An Ethnographic Study of Collaboration in a Game Development Team

Minh Q. Tran
Human-Oriented Technology Lab
Carleton University
mqtran@connect.carleton.ca

Robert Biddle
Human-Oriented Technology Lab
Carleton University
robert_biddle@carleton.ca

Abstract

This paper presents an ethnographically-informed study of the practices at a game company that develops serious games for business training. Games research focuses mainly on the game product, paying little attention to the design and development process. This study attempts to address this gap. We examined the day-to-day activity of a team responsible for designing and developing game content. Our data collection methods were informed by ethnography; they include field observations, contextual interviews and audio recordings. A thematic analysis was performed on transcripts of naturally occurring conversation within the team. The results were triangulated with field notes and interview data. The result is a description of collaborative activity within a game development team, and an interpretation of how the socio-technical environment supported the game development process. This study suggests innovative game design can be supported by creating a culture of collaboration, but innovation from teams is largely dependent on the quality of the interpersonal relationships.

Author Keywords

Collaboration; game development; ethnography.

Introduction

This paper presents an ethnographically-informed study of the development practices at a game company that develops serious games for business training.[1] The study began with the assumption that the development process, and more broadly the development environment, influences the outcome of the game product. Research in game studies tends to look at the frontend of gaming, either with critical analysis of the gameplay (narratology, ludology), or by studying the gamer's experience (playability, usability). This is a study of the backend of gaming: we examine the game development process. The quality of a game product is not determined solely by decisions about gameplay and user experience. The quality of a game product is also determined by the development process. The expertise and values individuals possess, as well as the culture and environment of the game company makes a difference on the outcome of the game product (Malaby, 2009). It is during the development process when design ideas are generated and implemented. It is also where the user experience is tested and refined. Understanding the development process reveals design constraints that cannot be known by looking at the game product alone. Furthermore, understanding the process can help us create better game development tools and game development processes to support innovation in game

design.

This paper describes the activity of one development team creating serious games. This is a study of how development tasks are carried out individually and as a team, in practice. We highlight factors in the environment that contribute to successful collaboration. Collaboration was examined by looking at how the team communicated in face-to-face situations while working on a problem. Our hope is this study can provide examples of how collaboration can be supported in other game development teams.

Background

This was a qualitative study exploring game development practices. Rather than generating hypotheses from the literature, we used the literature to inform us about issues that are prevalent in software development teams and relevant to our study. In this section, we discuss the tension between the development process and the design process, teams and collaboration in making serious games and present a theoretical framework for understanding socio-technical systems. Our research questions are identified at the end of this section.

Development versus design

Creating a software product is typically done in teams. Each team member may have their own unique perspective on software development, depending on their expertise and personal values (Finkelstein et al., 1992). Furthermore, these differences are not always easy to reconcile. We briefly explore two perspectives: software engineering as a development process and software engineering as a design process. The view of software engineering as a development process may produce a bias towards management and control of delivery timelines. Meanwhile, the view of software engineering as a design process emphasizes the need for freedom to experiment with ideas. This may result in tension between freedom and control during game development.

As a software development process, the primary goal is to get from high level descriptions of the product into executable source code components (Scacchi, 2001). Managing the development process includes planning and organizing staff, costs and time. To help with the planning and organizing process, organizations may prescribe an existing software development model (Scacchi, 2001). These models provide structure, rules and principles to follow during the development process to ensure software is delivered on time and to specification. Development models guide the development process, but place limits on the range of activities and interactions. Even a "lightweight" development model, such as Agile Programming, can restrict the types of activities in a development team (Sharp et al., 2008). Development models also influence the professional relationships between members in a team (Whitworth and Biddle, 2007). Thus, while software development models are useful for guiding the development process, they also introduce constraints.

A design process is less structured and less predictable than a development process. Design starts with a goal and then becomes an idea that can be implemented. The hallmark of the design process is that the outcome is not known until the end. It is through the design process that the ideas take shape. The design process is difficult to plan because the output of innovative ideas cannot be managed as easily as the output of source code. However, innovative design ideas can

be supported by adopting certain strategies, such as allowing more experimentation (March, 1991). It is also supported by providing knowledge-representation tools and opportunities to interact with other people (Ye, 2006). Designers need freedom to experiment and interact.

In a game development setting, the process of development and design is likely to occur simultaneously. Where does development begin? Where does design begin? How can the objectives of development and design be incorporated into a model of game development to meet business requirements, while still fostering innovation in game design?

Teamwork and collaboration

A separate issue from the tension between development and design is the issue of teamwork and collaboration. Game development is likely to involve teams. A team is group of people who are all responsible for completing a shared task (Muchinsky, 2003). The level of coordination and shared understanding among the team depends on the task. The maintenance of this coordinated effort and shared understanding will be referred to as collaboration.

Collaboration is difficult, especially in multi-disciplinary teams. Many factors have been identified that suggest why collaboration is successful. Those factors that the individual can control are called interpersonal factors. Critical interpersonal factors are trust between collaborators and the ability to communicate well (Rossen et al., 2008). Willingness to collaborate and mutual respect are also important (Martin-Rodriguez et al., 2005). Factors that the organization controls include providing the physical and human resources, as well as having the appropriate protocols and culture to support collaboration (Pietroburgo & Bush, 2008). Beyond this, there are socio-historical factors, which are harder to recognize. Socio-historical factors are evident through the values held by individuals, and can be traced to the socialization of the individual by their discipline (Martin-Rodriguez et al., 2005). They present barriers to collaboration in the form of biased assumptions (e.g. efficient program code might be regarded as more important than usability in software engineering), or they may be issues related to training (certain disciplines favor isolated working conditions).

