Collaboration in Serious Game Development: A Case Study

Minh Quang Tran
Human-Oriented Technology Lab
Carleton University
Ottawa, Ontario, Canada
mqtran@connect.carleton.ca

Robert Biddle
Human-Oriented Technology Lab
Carleton University
Ottawa, Ontario, Canada
Robert_Biddle@Carleton.ca

ABSTRACT

This paper presents an ethnographic study of the development practices at a small but successful company that develops serious games for industry training. We concentrated on the day-to-day collaboration of the studio team responsible for the game content and software, and used qualitative research methods, including field observations, contextual interviews and conversation analysis. This paper reports our findings, emphasizing a holistic perspective encompassing social and technical factors influencing collaboration in serious game development. In particular, we report on how co-location and a positive social environment work together with the technical tools and infrastructure to provide an environment that facilitates full participation of professionals with differing disciplinary perspectives, and contributes to iterative development and refinement of the game.

Categories and Subject Descriptors

K.6.3 [**Software Management**] Software process

K.8.0 [**General**] Games

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Organizational Design, Synchronous Interaction, Theory and Models*

General Terms

Human Factors, Theory.

Keywords

Serious Games Development, Collaboration, Ethnography, Grounded Theory.

1. INTRODUCTION

Designing and developing computer games is a complex process that involves people from several disciplines working in a collaborative effort towards a creative goal. The distinctive nature of the computer game product means that the development process must allow for continuous reflection and refinement of the product throughout development. Moreover, as serious game development becomes more widespread as an approach to education, health-care, and communication, there is a need to understand the game development process outside large organizations with a focus on the commercial entertainment market.

In this paper, we present a study of the development practices at a small but successful company that develops serious games for industry training. In particular, we concentrate on the day-to-day collaboration of the studio team responsible for the game content and software. We use qualitative research methods, including ethnographic field observations, contextual interviews and conversation analysis. In our analysis, we emphasize building a holistic theory to describe how social and technical factors influence collaboration in game development. Our research questions were:

(a) How does the environment affect collaboration?

(b) How is information shared?

(c) How does the game product change as a result of the collaborative process?

2. BACKGROUND

Other studies have also looked at collaboration among technical teams using qualitative methods [1] [2] and quantitative methods [3] [4]. However, they are limited in scope, choosing to highlight only a few issues in collaboration or comparing a small set of variables. Elliot [5] comes closer to a general theory of collaboration by distilling collaboration to three core elements: people participating in communication network, emergent shared understanding and creative output. His conclusion was based on observations and experiences with mass collaboration, particularly Wikipedia and open source software development. Our research and resulting theory is situated in a different context. We present a model of collaboration of a small group working in the domain of serious game development.

Serious game development may differ from other collaborative endeavors due to the reliance on technology, the nature of the design activity and multi-disciplinary teams. Serious game development requires extensive use of software. It is used for the development activity and increasingly it is used for the coordination of the development activity. Game development cannot be thought of as a separate issue from software usage. Therefore, the study of game development cannot ignore issues of general human-computer interaction. Video game development is also a subset activity of software development. As a result, it shares many of the same software engineering problems relating to abstract design [6][7]. Furthermore, video game development tends to require multi-disciplinary collaboration. Game development teams integrate concepts and work practices from many different professional domains that are, methodologically, very distinct. They range from creative professionals to scientists and engineers; examples of typical professions involved in game development include artists, writers, domain experts, engineers and user experience researchers. These aspects may make serious game development teams more likely to encounter problems with communication and politics [8].

3. METHOD

We used qualitative research methods to learn and gather information about how collaboration was occurring within the game development team. This included ethnographic observations, contextual interviews and audio recordings. To generate a theory of collaboration, we used Grounded Theory techniques including coding and categorizing emergent themes. An affinity diagram was created to consolidate and connect the themes.

3.1 Participants

The participants were all working in the same game development team consisting of 5 males and 1 female. The team included two graphic artists, an interface designer, a usability expert, a game programmer and a production manager. The team develops serious games on the Flash platform [9]. They are part of a start-up company of 25-35 employees. The team members have experience working together for least 6 months and some have been working together for 2 years.

3.2 Field Observations

The production manager was sent a letter with the research proposal weeks in advance of the proposed study dates. He was asked to provide verbal consent on behalf of the team. Once they agreed to participate, the date and times were settled for when the ethnographic field observations could be performed. Before beginning field observations and data collection, each team member was required to sign a formal consent letter. This letter outlined the research purpose and conditions of participation. Participants were allowed to leave the study at any time, at which point their data would be removed from the study. All participants stayed in the study until the end.

