5-2014

Oral Communication in Genre Theory and Software Development Workplaces

Jason L. Cootey
Utah State University

Recommended Citation

Cootey, Jason L., "Oral Communication in Genre Theory and Software Development Workplaces" (2014). *All Graduate Theses and Dissertations*. 3896.
https://digitalcommons.usu.edu/etd/3896

ORAL COMMUNICATION IN GENRE THEORY

AND SOFTWARE DEVELOPMENT WORKPLACES

by

Jason L. Cootey

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Theory and Practice of Professional Communication

Approved:

Dr. David Hailey Major Professor	Dr. Keith Grant-Davie Committee Member
Dr. Ronald Shook Committee Member	Dr. Patricia Gantt Committee Member
Dr. Christine Hailey Committee Member	Dr. Mark McLellan Vice President of Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2014

ABSTRACT

Oral Communication in Genre Theory and

Software Development Workplaces

by

Jason L. Cootey, Doctor of Philosophy

Utah State University, 2014

Major Professor: Dr. David E. Hailey
Department: English

While "communication" often means written communication, professional communication involves other media that often goes unnoticed. However, once researchers look past print media to the digital world, there are a host of genres which professional communicators must understand to successfully practice their profession. The software industry is a field where written communication has a strong presence, indicating a failure to fully utilize the importance of other media.

Traditional software development methods require comprehensive documentation. The documentation thoroughly articulates both design details and a clear development map for the entire project. Such documentation can add to over 1000 pages in larger, government projects. However, smaller development shops work with much smaller, more contemporary projects; the software is not as big, does not service so many users, and is not expected to have a long lifespan before business needs change. In those cases, developers struggled with the same cumbersome documentation requirements required by larger software projects.

As more development shops adopted the contemporary practices, developers became frustrated with documentation requirements. Consequently, contemporary developers still struggle to stuff monolithic, traditional documentation standards into their nimble, contemporary practices.

In this dissertation, I use North American genre theory to both describe the documentation I expect developers to write and describe the way developers were supposed to write it. I use two postmortems in this dissertation to showcase my expectations and the results I found. Yet, the predictions and descriptions were wrong in each case. In my postmortems and industry, developers either fail to document or document very poorly.

The problem was not with contemporary practices, the developers, or genre theory. Rather, the problem is with the limitation of written communication and the necessary recognition of additional genres. Because recent programming practices have become so agile, the genres describing their production may include casual conversations and even quick instructions on Post-It notes stuck onto the programmers' monitors. In this dissertation, I refer to "rhetorical forms" to escape the limitations of the older tradition and include the new genres. I seek to highlight oral communication and other "pieces" of communication as the rhetorical forms genre theory predicts.

(263 pages)

PUBLIC ABSTRACT

My dissertation defines how software developers have abandoned traditional documentation practices for other kinds of media that work better in their workplace practices. Ultimately, even though other media like white boards, sticky notes, and "oral communication" are vastly different than traditional, written software documentation, they match the fast paced, decision-making situations of contemporary developer communities. I focus particularly on oral communication because it is the most unacceptable means to "document," according to traditional standards. I use North American Genre Theory to describe those decision-making situations contemporary developers and note how the theory does not account for all the documentation I expect to find. Via several projects and interviews I confirm that oral communication is a new means of "documentation" and reconciles North American Genre Theory

ACKNOWLEDGMENTS

Every page is for Karen because not a single page could have been written without her. She has waited too long but has managed to believe in me all along.

Every image is for my kids—they are only interested in the pictures anyway.

Thank you to Professor David E. Hailey for his enduring patience and rewarding collaborations.

Thank you to Professors Keith Grant-Davie, Ronald Shook, Patricia Gantt, and Christine Hailey for their mentorship and support.

