the theoretical framework itself that stresses emergent properties within a socio-technical context. This makes it hard to determine variables ahead of time without in-situ observations. However, with advancements in research methods using simulated environments and better tools for qualitative analysis, it will become possible to test more precisely how external artefacts affect and support human cognition. We hope that some of these new techniques can be adapted for studying Distributed Cognition.

As of yet, the Distributed Cognition may not have much predictive value as a theory, but we believe it still has practical value when applied to the study of systems. Since it focuses on information use, its findings are easily amenable to the design of
artefacts, particularly software. Information use is also important in terms of human communication. It can be adapted for studying information transfer and loss. Each of the models are also useful for identifying issues in the environment affecting information use. With the information flow models, we can identify where errors in miscommunication and decision-making are more likely to occur. The physical model can be used to identify how the environment affects an individual’s ability to sense and operate with the artefacts and people around them. Lastly, the artefact models addresses the design of artefacts in terms of their ability to help people hold and process information. Ultimately, the goal is to design environments that support cognitive tasks, thus contributing to the ability of the system to process information. Information processing is an integral part of all human activity, including collaboration. However, we also know that information processing is not all there is to collaboration.

The main limitation of the Distributed Cognition framework is it diminishes the role of personal communication and other social aspects. It lacks tools to capture the socio-psychological issues. We believe human communication is mediated by many personal traits, such as humor, trust, respect and friendship. These aspects are not explicit in the framework. Distributed Cognition analysis treats interpersonal communication the same as communication between artefacts and ignores the richness of human expression.
Conclusion

This study looked at collaboration and communication among a serious game development team. The objective was to gain an understanding of the team environment by observing the work in its natural setting and asking questions in context. We also wanted a broad perspective that would consider social and technical factors contributing to team performance. Therefore, we performed an ethnography using socio-technical systems and the Distributed Cognition framework to guide our observations and analysis. Our findings have been presented using the DiCoT method, content analysis and as a model based on Grounded Theory analysis. We acknowledge the limited generalizability of our results given the specific characteristics and context of team environment we observed. The trade-off is that we were able to provide concrete design recommendations that can be applied directly to this environment. In contrast to other studies of teams (Kang, Yang, & Rowle, 2006) that start with pre-defined factors and measures, we let the issues emerge during the study. Despite the different approach, many of the salient factors are consistent with the related literature. Based on our analysis, we believe these issues are the most important within this environment with regard to teamwork: situational awareness, role respect, co-location and open communication.

Situational awareness is each team member's awareness of the changes in their environment. This includes knowledge of what others are doing and knowledge of changes to the game product. This directly contributes to the team's ability to co-ordinate tasks. Role respect is important given the multi-disciplinary nature of the team. Team
members must recognize and accept the different responsibilities and contributions of other members. Working in the same physical space allows team members to use face-to-face communication and other non-verbal cues to keep up to date at work. There are also social benefits that are gained from physical proximity. Finally, open communication refers to the willingness and quality of communication between team members. The social environment was collegial and friendly, where ideas can be expressed without fear of criticism. Team members were encouraged to collaborate, and when they did, it was constructive.

The environment is also important for supporting collaboration by providing the means for teamwork. The team collaborates face-to-face, electronically and through physical artefacts (e.g. whiteboards). They are provided with a shared virtual and physical space to work in. The virtual and physical spaces are used interdependently and allow additional means to collaborate.

Collaboration is also motivated by needs of the product and development process. The game product requires contributions from multiple disciplines. The iterative process requires frequent group discussions to evaluate the current build and discuss how the game can be refined. Collaboration is a part of everyday work.

This study was an exploratory look at collaboration in serious game development. We have identified some factors and made claims about how they contribute to the collaborative process. Future work will need to be done to validate our claims. This can be done by re-visiting the site for a second study to focus specifically on the issues we
identified and discussing the current models with the participants to allow for their input. Other studies that can be done should look at the relation of face-to-face and electronic communication. We believe that different types of information are transmitted through each medium. A systematic study of the content of each channel may provide useful insight for developing (virtual and real) environments as well as other tools for effective collaboration.
References


Appendices

Appendix A. Ethics Package

Form A - Ethics Proposal

Distributed Cognition in Game Development

The proposed research seeks to investigate Distributed Cognition in a game development company. We are interested in analyzing and modeling the cognitive aspects of this socio-technical system. In particular, we will focus on the representations and transformations of information as they recorded, stored and modified through social interactions and through the use of artefacts.