The factors mentioned above are expected to influence how well multi-disciplinary teams collaborate. However, they deal with macro-level analysis of collaboration. This study is based on micro-level analysis: it seeks to explain how individuals collaborate on specific and discrete problems. A finer grain look at what is involved in collaboration is required. Two concepts that are useful for micro-level analysis of collaboration are grounding in communication and collaborative software.

Grounding is the process through which two people establish mutual understanding, or "common ground" (Clark, 1996). This is more than simply listening; it involves active reflection and assurance-seeking. In other words, both parties need to make sure the other understands what has been said. Without common ground, a conversation cannot move forward. How grounding is accomplished depends on the context and the communication channel. The context and communication channel can be evaluated using eight affordances: co-presence, visibility, audibility, cotemporality, simultaneity, sequentiality, reviewability and revisability (Clark and Brennan, 1991). The more affordances that are available, the easier it is for grounding in a conversation. Face-to-face communication offers the most affordances, but it lacks reviewability

and revisability. Electronic communication, however, is excellent for reviewability and revisability, but lacks the other affordances. Advanced features in computer-mediated communication are often motivated by the need to add more affordances in relation to grounding (Mühlpfordt and Wessner, 2005). Meanwhile, the easiest solution is using both face-to-face communication and electronic communication. The advantages of one channel supplements the weaknesses of the other. This is typical in high-tech work environments, such as game development studios. Common ground is established through the simultaneous use of multiple communication channels. Thus, the more communication channels that are accessible, the easier it will be to communicate and collaborate.

In addition to the use of computers as a communication channel, computers can also be used to collaborate. There are different software applications to help teams collaborate; some applications are made specifically for collaborating on software development. However, as with most technology, there is the issue of technology acceptance and appropriation. Regardless of how useful a product might be, it must fit within the existing workflow and computing environment (Huh et al., 2007). The effectiveness of collaborative software may depend on how well it can be adapted to an existing process. It is important to understand the whole work process when evaluating the effectiveness of collaborative software.

To provide a holistic understanding of collaboration, we need to understand how it occurs at the individual level (person-to-person), at the disciplinary level (role-to-role), and how it occurs in the context of a specific socio-technical environment. At the individual level we examine the use of communication and communication tools. How does common ground develop between individuals and among the group? At the disciplinary level, we examine practices and values that individuals bring to the team. How does the team use the materials each individual produces, and how do individuals reconcile their own priorities with that of the group? The socio-technical environment is also influential. What tools are there in the environment? How do personal relationships influence collaboration?

Serious games

This subsection reviews the characteristics of serious games. Serious games are a subset of computer games. We have already made the assumption that the development process determines the outcome of the product. We now assume that the product, in turn, influences the development process.

A formal definition of games is a "participatory activity, conducted in the context of a pretended reality, in which participants try to achieve at least one arbitrary non-trivial goal by acting in accordance with the rules" (Adams and Rollings, 2007). Computer games could be defined the same way, with the condition that computers synthesize the reality (game-world) and mediate the interaction. Following that definition, computers game developers need to focus on four components: the goal, the setting, the rule set, and interactivity. The goal gives the players' their tasks. It pulls (or pushes) them through the different scenarios in the game; the scenarios are the situations, challenges and environments that players experience. The rule set defines the behavior of the environment and determines the possible ways challenges are solved. Finally, interactivity determines how players are represented in the game and also how they interact with it. These are the basic components of a computer game. Being able to separate the game into components

suggests that work can be divided by having different people work on different components. How the responsibilities for components are negotiated is up to the team.

In addition to the components mentioned above, serious games include an educational or training component. A serious game, by definition, requires that the player learn something useful that can be applied outside the context of the game. Therefore, serious game developers need to worry about what to teach (learning objectives), and how to teach it (pedagogy). The requirements of having a learning objective and pedagogy present additional challenges in the development process. There are the practical challenges of collaborating with subject matter experts and educators. There is also the conceptual challenge of blending learning and play. The game must remain engaging and fun, without compromising on educational value (Dickey, 2005). How this is accomplished can vary from game to game (Gee, 2007). Not all games are successful at integrating these elements. The extent to which blending fun and education succeeds may depend on how well the team works together on integrating their separate components. Thus, it is worth asking how the team's ability to collaborate influences the quality of the game product.

Distributed Cognition: a perspective on socio-technical systems

Our objective was to create an understanding of the game development process by describing the developer's everyday activities and working environment. We use Distributed Cognition as our theoretical framework. Distributed Cognition is suited for describing socio-technical systems and processes within them. The advantage of using this framework is it provides the researcher with an existing vocabulary and models to describe systems and work. Furthermore, it encourages researchers to consider how some of the work is performed by tools in the environment (Perry, 2003; Hutchins, 1995). In this study, we looked at how developers use what 'is in their heads' along with what is in their environment to collaborate on solving problems and develop the game product. Our research questions were:

- How does the environment affect collaboration?
- How is information shared?
- How does the game product change as a result of the collaborative process?