Jason L. Cootey

CONTENTS

LIST OF TABLES

LIST OF IMAGES

CHAPTER I

INTRODUCTION

DISSERTATION SUMMARY

Many software industry development guidebooks outline the form and content for software development documents—user manuals, specifications, design documentation, etc. These guidebooks have prescribed rules, specific processes, and clear genres for these documents. However, as I interviewed developers, observed design meetings, sampled various documents, and managed my own projects, I discovered developers often ignored both these guidebooks and their recommendations. In fact, some of the contemporary developers with whom I spoke claim the industry guidebooks have long been out of touch. While guidebooks are still relevant to the documentation requirements of traditional development environments, where developers use more linear planning methods, that same relevance does not extend to contemporary environments, where developers use more iterative planning methods. When contemporary developers abandon industry guide recommendations they also abandon their traditional commitment to thoroughly document their development plans. Additionally, while strategies and values of community mindsets had all changed in contemporary development environments, the strategies and values for documentation activities had remained traditional.

Consequently, contemporary developers tried to fit traditional documentation in to contemporary strategies and values. After all, traditional guidebooks advocate thoroughly articulated written documentation and eschew oral communication as accidental chitchat.

In the minds of traditional developers, oral communication has such limited value they use a common idiom to diminish it: "if it isn't written down it didn't happen." However, traditional documentation will not fit in contemporary methods. Yet, contemporary developers still expected to write the traditional documentation. In short, the documentation recommended by traditional guidebooks worked poorly in contemporary software development environments, even while contemporary developers still labored to write the documents.

Dissertation Vocabulary and Research Plan

One of three things might be the explanation for the disconnect between traditional documentation and contemporary practices:

- Traditional documentation practice may be broken
- Developers may fail to comply with industry standards
- Developers comply in a nontraditional way

The answer is the last; they comply in a nontraditional way. Their written documentation may be poor but their documentation activities still meet their needs. Software developers simply do not have the language to describe nontraditional documentation activities. That is where I use North American genre theory as a meta-language to describe how contemporary developers shifted documentation activities to accommodate nontraditional values, like oral communication.

The dissertation's vocabulary. The guidebooks demand that careful project plans and comprehensive design details all be officially recorded, rather than merely discussed and forgotten. These prescribed documentation activities traditionally involve documents with sometimes encyclopedic levels of detail and comprehensively describe the objectives and milestones on a linear project plan. Developers are meant to draw their documentation vocabulary from those industry guides and describe documentation with vocabulary provided by those guides. Having been immersed in these prescriptions from the beginning of programming history, developers find it difficult to describe something other than a carefully written document in their documentation activities. In contrast, there is no vocabulary to articulate documentation that may not fit the traditional model. Therefore, when developers refer to "documentation," they refer to an officially labeled development activity and contemporary developers still use that traditional vocabulary when they talk about the act of "documentation" in their contemporary work environments. With that in mind, I employ North American genre theory because it comes with both predictive power and a vocabulary to describe communication practices outside the framework of traditional documentation practice.

North American genre theory is not the only theory of genre studies; however, it is the genre theory I use in my research. Classic genre theory and Sydney Australia genre theory will be designated as such but I will always refer to North American genre theory as simply "genre theory."

While I can easily describe traditional documentation with genre theory, I do not tend to find the contemporary documentation genre theory seems to predict. This is because 1) the industry standards and vocabulary exclusively privilege traditional (written) documentation and 2) contemporary developers either do not produce traditional documentation at all or they do so only superficially. Even so, it does not follow that contemporary developers do not perform documentation activities, even if there is no reliable trail of written communication. Rather, developers perform documentation activities with new rhetorical forms that include oral communication.

I describe contemporary documentation and the meta-language of genre theory gives me the vocabulary I need. In this dissertation, I use a meta-language provided by genre theory to describe two project postmortems and a set of interviews. The vocabulary of this meta-language serves as a foundation for my argument.

Dissertation research plan. I sought to produce documentation as prescribed by industry guidebooks and I had genre theory to describe the success or failure of my re-creations of traditional documentation. I vetted my findings with my contacts in the field and I sought new contacts with whom I could field additional questions.