A cognitive ethnography will be performed at a game company. The cognitive ethnography involves one or more researchers who take the role of participant-observer in the game development process. The researcher will join the development team and stay with them as they develop the game product. Data collection methods includes observation notes, interviews and digital recordings. The data will focus on the physical work space, interactions between people, use of artefacts and organizational culture.

Procedure

Get Consent. A game development company in Ontario will be asked to participate in this study. We will require the consent from the manager to allow a researcher to perform the ethnography in the company (Form A). Each employee who will be included in the data will also be asked to sign a consent form (Form B).

Preliminary Meeting and Memo. Once consent is obtained, the researcher will
have a meeting with key members of the development team. The key members will be identified in advanced based on their role in the development team and their willingness to participate. The preliminary meeting will involve a brief introduction to Distributed Cognition and socio-technical systems analysis. The meeting will be the opportunity for the researcher to become familiar with the key members and the work domain. Terms of the observation (where the researcher will be set up, what will be recorded, who will be interviewed) will also be negotiated. A memo will be sent to all of the company informing them when and how the ethnography will occur.

*Perform the Cognitive Ethnography.* When the preliminary meetings are complete and the terms of the observations are set, the researcher will begin the cognitive ethnography. The ethnography will last 1-3 weeks. The researcher will sit in the office where the work occurs. The ethnography mainly involved focused observations of the work being done. When appropriate, interviews will be conducted to gain better insight into the work. The role of the researcher will be both participant-observer and apprentice-observer. The researcher will not contribute to the design or development work, but will be present during discussions and meetings. The researcher will observe the work designers and developers as they work individually and as they work collaboratively. The researcher will also take an apprenticeship role with individual members, attempting to learn how and why their activity is performed. This will be done by sitting close to the employees as they work and conducting contextual interviews.

The researcher will conduct informal and unstructured interviews with the team
members. The purpose of the interviews will be to understand the work of each individual and also to gather their opinions and attitudes towards any aspect of the development process. The interviews may be one-to-one or may occur in groups. Data collection will be audio, video and hand-written notes of conversations, interviews, meetings and documentation. Documentation will include company guidelines, memos, drawings and notes. Of particular interest to the researcher will be to the design artefacts how these are created, encoded, transmitted, transformed and used throughout the development process.

Observations, interviews and analysis will be done iteratively. Several observations and interview sessions will be conducted at the start. Based on the initial data, descriptions and models of the system will be produced. Specific questions related to the models will be generated. Additional observations and interviews will then be conducted in a focused way to answer the new questions and complete the models.

*Debrief Meeting.* After the ethnography is complete, a debriefing meeting will take place to notify the company members of the ending of the ethnography. The members will be informed of ways to contact the researcher to obtain information about the analysis and results of the study. The members will also be given information about who to contact regarding ethical concerns or problems with the study (Form C).
Form A

Manager Consent Form: Cognitive Ethnography

Research Personnel

Minh Tran  
Principle Investigator  
Carleton University  
(613) 520 2600 Ext. 6628  
mqtran@connect.carleton.ca

Dr. Robert Biddle  
Faculty Sponsor  
Carleton University  
(613) 520 2600 Ext. 6317  
Robert_Biddle@carleton.ca

Purpose

The purpose of this study is to analyze and model the game development process.

Requirements

We will ask you to allow a researcher to become part of your development team, observe the social interactions (meetings and conversations) and use of artefacts (design and development with tools and software). The researcher will interview members throughout the study, observe various aspects of the game development process and take audio and video recordings.

Duration and Locale

The ethnography will last 1-2 weeks and take place at your company.

Potential Risk/Discomfort

There will be no psychological or physical risk.

Anonymity/Confidentiality

All data that is collected will be held completely confidential. The data will only
be made available to those people involved with this research. Data will be coded for identification purposes. We are happy to sign non-disclosure agreements if you desire.

*Right to Withdraw or Withhold Information*

You have the right to withdraw at any time, without any explanation as to the reason for withdrawing from the study. Any data related to you will be removed from the analysis. You may withhold any information and have the right to refrain from answering any question. All information can be reviewed and censored by any participant in the study. You will have an opportunity to review a copy of the thesis draft and remove any parts you do you wish to have published.