Methodology

This was an ethnographically-informed study of software game development using a mixture of methods for data collection and analysis (Robinson et al., 2007). Observations and interviews were conducted in a game development studio over a period of two weeks. The data gathered include field notes, audio recordings of communication, contextual interviews, digital photographs and game documents. The principal data analysis technique was thematic analysis of the conversations and researcher's notes. This was supplemented with a content analysis of conversations and a descriptive analysis of the physical site and work flow. The results are presented as a story and a model of collaboration of the game development team. The model was generated using an inductive coding process, similar to the coding technique used in Grounded Theory (Glaser and Strauss, 1967). Affinity diagrams were used to organize the resulting themes (Beyer & Holtzblatt, 1998).

Distributed Cognition was a useful analytic framework for organizing our fieldnotes and mapping the interdependencies between processes, people and tools (Blandford and Furniss, 2005). During observations we focused on how these three components (processes, people and tools) in the environment affect collaboration and communication. Problem solving was the main collaborative activity undertaken by the team[2]. The data collection methods were designed to observe the internalization and externalization of information during collaborative work. Therefore, we did not focus on problem solving as being an individual task. Our focus was on looking at how problems are solved with the help of other team members and shared tools.

Site of observation

The observations were within a serious games development company, specializing in games for the Flash platform (Adobe, 2009). The company employed roughly 30 staff members at the time of the observations. We observed the day-to-day activity of the 'creative studio'. They are the team within the company responsible for the design and implementation of the final game product. This was a team of 6 members: 5 males and 1 female.

Access to the site and participants

The CEO of the company sent a letter with the research proposal. Terms and conditions were negotiated; they included building access, access to staff, and a non-disclosure agreement. Separate letters of consent were signed by each team member who agreed to participate in the study. For this study, all team members agreed to participate.

Observation, interviews and recordings

The team was observed doing their everyday tasks over a period of two weeks. Typically, the researcher was at the site from 10am to 4pm, from Monday to Friday. The observations were shortened on two days due to snow storms. One researcher sat in the same room with the team most of the day. The researcher accompanied the team during their lunch breaks and meetings. An audio recorder was left 'on' throughout the day to record naturally occurring conversation in the development studio. The recorder was placed in the center of an open plan room and was able to record all conversations between team members. The audio recorder was shut off during the lunch period as a courtesy, but observations continued. The transcript of the audio recording was the core data source for this study. The conservations are believed to be a reliable reflection of the collaborative activities because collaboration usually involves communication. The recording includes 30 hours of 'tape', including conversation between team members and conversation of team members with other staff in the company. Brief, opportunistic 10 minute interviews were conducted throughout the study to learn about the daily tasks each team member was responsible for. Fieldnotes were taken to record information about the physical space, work flow, and individual work habits. Notes about individual work habits focused on their use of tools and artifacts. Various company documents were collected for analysis, including software change logs, job descriptions, game prototypes and internal reports. Photographs of the workplace and screenshots of the game and software tools were also taken.

Analysis and results

The results are based on consolidation and synthesis of the textual analysis of the transcripts with the researcher's notes and artifacts from the observation site. The textual analysis included a

thematic analysis and content analysis. These were performed independently (in different stages of the analysis), but by the same researchers.

The thematic analysis was done by hand, using notecards, sticky tape and poster paper. The themes were derived from the data, but aided by referencing field notes and the literature (Braun and Clarke, 2006). Step one in this analysis was going through the transcript line by line, coding each instance of communication (also known as speech acts). Each speech act was annotated based on their content and purpose. Each annotation was written on a separate notecard. Step two involved putting the notecards into groups based on similarity of content. This resulted in stacks of cards. The stacks where laid out on large poster paper and re-arranged based on their relatedness or causal links (shown in Figure 1). Groups were formed on the poster. Areas were cordoned off (using marker pen) separating unrelated groups. These areas became our sub-themes. The sub-themes, along with our understanding from the field notes, interviews and document analysis were used as the starting point for writing about the collaborative process and finally generating a qualitative model of collaboration.

Figure 1: A8 notecards and A1 poster paper were used to organize the themes during analysis.

Content analysis was used in a separate stage of the analysis to provide additional insights about the communication. The content analysis revealed other patterns not found from the thematic analysis. To extract communication patterns through content analysis, we first coded the transcripts descriptively (without interpreting the meaning of the speech acts). We added meta-data to the speech acts with information about who spoke, to whom and about what subject. The meta-data was analyzed with a custom-made script in R. The script was customized to read the format of our transcript files. R (2009) is a software application and programming tool for statistical analysis that is free to use. The results of the content analysis were graphs and a network diagram visualizing communication patterns. This gave us a more objective view of the data compared to the thematic analysis.

The results of the thematic analysis and content analysis were consolidated with our field notes and interview notes to produce a rich description of collaboration and communication within the team. The description includes the team's environment, their work, and relationships to each other. Furthermore, we identified a few key factors in the environment that were seen to have the largest influence on collaboration.