The narrative of my two project postmortems highlight the pain points contemporary developers have with traditional documentation. I cap my dissertation with a contemporary software development company in Salt Lake City, Utah; I interviewed seven developers. I used the meta-language of genre theory to predict what I would find and I expected that the interviews would verify an oral shift in traditional documentation standards. In the end, although the interviews did not yield what I expected, I had a meta-language to describe both the ecosystem of communication genres the interviewees employed, even if the communication genres I describe do not match the recommendations of traditional industry standards.

DISSERTATION RESEARCH PROBLEM

The research problem is informed by the fact that contemporary software developers no longer follow the old prescriptions to produce traditional documentation, according to industry standards. The problem suggests the following questions:

1. What do they use instead?
2. Are current approaches appropriate?
3. If they are not appropriate, what should the developers be doing?

It is hard to answer those questions with traditional vocabulary. For instance, "if developers don't write traditional documentation then what do they write?" As long as their vocabulary keeps referencing formally written communication (in the absence of all other possibilities), their vocabulary is not useful. I simply cannot use traditional rules and traditional guidebook language to describe the communication practices of contemporary developers. However, I can use rubrics derived from genre theory to describe what the contemporary developers actually do, the effectiveness of what they actually do, and what their documentation should look like.

Research Problem and North American Genre Theory.

Genre theory is a model researchers use to describe how communicators might formulate flexible rules based on social interaction (e.g., proposing a service) for the form and content of genres, rather than conform to rigid, preset structural categories (e.g., detective story, news story, or magazine ad). Genre theory gives me the meta-language to describe the community actions that formulate the flexible rules in the new workplace, giving me a framework I can use to describe the various elements I see in contemporary documentation practices—traditional documents, plus oral and other informal communication practices. Moreover, the theories aid me in describing my documentation research experience as a professional.

In this dissertation, I highlight how genre theory can describe more than merely written texts; my objective is to show how oral and other rhetorical forms are missing pieces in what can be accurate descriptions of contemporary practice. To that end, Berkenkotter and Huckin (1995) used the words “rhetorical form,” instead of words like “written” or “text,” to conceive more than written communication: “Genres are dynamic rhetorical forms that are developed from actors' responses to recurrent situations and that serve to stabilize experience and give it coherence and meaning” (p. 4). By dynamic rhetorical forms, Berkenkotter and Huckin meant that communities decide on strategies of communication and reconfigure written communication as needed. The key to genre theory is there are not inflexible, predetermined parameters for genres. Rather, communities determine the parameters of communication, based upon the needs of the community. Insofar as communities must frequently make decisions about rhetorical form, Berkenkotter and Huckin called those decisions “recurrent situations.” Genre forms are organic; each time the community responds to a recurrent situation, the community renegotiates form and content. Communities reuse the solutions that work and that is what Berkenkotter and Huckin meant by stabilizing experience—a genre is simply a pattern of solutions. Contemporary software developers use genre forms to stabilize their recurrent situations.

Applying genre theory to software developers. My research suggests software industry documentation guidelines need to describe documentation practices so they include both written communication and oral communication acts. The problem with contemporary software developers is the surprising dependence on oral communication. By the standards of industry guidebooks and the predictions of genre theory, the absence of written communication means that key development decisions are made off-stage—or outside the documentation process. However, if oral communication is in fact a rhetorical form then development decisions are no longer off-stage.

I originally and erroneously assumed that something was either wrong with the software developers or with genre theory. My mistake, however, was using the theory to examine writing practices when the communication of many contemporary developers was no longer limited to writing. Developers were using oral communication to meet their objectives. It was clear that I needed to expand my examinations to include a full spectrum of communication.

My Research Problems and Software Development.

Contemporary developers are an excellent model for testing genre theory; they continually adapt their plans and rewrite their internal documents. However, where genre theory predicts that the community should harness a cycle of documentation to formulate decisions, the contemporary community does not. Documentation is often an activity they do after all the decision making is done. For instance, Brad is one of the Salt Lake City developers I interviewed; he claimed that only if his team lived in a world where written communication was the only way to communicate would they only rely on written documents to make decisions and communicate. He suggested that there are other ways—other rhetorical forms—that facilitate communication and design decisions. Therefore, contemporary developers chitchat in weekly, undocumented meetings, rather than use rigid documents to govern projects, transmit design direction or sustain team unity.