*Signatures*

I have read and understand the above terms and I understand the conditions of participation. My signature below indicates that I have read the above information and that I agree to allow my employees to take part in this study. I have detailed restrictions to be placed on the research below.

Restrictions on the Research:

________________________________________________________________________

________________________________________________________________________

Manager Name  Signature

________________________________________________________________________

Researcher Name  Signature
Form B

Employee Consent Form: Cognitive Ethnography

Research Personnel

Minh Tran Dr. Robert Biddle
Principle Investigator Faculty Sponsor
Carleton University Carleton University
(613) 520 2600 Ext. 6628 (613) 520 2600 Ext. 6317
mqtran@connect.carleton.ca Robert_Biddle@carleton.ca

Purpose

The purpose of this study is to analyze and model the game development process.

Requirements

We will ask you to allow a researcher to become part of your development team, observe the social interactions (meetings and conversations) and use of artefacts (design and development with tools and software). The researcher will interview members throughout the study, observe various aspects of the game development process and take audio and video recordings.

Duration and Locale

The ethnography will last 1-2 weeks and take place at your company.

Potential Risk/Discomfort

There will be no psychological or physical risk.

Anonymity/Confidentiality

All data that is collected will be held completely confidential. The data will only
be made available to those people involved with this research. Data will be coded for identification purposes. We are happy to sign non-disclosure agreements if you desire.

*Right to Withdraw or Withhold Information*

You have the right to withdraw at any time, without any explanation as to the reason for withdrawing from the study. Any data related to you will be removed from the analysis. You may withhold any information and have the right to refrain from answering any question. All information can be reviewed and censored by any employee of the company. You will have an opportunity to review a copy of the thesis draft and remove any parts you do you wish to have published.

*Signatures*

I have read and understand the above terms and I understand the conditions of participation. My signature below indicates that I have read the above information and agree take part in this study.

__________________________  _______________________
Employee Name               Signature

__________________________  _______________________
Researcher Name             Signature
The purpose of this study was to describe and model the game development process.

Thank you for participating in this study. Your time and effort are greatly appreciated!

If you have any further questions regarding this research, please contact:

Minh Tran                   Robert Biddle
Principle Investigators     Faculty Sponsor
Carleton University        Carleton University
(613) 520 2600 Ext. 6628    (613) 520 2600 Ext. 6317
mqtran@connect.carleton.ca  robert_biddle@carleton.ca

If you have concerns about the ethics of this research, please contact: The principle investigator, Minh Tran at the Human Oriented Technology Lab (HOTLab), Department of Psychology, Carleton University, (613) 520 2600 Ext. 6628, or

Dr. Anne Bowker                   Dr. Avi Parush
Chair, Department of Psychology  Chair Ethics Committee for
Carleton University             Psychological Research
(613) 520 2600                  Carleton University
anne_bowker@carleton.ca          (613) 520 2600 ext 2026
                                   avi_parush@carleton.ca
Appendix B. Examples of DiCoT models

Physical Layout Model (Furniss & Blandford, 2006).
Information Flow Model (Furniss & Blandford, 2006).
Artefact Model showing artefact use by different individuals (Furniss & Blandford, 2006).
“Wall” artefact with sticky notes to inform developers of progress made on software (left) and the artefact model depicting high level meanings represented in artefact (right) (Sharp, Robinson, Segal & Furniss, 2006).
Appendix C. Physical Layout Sketches

Company floorplan
Studio room sketch
Appendix D. Grounded Theory coding examples on paper (hand-written)

Source Material:
jo: right right ok so I was just fixing the ground here
st: oh that's my fault
jo: you know I think I did something wrong
st: I wouldn't be surprised
ah what a shock
jo: was the office ground always like that?
source person (poison)
reducing steps
st: what do you think?
jo: did I screw something up? have I made a mistake?
st: I don't see the ground
jo: that is the ground right
st: what was?
jo: this? that's just grey
st: that's just grey, haha I see
jo: that's fine
st: that is what we are looking for
jo: I put the floor on the ... yeah I kind of arrowed that up