3.3 Data Sources and Collection

The team was observed working on game development for two weeks. The observations sessions lasted an average of 4 hours a day. The main data source is audio recordings of conversations and observations of group work. The conversations were captured with a digital audio recorder placed in the center of their work room. All team members work in the same room. Since the room is an open space without cubicles or walls, the conversations were easily captured. In total, the audio files are 30 hours in length. They were then transcribed into text format for further analysis. To complete the data set, information was gathered about the physical room layout, work procedures and use of media, such as software and whiteboards. This was done through informal interviews, collecting corporate documents and taking photographs.

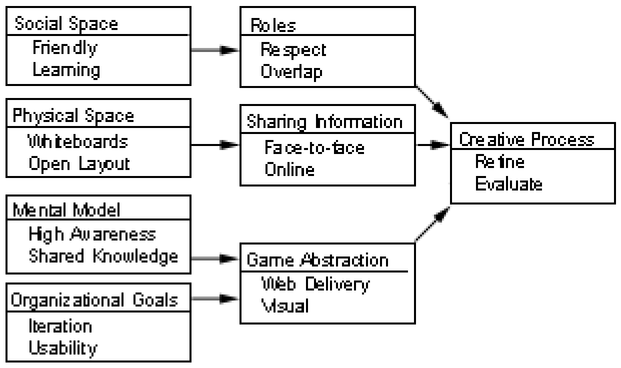
3.4 Data Analysis and Writing

The analytic process was based on Grounded Theory coding techniques [10]. This process included manipulation of the textual data by performing a series "coding" exercises. The goal was to build a theory by extracting themes surrounding and leading to instances of collaboration. The content analysis exercise started by finding recurring patterns of action or speech where collaboration occurred (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). [11]

In the following section we start by presenting a diagram of our model of collaboration for this team. Then we explain the themes in more detail. Major themes are noted by using italics. We include some conversation passages to help ground the reader in the data. For brevity, we have limited the number of passages. Instead, we provide line numbers as references to where they are in the textual data. These references are noted inside parentheses.

4. RESULTS

Our model for collaboration among serious game developers is shown in Figure 1. Each box represents a collection of themes that were identified through the analysis.



The first column of themes is contributing factors to the collaborative process. The components of the collaborative act are noted in the second column. Finally, the collaboration leads to the

Figure 1. Theoretical model for collaboration in game development teams.

collaborative process identified in the final column

4.1 Contributing Factors to Collaboration

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

4.1.1 Physical Space

The layout of the room and availability of shared artifacts 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 for gauging overall team progress. It also helps keep the team informed about what others are doing. This results in increased *team awareness*.

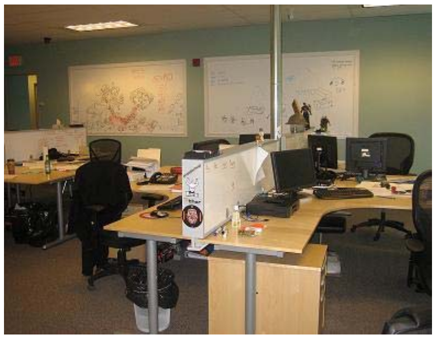


Figure 2. Physical Space Photo.

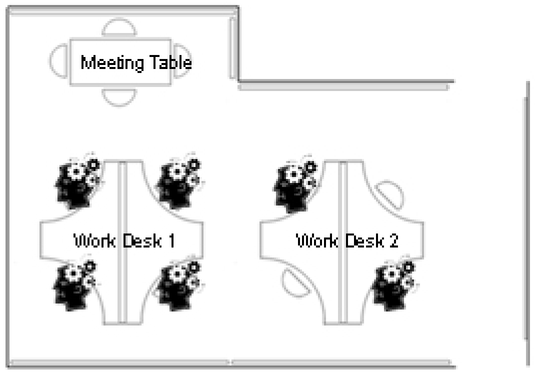


Figure 3. Physical Space Layout.

We also found the tendency to share experiences and expertise very high 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. As a result, the team is more empathetic to different disciplinary perspectives and approaches.

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 [12]. In this study, we notice mentorship was especially prevalent (211, 1929).