The cycle of documentation varies according to specific software development methods. These methodologies differ a great deal on the preplanning and the documentation developers require. For instance, chitchat is not conducive to preplanning requirements. The “Waterfall” methods and “Agile” methods are the chief software development methodologies.

- *Waterfall development* is built on the principle of a tiered waterfall, in which water rushes off several discreet, linear stages. The belief is that if the team plans sufficiently then the project will flow smoothly with its own momentum. In addition, the Traditional method values preplanning.
- *Agile development* is built on the principle of recursive cycles, in which a prototype is repeatedly presented to the client. Developers believe that the existing plan should change as required; they value the flexibility of their development method.

While the purpose of traditional documentation suits traditional development, the purpose does not suit contemporary development. The emphasis on documentation makes traditional development optimal for genre analysis; traditional development environments are clear recursive situations with responsive agents. While contemporary developers have documents that help teams negotiate recursive situations, contemporary developers change the document with each recursive situation; the document does not stabilize contemporary developers. Some other rhetorical form performs that purpose.

Traditional software development methodology. Traditional development has four chief steps (Larman, 2003). The steps begin with an enormous preplanning effort that thoroughly defines as many elements as possible (p. 57):

1. Define up front, in detail, the requirements.
2. Define the "design" (text and diagrammatic descriptions of the software and hardware elements.
3. Implement the system (programming, and so forth).
4. Integrate and test the components.

The Traditional method's enormous preplanning requires defining, elaborating, diagramming, graphing, and stating projected specifications. This network of documentation excites Genre theorists who seek to see how genres shape recursive situations and how agents respond to those rhetorical forms. Therefore, in traditional development, a document detailing the projected specifications will guide developers who all work on different components—necessitating a unified document.

Contemporary software development methodology. However, contemporary developers know preplanning and documentation do not govern development. In fact, Hudson is one of the Salt Lake City developers I interviewed; he stated that he once did traditional development and knew that even traditional developers did not document like they said they should. Therefore, documentation might perhaps form from actors' responses to recurrent situations but oral communication is what actually stabilizes the situation. Larman (2003) suggests it is not possible to "define [contemporary] methods, as specific practices vary"; yet, Larman still identifies some basic practices: "Timeboxed, iterative and evolutionary development, adaptive planning, promote evolutionary delivery, and include other values and practices that encourage agility" (p. 25). Contemporary values are diametrically opposed to Traditional values. In fact, Andreas Rüping (2003) echoes the Agile Manifesto (p. 1):

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The agile manifesto highlights the small role of documentation in contemporary development.

Genre theory provides a meta-language that can describe the documentation industry guides recommend. For instance, actors' situated decisions are supposed to be typified rhetorical actions that generate documentation. Yet, contemporary developers do not document well or do not document at all. Their documentation is more of a legacy record than a product of situated decisions. As I expand genre theory to highlight the rhetorical forms contemporary developers use to make decisions, idle chitchat turns into the oral communication that stabilizes contemporary development.

DISSERTATION PURPOSE AND OBJECTIVES

Genre theory is useful for describing recursive situations like the documentation in software development practices. However, contemporary developers negotiate decisions without recursive documentation, even if they scrawl a legacy document at the end of each development cycle. The result is a development practice with no notable written documentation practice. However, written documentation is not the limit of documentation activities. Contemporary developers do in fact have an as yet undefined documentation practice. My objective is to identify the rhetorical forms on which contemporary developers rely for their rhetorical situation. Genre theory works as a meta-language to describe the rhetorical forms in contemporary development.

Whereas the plan-driven methodology of traditional documents easily lends itself to genre theory, the same document methods do not fit in a description of a flexible and contemporary methodology. Along with the methodology, software developers changed their values too, as they adapted to more flexible projects. Where they should have adapted the way they performed documentation activities too, they did not. This amounts to describing the purpose of a genre in the wrong situation.