Source Material:
jo: (whistling)
jo: what did you put in this week that makes it go way up in the screen. We'll never know
st: on the floor?
jo: yeah
st: there's just these rogue pixels that I can't see on the side there
jo: oh I think I got it
st: cool
st: uh I got something installed on my computer that isn't letting me use this driver
st: class? (mockingly)
st: oh wouldn't that be ironic
st: oh ah I got a word doc from a writer, it looks pretty normal for me. I did track changes to tools, anyways it looks pretty normal to me.
st: and when I load it in here it looks
st: normally this happens when there are extra spaces.
st: I just delete when it happens... there like an extra space here or something, but this time there aren't extra spaces so I don't know why it's happening like that... am I making sense to you?
jo: ok send it to me.
sh: ok I'll send you the rules too
st: ok good
jo: hey how come this gets an error when the escort shows up?
st: have no idea
jo: think I made a mistake
st: that's not very likely
jo: what did I do
jo: (probably my fault)
sh: would you look at that?
jo: makes sense... (makes changes to game) our little secret? hahaha

Version

game object

helping troubleshoot
jo: yep, pretty awesome
ch: that's the biggest one
jo: yeh, I think so, lemme double check that
ch: oh there's a design product engineer... that's not as long as production/ships supervisor
jo: using Arial font eh?
ch: yeah.
jo: oh ok... [sits to review latest iteration of game]
ch: looks good... and you changed the occupation and
jo: and satsey font as well, good job... and
ch: this is the era logo with the changed colors
jo: in it?
ch: dark blue yeah...
jo: is that changed?
ch: doesn't look like it, do you want to look at
jo: I don't even need to look at it, I just need to
ch: know where the new one is cause I just used
jo: whatever someone gave me...
ch: do you have any information on... [eh asking jo for opinion on something]
jo: no I haven't even had a chance to get on that, but I will sorry...
ch: I have a question for you [to someone]
[somthing about databases]
jo: for... based on 3rd core? 3rd core stuff ok
ch: [more talk about db]
jo: we have DBAs signed off [] the question is do
ch: we give them access to the database or just data
jo: requests... so I... have to understand the pros and
ch: cons...
jo: what's going on with this logo, that?
ch: sup.
jo: no that's the same one for sure chris
ch: 1832291
jo: it's the same color, that's the one on
ch: the one I have is dark
jo: sorry
to: then save it on same
jo: what's it called? just
st: that's right
jo: come here look at this
ch: I have it open here... 1832291
jo: alright I'm going online... right now
ch: just refresh... 1832291
jo: it's 1832291... so why are people telling me the
jo: online is wrong?
ch: [laughing]
jo: have they cleared their cache? want me to
ch: check what are we looking for?
jo: maybe
ch: I guess it is online then

[third party double check]
because this the first game that uses it... and this
is the first time it's online... OK, yes.
so that's why if it's online it's better cause if
you catch things earlier... cause
OK, yes. I will stick this in and...
so might be...

ke, John?

jo

ke so the issue with this lady's name tag being
behind the crane.

jo, yeah.

ke you marked it as resolved.

jo, it is resolved.

ke so is it up on sales?

jo, yeah it should be.

ke so...

jo, have you cleared your cache?

ke it was just done today wasn't it? I need to
clear my cache... ah, I hate doing this... ok.

jo, yeah reach to lvl 4.

ke well, yeah doesn't matter I have to get to lvl 4
at some point soon... and it's in both story 1 and
story 2 fixed?

jo, yeah well I mean it... it would appear the bug
would happen in story 1 if you unlock the crane
right?

ke, right... but it will be fixed in both places.

jo, yeah cause they use the same art, it was in
steph's art thing he was just... it was hard with
all the things earlier... cause the scene...

sometimes something off the in the distance is at a
higher distance and you don't notice until you
move...

ke, yeah I get it.

jo, and in the case the navy... like how
looked like he was above the table... but we didn't
catch it because you don't see it until he his
cause that makes him longer.

ke, bingo.

jo, good.

ke, yeah I see it now... I will close it out (closing
report or mantis).

ke my goodness, we only have 2 things in the
database...

ke, ok.

jo, ha ha

ke, what

jo, so much cleaning up you know...

ke, ah got it.

ke, so so what are your thoughts on... so
something said they are confused by who the smg is
by the story have [I you get any thought about
anything to do...

jo, does it not say smg under the person's head? oh
no, you know that's my fault.

ke, should it say smg?
The notes were written on cards. Then using the affinity diagram technique, we consolidated the notes and themes together. The codes were then linked based on relatedness or causality.
Appendix F. Grounded Theory text corpus sample