Results

The team

The team had 6 members: a game programmer (male), a graphical user interface designer (male), a lead graphic artist (male), a graphic artist (male), a usability expert (female), and a manager (male). The manager had a dual role in the company. He was also the CEO. Therefore, he only spent half of his time in the development studio with the team. The manager's role was to act as a link between the team and rest of the organization. His duties were to keep the team informed of the company's short term and long term objectives (delivery deadlines, client requests, etc.). The usability expert was responsible for improving the gameplay experience by identifying issues with player-game interaction. Her duties were to conduct usability tests and report any issues to the team. The lead graphic artist and graphic artist were both responsible for creating the artwork and sound for the game. The lead artist was more senior ranking than the artist, but they shared similar tasks. The graphical user interface (GUI) designer was responsible for the interface components and scripts to control the interface. The game programmer was responsible for compiling the game scenarios and integrating the components from the other team members. He also had a role in supporting other team members; he had to guide others on how to create components to fit properly into the game. The team had been working together for 6 months prior to the start of the observations. They were all between the ages of 20-30, with the exception of the manager who was older.

The type of work and product

The work being done by the team during the observation period was testing and polishing of two games. Both games were for teaching business skills. Thus, the target audiences of the games were business professionals. Each game was developed in Flash (Adobe Flash, 2009) using standard, commercially available Flash development tools. A game takes roughly two to six months to complete. These games were at the end of their development cycle.

The main objective for the team was to have both games ready for delivery to the client without any bugs. Bugs are errors in any of the game components that negatively affect gameplay, which would be revealed through testing. Bugs are documented in an online bug-tracking tool (a special database designed for managing bug information). Bug information came from a variety of sources; these sources included clients, professional game testers, and the development team. When bug information is entered into the database, the team manager is notified. The team manager then forwards the information to the team member who he thought was most adept to resolve the bug issue. Sometimes, resolving the bug issue could be done faster or more easily by working together. These situations prompted collaboration. The following subsections describe the collaborative process and identify the main factors that influence it.

Collaboration in game development

Figure 2 shows our qualitative model of collaboration within this team. The factors in the model are explained in more detail throughout the results section. The *social atmosphere* produces *role differentiation*. Team members know what their duties are and respect the role boundaries. The *office space* facilitates *knowledge sharing* through an open plan layout. High levels of knowledge sharing increases *team situation awareness*. The team members have up-to-date and accurate

information about what is required from them and what everyone else is doing. The *organizational goals* define the *product goals*. The organization brands the product: the product has to meet the organization's standards for quality and style. *Collaboration* occurs when the roles within the team share knowledge to work towards the product goals. Team situation awareness is a necessary precondition for collaboration. Collaboration results in an *improved game*.[3]

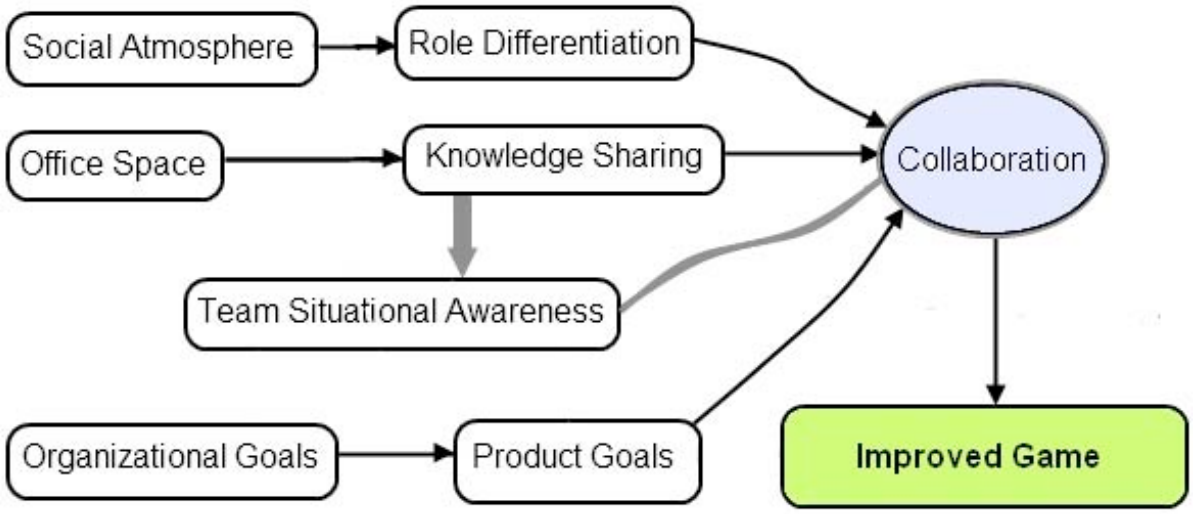

Figure 2: Qualitative model of collaboration in the game development team.

Office Space

The layout of the office and availability of shared artifacts can be designed to facilitate and encourage collaboration. Figure 3 provides pictures and an overhead sketch of the open plan office. The office has two large desks (partitioned into four workstations) and one small table. The room is designed to fit a team of eight. The open layout of the office encourages group conversations and *informal cross-training*[4]. Group conversations are conversations that involve more than two people. Group conversations tend to be more productive because there are ideas, and the ideas come from more perspectives (280, 394, 835, 899, 2535)[5]. Group conversations also increase the likelihood of consensus and acceptance of design decisions. Informal cross-training refers to learning across disciplinary boundaries. Since the entire team sits in the same open office, team members can listen and learn from the conversation between team members, even if it does not relate directly to their work. Learning also occurs when members talk aloud about their work. It is common for team members to announce milestones they've achieved or vent their frustration about problems they cannot solve. In the open layout, no conversation is private. Thus, team members are always aware of the main issues. This results in an increased awareness of what is going on with the work of others. The open plan, along with the team's willingness to discuss their work aloud, made the environment effective for maintaining situational awareness.