The decisions that propel contemporary software development are not often written. Consequently, by the standards of traditional methodology, contemporary development's undocumented decisions spell out unguided, poorly designed projects destined for ruin. The rapid cycles of development are what save contemporary projects. Recursive decision-making is so quick that oral communication seems sufficient to sustain undocumented activities on a project. In addition, the social situation transforms so quickly that developers often cannot act on patterns of recursive writing. For instance, whereas traditional developers need documents to easily recycle software assets, contemporary developers change things so frequently that there often is not anything to recycle. Consequently, contemporary developers deploy the rhetorical form of oral communication to sustain progress in their community.

Risks Arising from Oral Communication

There are some potential risks in a development practice that operates on the strength of oral communication. Traditional documentation is supposed to be a record of official communication to which people can refer and on which people can rely, rather than chitchat in the elevator. In fact, many developers do not believe a conversation counts as a medium for decision-making because decisions need to be mapped out and articulated in writing.

Written documentation has an important place in the professional workplace if for no other reason than to have a record of decisions in writing. The standard phrase: "Can I have that in writing?" provides a safety net when developers do not agree on a decision's exact content. Dorothy Winsor (1999) observes this safety net when she conducts a wrap up of a 5-year longitudinal study interviewing engineers about their documentation practices. Winsor interviewed an engineer who identified the importance of documentation: a paper trail of documentation is protection. Put simply, Winsor's subject explained there are two experiences a person needs for protection before writing: 1) getting burnt 2) burning someone else (p. 209). Consequently, the importance of documentation is significant when decisions have a long-term lifespan. In contrast, the shorter lifespan of contemporary decisions impacts development differently.

Oral communication cannot be a comprehensive replacement of documentation but the absence of documentation does not equate the absence of communication. The contemporary developers may or may not write down decisions but that does not mean the communication did not happen or that there was a failure of documentation—they just did not have time to write it. Consequently, I plan to demonstrate that contemporary development still bears the rhetorical forms that genre theory requires and that genre theory can adequately describe the oral and written communication in contemporary development environments.

DISSERTATION PROCEDURES

Genre theory provides both a way of conceptualizing the situations, as well as a schematized method of describing the situations. In addition, the meta-language of genre theory helps me describe documentation activities that are not limited to written communication. I use genre theory as both a conceptual framework and meta-language with which I can describe the two research situations I have selected, as well as the interviews I have conducted.

Theoretical Dissertation Procedure

Of the many authors who promote genre theory, I find Berkenkotter and Huckin's (1995) *rhetorical forms* to be the most useful for describing oral communication. Not only are *rhetorical forms* less restrictive than the written texts other researchers favor but Berkenkotter and Huckin break those rhetorical forms down into specific principles. The authors identify five principles by which I can describe genre features and community actions.

- Dynamism
- Situatedness
- Form and Content
- Duality of Structure
- Community Ownership

I have broken down the five principles into 11 value statements. These value statements constitute a Model of Expectations with which I will describe each of the three software development situations.

I also use a second model to assess how well the documentation I find actually works in each of the research situations. In his forthcoming book, David E. Hailey (2014) uses genre theory to identify the exigencies, urgency, purpose, audience, rhetorical stance, and structure of online genres. His EUPARS rubric assesses how appropriate a particular document is for the particular situation. While EUPARS is useful to describe how traditional documents are not appropriate in my first three contemporary situations, the model will also be useful in the final analysis of oral communication.

Dissertation Research Situation Procedures

I use the Model of Expectations for three research situations. Two of the situations are Postmortems of research projects in which I was closely involved. In an effort to triangulate my work as a participant observer, I utilize a third research situation, in which I interview seven Salt Lake City developers. Finally, I use the EUPARS model to verify that the rhetorical forms are appropriate in contemporary environments — among those rhetorical forms, oral communication is key to describe contemporary development with genre theory.