4.1.2 Social Space

The social environment is *friendly* and team members are committed to *learning* from each other. We present the main people in the *social space* via a social network diagram in Figure 4. The thickness of the line indicates the strength of the *collaborative partnership* between two people. A *collaborative partnership* refers to the frequency or need for two people to communicate or work together. The lines in the diagram were generated from aggregated data based on frequency analysis of the face to face conversations between team members. Based on this diagram, the Game Programmer (GP) is involved in the most number of conversations. This is expected since the game programmer role is to integrate the contributions of all other team members.

Game components are given to the programmer so that he can integrate them 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 all other team members at some point during development. It is evident in the social network that there is a funneling of information and collaborative activity towards that role.

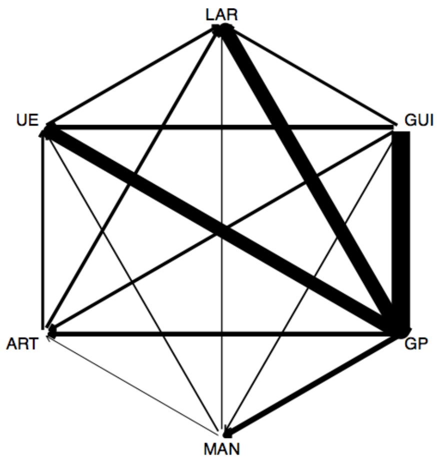


Figure 4. Social Network with Lead Artist (LAR), Interface Designer (GUI), Game Programmer (GP), Production Manager (MAN), Graphic Artist (ART) and Usability Expert (UE).

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 enjoy 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 build 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.

4.1.3 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 work effectively since the work of one member often depends on the completion of some other work by another member.

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.

4.1.4 Organizational Goals

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 way of working.

4.2 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.

4.2.1 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 making notice of the 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: 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, 3D fonts are being added to the game, and the Game Programmer and GUI Designer are negotiating how the work will be done:

> *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.*

4.2.2 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 are easier to solve when working-in-pairs, as seen in this exchange between the GP and Lead Artist (LA):

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 reduces frustration from troubleshooting because two people can pool their effort to solve a problem more effectively. It also eliminates problems arising from working on two differently configured computers. Computers that are configured differently display the game product differently. This is especially a problem in terms of color and resolution. Troubleshooting on two different computers can present an additional challenge because errors or differences in the game product can result from having a different computer configuration.

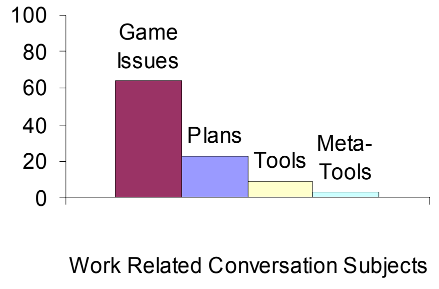
Public conversations are conversations that involve more than two people. Since everyone is in the same physical space, even the private conversations can be heard. Occasionally, if a third person finds the private conversation interesting, they will join in, turning it into a group or public conversation. 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 a development milestone 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.

Figure 5 shows the relative frequency of work related topics discussed during face-to-face communication. The most popular topic of conversation relates to game issues. However, a more interesting finding is that more than 20% of the communication is for planning and coordinating work. The other topics include discussion of game development tools and tools for collaboration (Meta-Tools).

Figure 5. Conversation topics.

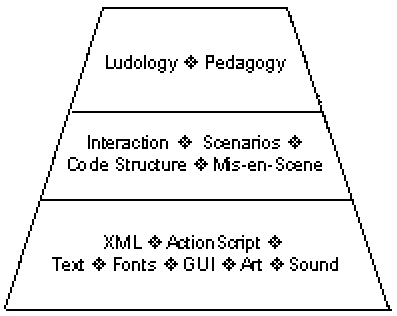


Aside from face-to-face communication, another way information is shared is via external artifacts. Specifically, we are referring to software and whiteboards. The software for communicating includes email, bug tracking software and file management systems. Whiteboards are also placed throughout the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

4.2.3 Game Concept

The game concept is how the team thinks about the game and is illustrated in Figure 6 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.

Figure 6. Game Implementation Hierarchy.


Ludology ❖ Pedagogy
Interaction ❖ Scenarios ❖
Code Structure ❖ Mis-en-Scene
XML ❖ ActionScript ❖
Text ❖ Fonts ❖ GUI ❖ Art ❖ Sound

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 Production Manager is primarily responsible for defining objectives at this level, but does take 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 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 scene 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 Programmer 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. Here, three team members discuss an interaction issue, but avoid referring to how the solution would be implemented at a lower level:

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?

4.3 Creative Process

The game product is development iteratively. 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* refers to making several changes in many places to achieve one effect. This is normally due to functions and objects that 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.
GUI: Yeah.