Figure 3: The open plan office made it easier to start and join conversations.

As alluded to earlier, we found the team members discuss their work and share knowledge frequently (24, 183, 547, 1420, 2574). This may be due to the opportunity afforded by the office layout. The layout of the office ensures team members are always visible and within talking range. There is usually no need to schedule times and rooms for meetings. Everyone is always *right there*. This results in frequent, but brief, group meetings.

An advantage of frequent group meetings is the ability to coordinate and plan work on short time scales, even by hour (125, 475, 1313, 1810, 1904, 1929, 2183). Assigning tasks and updating on progress occurs regularly throughout the day. This type of coordination is a critical feature of the team because the work of one member always feeds into the work of another. 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. The ability to synchronize inputs and outputs on short notice accelerates the development process. With team members present when they pass off their work, they can also receive feedback on short notice (68, 168, 551).

Being physically present also allows them to *work in pairs* (211, 273, 784, 1101, 2447). Working in pairs increases the quality of the work, reduces errors and facilitates problem solving. Working in pairs also allows for teaching and mentorship to occur (211, 1929). Individual members learn about how things are done from different viewpoints. Also, since individuals in this team all have different disciplinary backgrounds, the learning that occurs is even greater. The outcome is team members who are more knowledgeable of, and empathetic towards, different disciplines and ways of working. An example of troubleshooting as a pair is shown here:

> *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 [game component] always like that?*
> *LA: What do you [see]?*
> *GP: Did I screw something up? I made a mistake?*
> *LA: I don't see the [component]*
> *GP: That is the [component] right?*
> *LA: What is?*
> *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

Social Atmosphere

The team consists of the lead artist (LAR), artist (ART), interface designer (GUI), game programmer (GP), usability expert (UE) and manager (MAN). The thickness of the line indicates the strength of the *collaborative partnership* between two members. A collaborative partnership refers to a link between two team members, based on how often they communicate with each other. Figure 4 is a social network diagram of collaboration partnerships, generated from statistical analysis of conversation transcripts.

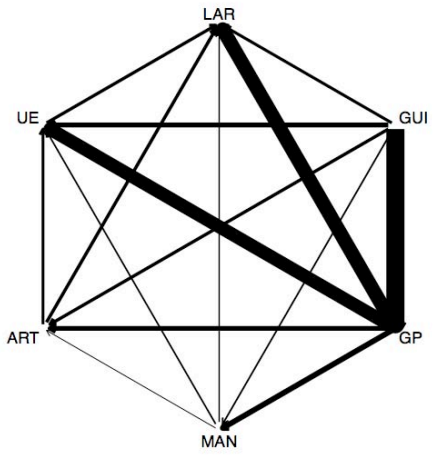

Figure 4: The thickness of the line indicates stronger collaborative partnership.

Figure 4 shows that the game programmer (GP) is involved in strong collaborative partnerships with the interface designer (GUI), lead artist (LAR) and usability expert (UE). This is not surprising because the game programmer takes input from those team members and compiles them into the final game product. The other team members do not need to collaborate as much with each other. The game programmer is the only one who works with all components. This creates a type of *funneling of work.* The component pieces of the game are created mostly as individual work, but then the collaboration occurs when it is time to integrate the pieces. The corollary is that the game programmer also assumes the role of a generalist on the team. He has advanced technical knowledge of the work of other team members.

Off-work friendships have an effect on the social network in the office as well. Overall, the social atmosphere is friendly. Team members know each other and socialize with each other when off work. These collegial bonds spill into the workplace. There is very little friction between team members, and they enjoy working together and want to collaborate. The positive social atmosphere makes negotiation work and learning easier. Team members are willing to admit their own faults and in turn, are willing to help one another (4, 394). Humor is also used often during conversations; it keeps the atmosphere informal and energy level high. Humor is used to relieve tension in stressful situations and likely helps build friendship bonds (106).

Team Situational Awareness

The team situational awareness is having a strong shared mental model of what is going on in the working environment. A shared mental model is essential for collaboration. A common background of experience and knowledge is required for working together. Otherwise, the team members could not understand each other. A strong shared mental model allows team members to work together without being explicit about everything. This allows them to skip the steps explaining the background to a problem and start straight away on the problem itself. The team mental model is built from communicating to each other and simply being part of the same team. They accumulate experience working on games together and learn to leverage each other's strengths. The physical layout and social atmosphere supports a strong team mental model by making communication and passive information gathering easy.

Organizational Goals

The organizational goals give the team direction to put their effort. This company set high standards for their game products; they were client-centered, committed to usability and conducted rigorous in-house testing. While it is easy to overlook or be cynical about organizational goals, without clearly defined objectives it is difficult to apply a concerted effort. The team members are, after all, employees who are paid to do tasks determined by the organization. The organization establishes procedures, work flow and provides the software and equipment. These are typically in place to improve the efficiency and accountability of the team. For example, the management was keen on implementing systems to track source code changes and also create a database of game bugs. The team was strongly encouraged to start using these systems. While it is unknown whether these systems were successfully implemented, the organization can influence work practices through these initiatives. This created additional tasks for the team, which they must adapt to their preferred way of working.

Role Differentiation

The roles on this team were diverse. The diversity of roles means there were many specialists on this team. Each individual is the expert in their domain. Since the team was a small team, it meant they were the sole expert on the team. Every member was critical. A game could not be completed without a full collaborative effort from all members. This fact may have lead to increased *role respect*.