Full corpus can be provided upon request

1. Source Material:
2. jo right right ok so I was just
3. jo fixing the ground here
4. st oh that's my fault
5. jo you know I think I did something wrong
6. st I wouldn't be surprised
7. sh what a shock
8. jo was the office ground always like that?
9. st what do you got?
10. jo did I screw something up? have I made a mistake?
11. st I don't see the ground
12. jo that is the ground right
13. st what was?
14. jo this? that's just grey
15. st that's just grey, haha I see
16. jo that's fine
17. st that is what we are looking for
18. jo I put the floor on the ... yeah I kind of
19. screwed that up
20.
21. Source Material:
22. jo (whistling)
23. jo what did you put in this steph that makes it go
24. way up in the screen. We'll never know eh
25. st on the floor?
26. jo yeah
27. st there's just some rogue pixels that I can't see
28. on the side there
29. jo oh I think I got it
30. st cool
31. st uh I got something installed on my computer that
32. isn't letting me use this driver
33. sc flash? (mockingly)
34. st oh wouldn't that be ironic
35. sh ok [engineer], i got a word doc from a writer,
36. it looks pretty normal for me.. I did track changes
37. to tools.. anyways, it looks pretty normal to me..
38. sh and then when I load it in here it looks..
39. normally this happens when there are extra spaces..
40. I just delete when it happens.. there like an extra
41. space here or osmething.. but this time there
42. aren't extra spaces so i don't know why it's
43. happening like that.. am I making sense to you?
44. sha ok send it to me.
45. sh ok i'll send you the rules too?
46. sha ok good
jo hey [st], how come [gamechar] gets an arm when
the escort shows up?
st I have no idea
jo I think I made a mistake
st that's not very likely
jo what did I do
st probably my fault
st oh would you look at that?
jo makes sense.... (makes changes to game) our
little secret? hahaha
tr (unintelligible)
sh ahh... ... i'm doing usability testing .. 4 out
of 5 have the first part finished
tr I think [writer]... (unintelligible)
sh and also on the top of my head.. I have not
played it, but I have watched in painful detail 4
people play it.. I don't know the answer it takes
to answer these.. and now I'm playing it, these are
really good questions.. oh and you know a good
person you should ask, [engineer], my pilot
ca I need a favor of you studio guys
st I think she is talking to you [jo]
ca I'm talking to all of you. I have this business
and balance quiz
ke oh a business and balance quiz!! (excited) I'm
not going to do it now, I'll do it later
ca no no, at any point when people have 5 minutes..
we are trying to get a questionnaire together for
bsi.. we need a test.. I need to make something not
to hard, not to difficult
ke can we do the quiz after we play business and
balance?
ca oh of course.. [writer] is actually going to do
it twice for me.. he's going to do it once.. then
he's going to play the game and do it again.. mark
on there that that's what you did.. only if you
have 5 minutes if you dont I totally understand
ch [sh] (seeking advice on quiz)
sh yeah
sh if [ch] and I were in the same class in
university, he'de be the guy always asking for my
notes, don't you think?
st I'de be right there with you
sh hehehe... one of our customers want a test to go
with business and balance. so She want to do a dry
run.. and you can't copy my notes
ch just right it down.. a .. b .. c .. c. c. c. c
st wait til you see the stuff we come up with

Source Material:
sh ha ha ha the original text was.. if I wanted to
work in a beaurocratic company I would have not
started my own business.. we are sucessful because
we are small and nimble. Maybe the governement can
pay a bunch of people to sit around and write
policy all day.. so that got changed to maybe some
"places" can pay a bunch of people, hehe.. they got
rid of the government
st yeah, that would fly like a lead baloon.. when
we get a bunch of civil servants playing our game..
jo womp womp wommmm
st would you consider changing careers, distil is
hiring
sh let me go to their website and check
([engineer] arrives)
sh did it work
sa no, there is something wrong with the document.
there is nothing wrong with application
sh oh yeah yeah
sa you define the column as a comment and you put
the real node in a comment
jo why would you do that sh?
sh shut up
(giggling) jo just a little more time
st sh don't you like [engineer]? you can't do that
to my friend
(laughing)
sa ok I want to show you this.. this is 2007
right.. where is the style..
sh I have no idea.. isn't this wonderful
microsoft.. they changed everything.. serouisly I
thought I had to buy microsoft for dummies book
because they totally changed everything.. so what
are you saying is the font is comment or something?
sa I can show you on my desktop.. cause this is, is
causing problems
st flash doesn't have a framerate, your monitor has
a framerate
jo flash doesn't have a framerate (sarcastic), what
are you talking about. you clearly don't know
anything about anything..
st you can use a smooch like we do
jo we have 25 years of web devloepment
(laughing)
jo figure that one.. we were using flash before the
internet (laughing)
jo alright, speed of game massively increased
jo still kind of slow.. I don't know why it's
slowing down so much, it's alright down
(unintelliglge).. I always wondered.. oh it could be
these things here (showing ch how game works).. see
that's how we want it to go, we want it to go like
that all the time right?
st it's slow.. i'm sure it will go faster for you
guys once we get it to work
(shamina comes back to desk)... sh oh random.. I
can't see it on my [program], but randomly inside
the table font changed to comment
jo crazy
Job Title: Lead Artist