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 result is summarized by this quote 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). Modifications to the game product are evaluated by how it affects the visual end-product, with less emphasis on efficiency of the *code* (56, 349, 992, 2235, 2266).

A related tactic used in the creative process 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. This 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.

5. INTERPRETATION

This section gives our interpretation of what we saw as the most important factors contributing positively to the collaborative process. The three main factors we believe have the largest influence are *role respect*, *iterative development* and *shared vision* for the game product. Two secondary factors are a *collaborative spirit* and supportive *technical environment*.

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. Each member 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 the game product requires input from the entire team. 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 influence collaboration is the *iterative development* process because it demands frequent meetings and evaluation of work. 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 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 the 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, 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 are no signs of seniority or hierarchical structure. Several things contribute to this collegial mood. The team members are all roughly of 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 limited by bureaucratic controls.

The informal nature of the technical and social environments is ideal for fostering ideas and collaboration. The environment is relaxed and non-judgemental. 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. Accordingly, the playful and business-like aspects of this work environment can be seen reflected in their serious games products.

6. CONCLUSION

We have described a collaborative process of game development based on ethnography and qualitative analysis. The ethnography was performed over 10 days during which we gathered data via in-situ observations, contextual interviews and conversation recordings. A Grounded Theory analysis was then performed using the conversation data while leveraging the richness of information from the ethnographic observations. Several themes emerged from the analysis, which we then consolidated into a model of collaboration.

The model of collaboration includes a broad range of factors. It puts the focus on social relationships, physical resources, team knowledge and organizational goals. Our main finding can be summed up as such; *effective collaboration requires a team that respects each other's contributions, communicates frequently and shares a similar conceptual model of the product and goals.* Collaborative acts are also more likely to occur in development cycles that are iterative, where developers are constantly refining and evaluating the product. The organization also plays a role by fostering the collaborative spirit and providing the physical tools.

Future plans include validating the model by continuing observations specifically with a focus on refining the current model. Participants will also be invited to discuss the current work to obtain their thoughts on how well it characterizes their environment.

7. REFERENCES

[1] Farooq, U., Carroll, J.M., & Ganoe, C.H. (2007). Supporting Creativity with Awareness in Distributed Collaboration. GROUP'07, November 4–7, 2007, Florida, USA.

[2] Zhang, Y. & Candy, L. (2007) . An In-depth Case Study of Art-Technology Collaboration. Creativity & Cognition '07, June 13–15, 2007, Washington, DC, USA.

[3] White, J., Lyons, J.B. & Swindler, S.D. (2007) Organizational Collaboration: Effects of Rank on Collaboration. Proceedings of the ECCE 2007 Conference, 28-31 August 2007, London, UK.

[4] Zumbach, J., Schönemann, J., and Reimann, P. (2005). Analyzing and supporting collaboration in cooperative computer-mediated communication. In Proceedings of The 2005 Conference on Computer Support For Collaborative Learning, Taipei, Taiwan, May 30 - June 04, 2005, 758-767.

[5] Elliott, Mark. (2007) Stigmergic Collaboration: A Theoretical Framework for Mass Collaboration.(Ph.D. Dissertation, Centre for Ideas, Victorian College of the Arts, University of Melbourne).

[6] Wirfs-Brock, R, McKean, A. (2002). Object Design: Roles, Responsibilities, and Collaborations. Addison-Wesley Professional.

[7] Keil-Slawik, R. (1992). Artefacts in Software Design. In C. Floyd, H. Zullighoven, R. Budde, R. Keil-Slawik (Eds.), Software Development and Reality Construction (pp. 168-188). New Collaboration in Game Development 116 York, NY: Springer-Verlag.

[8] Chamberlain, S., Sharp, H. & Maiden, N. (2006). Towards a Framework for Integrating Agile Development and User-Centered Design. In P. Abrahamsson, M. Marchesi, and G. Succi (Eds.): XP 2006, LNCS 4044 (pp. 143-153). Berlin: Springer-Verlag.

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

[10] Morrow, S.L., & Smith, M.L. (1995). Constructions of survival and coping by women who have survived childhood sexual abuse. Journal of Counseling Psychology, 42, 24-33.

[11] Creswell, J.W. (1998). Qualitative Inquiry and Research Design Choosing Among Five Traditions. Thousand Oaks, CA: Sage Publications.

[12] Nosek, J.T. (1998). The case for collaborative programming. *Communication of the ACM, 41(3)*, 105-10.