The team members respect each other's roles. Each member contributes something different to the team: they all have different jobs. When someone exceeds the obligations of their job, it is noticed (494, 648, 1090). In this example, the usability expert (UE), in half-jest, is taking notice of game programmer's (GP) 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: (mockingly) Anyone get the feeling that [GP] wants to be our manager?*

The team member's roles are not completely exclusive. There are areas where the skill set of the members overlap. Sometimes when the skills overlap, it becomes 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 capable of making game animations. There is also

role overlap of usability skills between the game programmer, GUI designer and usability expert (1233, 2347). In this example, the game programmer and the GUI designer are negotiating a problem. Each is approaching it with a different solution:

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: [no]
GP: (perplexed) You put a grey [shadow] behind a white [letter].
GUI: No actually I didn't, but you could do that

Sharing Information

Communication is the foundation of collaborative activity. Broadly speaking, there are three types of face-to-face communication that occur in this environment. There are private conversations, group conversations, and broadcasts. Private conversations are simply conversations between two people. It does not mean it occurs in a private location. Since the team works in an open plan office, private conversations can be heard by the rest of the team.

Group conversations evolve from private conversation. Occasionally, if a third person finds the private conversation interesting, they will join in the conversation to make it a group conversation. In this next example, the GUI designer is troubleshooting with the Game programmer, but the Usability expert hears the conversation and ends up helping:

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: (Joining the conversation) 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.

The third kind of communication is broadcasting. Broadcasts start with an open-ended remark. Sometimes, they are not even mean to elicit a response. Their purpose can be to provide updates

to the group. Typical broadcast messages are the Program Manager announcing a development milestones or team member boasting about a recently finished task. Broadcasts add to the situational awareness of the team.

A content analysis of the communication reveals the frequency of topics discussed in conversations. Figure 5 shows the breakdown of topics discussed when the conversations were related to work; conversations un-related to work are not included in the graph. The graph shows close to one quarter of the work-related conservations were for planning (scheduling, progress updates, etc.). More than 60% of conversations were about the game itself. The rest of the conversations were about the development software (tools) and other topics.

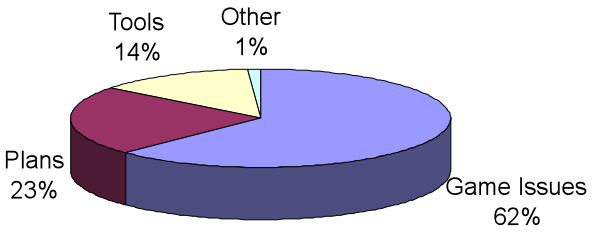

Figure 5: Breakdown of conversation topics.

Other ways of communication are through the computer (email, messaging) and by making notes on the whiteboard. There are four large whiteboards in the office space. The whiteboards are used to jot notes, ideas, and to sketch game content. The whiteboards are a way to communicative asynchronously. 'Conversations' on the whiteboards also persist for a longer period. They can be reviewed and revised at anytime. Communication through the computer has the same effect, but is even more convenient and permanent than the whiteboards. Conversations through e-mail and messaging were not monitored during this study. However, it was seen to be used by the team in conjunction with face-to-face communication. In many cases, the conversations would refer to things being said in an e-mail or other software messaging system.

Product Goals

The product goals are manifest through the *game implementation hierarchy*. The hierarchy is our interpretation of how the team conceptualizes what the game product is about (Figure 6). The hierarchy was created based on the language the team uses to describe the components of the game. The team's conception of the game influences how they talk when they collaborate. They tend to move up the hierarchy of abstraction when talking with other members who are not as knowledgeable with the technical domain. Moving up and down the hierarchy facilitates communication between disciplines because it avoids the use of technical terms.

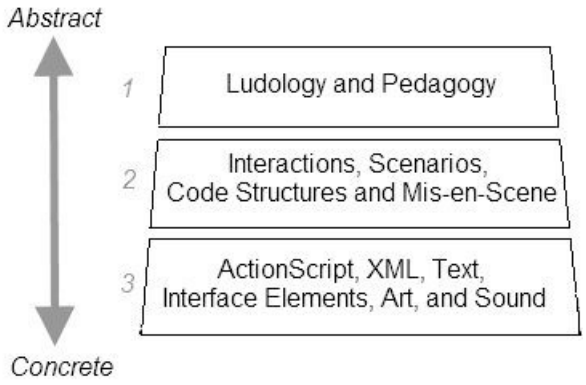

Figure 6: The main components of the game, organized into a hierarchy.

The game hierarchy has three levels. The highest level of implementation involves ludic and pedagogic components. These two components capture the high level goals of the game; the objective is to play and learn at the same time. The second level of implementation includes interactions, scenarios, code structure and mis-en-scène. Interactions refer to objects in the game that are manipulated by the user (1490, 2461). Scenarios are the stories and setting of the game (643, 973). The mis-en-scène is the totality of what is presented on the users screen (2570). The code structure refers to how the source code is organized (1400). The source code and source files make up the lowest level in the implementation hierarchy; it includes XML, ActionScript, text, interface widgets, art and sound. The XML is the file that contains the game data and text (691). ActionScript is the programming language that determines the runtime execution of the game client. Text refers the 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). Lastly, there are the sound components (506).