Overall Responsibilities
- As part of a team focused environment, the Lead Artist will contribute to the creation and architectural design of innovative game products
- Working closely with other functional groups, Lead Artist participates in all phases of the development lifecycle and assists in the execution of successful product development/launch
- Assists in leading top quality art team for an exciting, ambitious and cutting-edge game products with huge potential and scope

Additional Responsibilities:
- Apply experience with an understanding of large project pipelines to create effective art schedules
- Set goals and objectives for junior team members and act as visionary, mentor and coach
- Interact with subordinates, peers and/or managers in other work groups / teams and/or vendors.
- Design and create art for products
- Lead other artists in the creation and implementation of art for assigned projects toward the achievement of defined outcomes, according to schedule
- Track art resources and the development of art assets
- Verify that art assets are completed on time and meet the requirements of DISTIL products
- Provide ongoing training and support to art team on utilized software tools.
- Establish, maintain and enhance relationships with internal and external creative resources
- Be a proactive and dynamic force in the project as a whole with a focused goal of ensuring the product is cutting edge in terms of graphical techniques and technology
- Determine the technical and artistic challenges that must be met to achieve the art vision and ultimately ensure the vision is fully adhered to and realized

Skills and Knowledge
- Ability to work in large groups, including strong interpersonal skills.
- Demonstrate ability to be self-motivated, multi-task, and meet deadlines.
- Demonstrate ability to work well with Design, Marketing, and Engineering staff to efficiently deliver the best products while meeting the goals and requirements of each area

Appendix G. Official Job Descriptions
• Ability to adapt quickly to a rapidly changing, fast-paced work environment.
• Five + years of industry experience.
• Sketching and drawing an asset
• A strong technical mind for game art and relevant technology.
• Strong analytical skills a plus.
• Strong proficiency in the use of multiple software packages such as Flash, Photoshop and 3d software packages
• Demonstrated leadership, management and organizational skills.

Job Title: Interface/Graphic Designer

In this forward thinking company, the Graphic Designer works within a creative teamconcepting, developing and delivering high quality game products to DISTIL customers.
Using story boards and other visual solutions, the Graphic Designer produces art and sound assets for game products and corporate communications. This is achieved by using a mix of creative skills, commercial awareness, imaginative flair and knowledge of the latest technologies and computer software.

Overall Responsibilities:
• Providing graphics design services
• Working closely with other functional groups to rapidly execute a successful product development/launch
• Participating in all phases of the development lifecycle as required
• Brainstorming with other functional groups on new game products and corporate communication ideas
• Interact with subordinates, peers and/or managers in other work groups/teams and/or vendors
• Establish, maintain and enhance relationships with internal and external creative resources
• Reviewing trends in art and using the best design practices
• Ensuring that the most current tools are in use
• Interact with usability group

Knowledge and Skills:
• 3+ years of industry experience
• College education in Graphics/Animation program
• Using a wide range of media, including photography and computer aided design;
• Producing accurate and high quality work;
• Contributing ideas and design artwork to the overall products and corporate communications
• Working to tight deadlines and understanding material costs and time limits, all of which can impact on design
• Competency in a Windows environment using Adobe Creative Suite: Illustrator, InDesign, Photoshop, Acrobat, as well as MS PowerPoint and Word
• Understanding of corporate identity and branding
• Creative problem solver from initial concept to final output
• Ability to design to an existing corporate image for print, website or company software
• Layout skills for multi-page reports, forms, booklets, onscreen presentations, newsletters, brochures and invitations and other corporate communications
• Able to design for print and on-line distribution

Job Title: Usability Expert/Editor/Writer

Key Responsibilities
• Assist in the design and development of a writing style guide to suit the game style.
• Work closely with Subject Matter Experts (SMEs) in translating content into accessible, engaging and enjoyable content that works in a game setting.
• Design and conduct usability tests, contextual inquiries, surveys and/or other research, and analyze data.
• Recruit participants with required characteristics for user research projects.
• Communicate usability design requirements to both technical and non-technical colleagues.