Individual work usually involves components at the lowest level of the hierarchy. Individuals are responsible for concrete elements of the game, such as source code and source files. However, when the team collaborates, it is usually on a component from the second level of abstraction. In this example, three members are discussing a problem with the mis-en-scène. The game programmer understands it in terms of code and xml. The GUI designer understands it in terms of GUI components. The lead artist understands it in terms of graphics (art components). However, they don't discuss it in their own terms during the conversation. They use terms from components higher in the hierarchy. By speaking about the game more abstractly, it is easier to talk about the game without getting bogged down by technical issues:

> *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 [*interaction *object] and it takes you there, but you can't see the [object anymore] because the panel is up.*
> *LA: That is a big problem.*

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

Game Improvements

The game product is developed in rough form, and then polished throughout the development process (1309). It is important to have a working prototype early. Polished components are then added piecemeal (537, 584, 2351). Components along with the entire game get tested as soon as something new added. The game product is under continuous evaluation throughout development through quality assurance testing (2119) and usability testing (1297. 1769, 2543). One downside of this way of creating the game is that when new components are added, the problem of *cascading dependencies* often appears. Cascading dependencies refer to having to make changes in previously tested code to allow the new component to work, forcing 'good' code to be modified and re-tested (2481).

GUI: You can do the [change] right? Just scooch the [object] down, 6 pixels, 6 notches, it looks fine still
GP: And you're 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 component out of place)
GP: Oh a number. Oh cause all the bars. (realizing the problem) 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
GUI: Yeah.

The team has a highly *visual approach* to game development. This means they are concerned primarily by how the game looks, above all else. The value placed on the visible result is summarized by this quip from the game programmer: "I don't know anything about the scaling or the size. This is just what it looks like in the game" (2698). The game programmer does know about the scaling, but is emphasizing his view that the numbers in the code are secondary to the result on screen. Changes to the source code are dictated by how it affects the visual result (56, 349, 992, 2235, 2266). Similar to the visual approach, the team also has a *do-it-and-see* approach. Since the development platform (Flash) is a lightweight platform, the do-it-and-see approach works well. Components can be easily added, tested, and if necessary, removed from the game. The consequence of a *do-it-and-see* approach is the de-emphasis on documentation and elaborate design planning. The positive, however, is an emphasis on experimentation and testing. In this example, the game programmer and lead artist are discussing an early prototype of the game. They are gauging the next steps based on how it currently looks:

GUI: I'll look at them cause it's all like code and then you can make some suggestions 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.

Conclusion

This section is our interpretation the most important factors contributing to a successful collaborative processes. There are three core factors: *role respect, short iteration cycles* and *shared vision*. These factors constitute the team's *collaborative spirit*. The collaborative spirit is strengthened by the *socio-technical infrastructure.*

Role Respect
There was a clear understanding of how each member contributes to the game product. Each member focuses on their own work. They do not negatively criticize the quality of other member's work because they are not in competition with each other. Team members understand how their roles are complementary in the multi-disciplinary environment and concentrate on their own responsibilities. The team values collaboration because they could not create the game product by themselves. The team members also show interest in learning about other member's work because they view the other's work as valuable.

Short Iteration Cycles
The game product goes through changes (and testing) almost daily. This requires constant communication between team members to integrate the components. The game product is evaluated early and often throughout development. The short 'refine and test' cycles forces team members to remain engaged and up-to-date about development milestones. Team members are usually working on something that will go into the game product within a few days. The iterative process itself is therefore a driver of the collaborative process because it demands constant interaction.

Shared Vision
There is common understanding about what the game product is and should be. There are no conflicting goals about the end product. Team members may disagree about implementation details, but they all have the same vision of what a good serious game should be. They understand that the product should be fun, educational and easy to play. Having the same values about the game allows them to handle disputes and move forward with decisions more easily.

Collaborative Spirit
The three factors mentioned above produce a strong *collaborative spirit*. Collaboration has become an essential part of their work. More importantly, it is seen as a normal part of their work; it is something they do without second thought. Collaboration is easy for them and they have a common goal to work towards. The development model rewards frequent communication and interaction, and their contributions toward the goal are valued.

Furthermore, the social atmosphere is a catalyst for the collaborative spirit. The social atmosphere is friendly. All opinions are valued equally; they feel they are working amongst peers. There is no sign of a hierarchal power structure, and they do not have to be reserved or censor themselves. Team members get along in and outside the workplace. This relaxed environment allows them to speak freely and honestly; it is a place where the team members are comfortable working alone, and with others.

Summary

We have described a process of game development based on ethnographic field work and qualitative analysis. Our main finding can be summarized as this: *effective collaboration requires a team that respects each other's contributions, communicates frequently and shares a similar conceptual model of the product and goals*. This study gives a holistic view of the collaborative process, emphasizing the importance of communication, social relationships, physical infrastructure, shared knowledge and organizational goals. We hope it can provide some useful insights on how to foster collaboration within game development teams. We also hope it contributes to the understanding of games, by illuminating the development process through which they are created.

Acknowledgements

We thank the members of the development team who participated in this study. We admire their dedication to creating great games and professionalism that was evident in their work. We are also grateful for the financial support provided by the Ontario Research Network for Electronic Commerce (ORNEC).

References

Adams, E., & Rollings, A. (2007). *Game Design and Development: Fundamentals of Game Design*. New Jersey: Pearson Prentice Hall.