Specific skills and certifications
• Excellent analytical ability, especially with regard to observation of user behavior and data analysis.
• Strong skills in writing English.
• Thorough understanding of and respect for English grammar.
• A passion for revising work until the audience both understands it and enjoys it.
• Enthusiasm for working in a team and gathering input from people.
• Comfort in reading and understanding diverse documentary sources.
• A focus on details and getting the facts right.
• Flexibility in writing styles.
• An open mind in how to approach and resolve problems.
• Comfort with using Microsoft Office software or similar.
• Proven ability to plan and conduct usability tests, cognitive walkthroughs, heuristics evaluations, surveys, interviews, competitive analysis, card sorts, task and needs analysis, user profiling, and other usability methodologies.
Intermediate knowledge and practical application of laboratory & field study techniques for usability research, experimental design and statistics.

Familiarity with the Agile Development Process.

Preferred education & work experience

- Advanced degree in Human Factors, Psychology, Human Computer Interaction, or related HCI discipline is preferred.
- Bachelor degree in English, Psychology, Journalism, or HCI related discipline with 2+ years experience in designing, conducting and analyzing usability evaluations and user research.
- Experience working collaboratively with designers and developers as part of the creative design team.

Sample work

- Candidate should be able to show sample work, e.g., writing samples, usability evaluations.

Job Title: Game Programmer / Game Designer

Responsibilities:

- Plan, design, and develop Flash based serious games
- Prepare and maintain Game Concept and Game Design documents
- Lead a small team to create games to a tight deadline
- Some high-level scripting using our proprietary Flash-based game engine

Knowledge & Skills:

- Four-year college degree,
- Some experience developing in Flash
- Two years of leading and/or project managing teams of 3-5 people
- Strong portfolio that includes numerous examples of games that you designed or contributed to
- Excellent English written and verbal communication skills and ability to collaborate effectively with team mates.
- Ability to work closely with customers, understand their requirements and deliver accordingly.
- Motivated self-starter who likes working on very productive, fast-paced teams.
- Passion for games.

Computer programmers perform some or all of the following duties:

- Write, modify, integrate and test software code
• Maintain existing computer programs by making modifications as required
• Identify and communicate technical problems, processes and solutions
• Prepare reports, manuals and other documentation on the status, operation and maintenance of software
• Assist in the collection and documentation of user's requirements
• Assist in the development of logical and physical specifications
• Research and evaluate a variety of software products.

Interactive media developers perform some or all of the following duties:
• Program animation software to predefined specifications for interactive CDs, DVDs, video game cartridges and internet-based applications
• Program special effects software for film and video applications
• Write, modify, integrate and test software code for e-commerce and other internet applications
• Assist in the collection and documentation of user's requirements
• Assist in the development of logical and physical specifications
• Research and evaluate a variety of interactive media software products.
Appendix H. Social network graph and matrix

<table>
<thead>
<tr>
<th></th>
<th>GUI</th>
<th>GP</th>
<th>MAN</th>
<th>ART</th>
<th>UE</th>
<th>LAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI</td>
<td>0</td>
<td>287</td>
<td>15</td>
<td>41</td>
<td>67</td>
<td>36</td>
</tr>
<tr>
<td>GP</td>
<td>295</td>
<td>0</td>
<td>79</td>
<td>65</td>
<td>205</td>
<td>222</td>
</tr>
<tr>
<td>MAN</td>
<td>13</td>
<td>81</td>
<td>0</td>
<td>1</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>ART</td>
<td>46</td>
<td>54</td>
<td>1</td>
<td>0</td>
<td>33</td>
<td>42</td>
</tr>
<tr>
<td>UE</td>
<td>57</td>
<td>221</td>
<td>15</td>
<td>23</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>LAR</td>
<td>38</td>
<td>217</td>
<td>8</td>
<td>47</td>
<td>36</td>
<td>0</td>
</tr>
</tbody>
</table>
Appendix I. Humor and conversation charts

Session 1.1

Session 1.2

Session 2.1