Adobe Flash [Software]. (2008). http*://www.adobe.com/products/flash/*.

Beyer, H., & Holtzblatt, K. (1998). *Contextual Design*. San Francisco: Morgan Kaufmann.

Blandford, A., & Furniss, D. (2005). DiCoT: a methodology for applying Distributed Cognition to the design of team working systems. *Proceedings from Design, Verification and Specification of Interactive Systems, Newcastle-upon-Tyne, UK, 2005*, 26-38.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3(2),* 77-101.

Clark, H. (1996). *Using Language*. Cambridge, MA: Cambridge University Press.

Clark, H., & Brennan, S. (1991). Grounding in communication. In L. B. Resnick, J. M. Levine, & S. D. Teasley, *Perspectives on socially shared cognition*. Washington, DC: American Psychological Association.

Dickey, M. D. (2005). Engaging by Design: How engagements strategies in popular computer and video games can inform instructional design. *Educational Technology Research and Development, 53(2),* 67-83.

Finkelsetein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., & Goedicke, M. (1992). Viewpoints:

A Framework for Integrating Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering, 2(1),* 31-58.

Gee, J. P. (2007). *What Video Games Have to Teach Us about Learning and Literacy* (2nd ed.). New York: Palgrave Macmillan.

Glaser, B. G., & Strauss, A.L. (1967). *The Discovery of Grounded Theory. Strategies for Qualitative Research.* Hawthorne, NY: Aldine Transaction.

Huh, J., Ackerman, M. S., Erickson, T., Harrison, S., & Phoebe, S. (2007). Beyond usability: taking social, situational, cultural, and other contextual factors into account. In *Proceedings of CHI*, *San Jose, California, 2007*, 2113 - 2116.

Hutchins, E. (1995). *Cognition in the Wild.* Cambridge, MA: The MIT Press.

March, J. G. (1991). Exploration and Exploitation in Organizational Learning. *Organization Science, 2(1),* 71-87.

Malaby, T. M. (2009). *Making Virtual Worlds: Linden Lab and Second Life*. New York, NY: Cornell University Press.

Martin-Rodriguez, L.S., Beaulieu, M. D., D'Amour, D., & Ferrada-Videla, M. (2005). The determinants of successful collaboration: A review of theoretical and empirical studies. *Journal of Interprofessional Care, 19*, 121-147.

Muchinsky, P. (2003) *Psychology Applied To Work, Seventh Edition*. Toronto, CAN: Thompson Wadsworth.

Mühlpfordt, M., & Wessner, M. (2005). Explicit referencing in chat supports collaborative learning. In *Proceedings of Conference on Computer Support for Collaborative Learning, Taipei, Taiwan, 2005*, 460-469.

Perry, M. (2003). Distributed Cognition. In J.M. Carroll (Ed.) *HCI Models, Theories and Frameworks: Toward a Multi-disciplinary Science*. New York: NY: Morgan Kaufmann Publishers, 193-233.

Pietroburgo, J., & Bush, B. (2008). Coming to Terms: A Case Study of Hospice Collaboration Challenges. *American Journal of Hospice & Palliative Medicine, 24(6),* 487-492.

R [Software] (2009). *http://www.r-project.org/.*

Robinson, H., Segal, J., & Sharp, H. (2007). Ethnographically-informed empirical studies of software practice. *Information and Software Technology, 49(6),* 11.

Roschelle, J., & Teasley S.D. (1995). The construction of shared knowledge in collaborative problem solving. In C.E. O'Malley (Ed), *Computer-Supported Collaborative Learning.* Berlin: Springer-Verlag.

Rossen, E. K., Bartlett, R. & Herrick, C.A. (2008). Interdisciplinary Collaboration: The Need to Revisit. *Issues in Mental Health Nursing, 29,* 387-396.

Scacchi, W. (2001). Process Models in Software Engineering. In J.J. Marciniak (Ed.) *Encyclopedia of Software Engineering, (2nd Ed),* New York, NY: John Wiley and Sons.

Sharp, H., & Robinson, H. (2008). Collaboration and co-ordination in mature eXtreme programming teams. *International Journal of Human-Computer Studies, 66(7),* 506–518.

Whitworth, E., and Biddle, R. (2007). The Social Nature of Agile Teams. In *Proceedings of the AGILE Conference, Washington, DC, 2007,* 26-36.

Ye, Y. (2006). Supporting Software Development as Knowledge-Intensive and Collaborative Activity. In *Proceedings of International Workshop on Interdisciplinary Software Engineering Research, Shanghai, China*, *2006,* 15-22.

---

[1] A shorter version of this paper was presented at ACM FuturePlay: International Academic Conference on the Future of Game Design and Technology 2008, Toronto, Canada. DOI: http://doi.acm.org/10.1145/1496984.1496993

[2] Problem solving was confined to troubleshooting bugs in the game code. Most commonly, bugs in this game are from failure to recognize all the dependencies when adding new elements to the existing program code. This results in anomalies when playing the game, such as missing graphics textures or non-responsive interface elements.

[3] An extension of this model would ideally show that major improvements in the game product lead to *innovation in game design*. In other words, the team would have refined a specific feature well enough that they would want to incorporate it into other games. However, claims about innovation overstep the findings of our study.

[4] Novel themes are marked in italics.

[5] Numbers in parenthesis are references to the original transcript file. Where possible, we have included direct quotations from the transcripts instead.