American west heritage center (AWHC) spring 2009. Under the faculty supervision of Dr. Brett Shelton, I served on one of three contemporary teams developing a tour simulation of Logan Utah's historic American West Heritage Center. My team produced documentation much like Owens suggested.

I was one of four graduate students on one of three teams. Dr. Shelton closely directed the aggressive, contemporary project. My team maintained all documents on Google Docs. In this way, we could update the same document in real-time, even if we were each working in remote locations. I have extensive documentation samples to represent real-time development. I planned a formal conversation, along with three talking points, with each of my teammates. They each had very different previous experience with documentation. However, they each came to the same conclusions about our project's documentation.

This situation will be a chapter that illustrates what I thought was successful, contemporary documentation. However, the meta-language of genre theory helps me articulate why this was not actually the case and why I tried to do it again.

USU engineering department summer 2009. Under the faculty supervision of Dr. David Hailey, I coordinated filming and development of an online, modular interface for engineering students at remote Utah State University campus locations. The team worked in such close proximity in a fast paced contemporary development process so that the absence of documentation was insufficient to account for the success of the project.

As the project coordinator, I was responsible for coordinating a film crew with four engineering courses. I coordinated the participation of six Engineering faculty, in addition to processing, editing, and converting film to Flash learning modules.

Strictly speaking, the project did not generate any internal software documentation. I can use the meta-language of genre theory to articulate why it became clear that there was another rhetorical form at play and that the rhetorical form was oral communication.

Local developer interviews spring 2012. A local Salt Lake City, Utah software company has a small team of seven developers. The team leader led a very contemporary style of software development. The team leader related to me his difficulty trying to complete documentation the way he is supposed to do. I interviewed all seven team members during seven individual lunchtime appointments.

I matched an open-ended interview question to each of the value statements in the Model of Expectations. I interviewed the seven and will use my set of interview questions to assess the Model of Expectations in his situation. I expect the rhetorical form I find will be oral communication. I follow the interviews with Hailey's EUPRAS Model to assess the appropriateness of oral communication in a situation that traditionally demands documentation.

CHAPTER OVERVIEW

The following is an outline of my dissertation’s six chapters.

Chapter 1 introduction. I strengthen the exigency of oral communication and genre theory in contemporary development by identifying the gap in contemporary development practices and the utility of genre theory to describe the gap. I introduce rhetorical forms of genre theory, rather than the written communication to which North American genre theorists typically refer. Particularly, I mention the utility of oral communication as a rhetorical form in contemporary development practices.

Chapter 2 literature review. The literature review define the principles of both genre theory and software documentation. I outline the meta-language of genre theory and articulate how it is a meta-language. I support both the Model of Expectations and the UEPARS model with that meta-language. The central objective is to demonstrate that oral communication is a rhetorical form that fits in the meta-language of genre theory research.

Chapter 3 methodology. I drew on the literature review to support both the Model of Expectations and EUPARS Model. I detailed the postmortem projects and the manner by which I gathered samples. I also add more detail about the interview questions and how I drew the questions from the Model of Expectations.

Chapter 4 postmortems: american west heritage center

I have extensive documentation archived for this project. I also have notes from three conversations with my development team peers. The organic documents my team maintained were an achievement of practical experience and good documentation practices in contemporary development. At the time of the project, I felt the project was an example of documentation best practices. However, I did not count on my inability to replicate the success in my next project.

Chapter 5 postmortems: engineering modules project. This Engineering Modules project was serendipitous because of the absence of documentation; however, genre theory can only account for how we were successful if I describe rhetorical forms like oral communication, rather than judge the project a failure. This chapter's main objective is to support oral communication as the central rhetorical form, as well as an important research topic in the field of professional communication.

Chapter 6 local developer interviews. The interviews will be the final chapter of the dissertation. I interviewed seven senior Salt Lake City software developers. I used interview questions extracted from my Model of Expectations. I wanted to challenge their professional practice by getting them to talk about the absence of their documentation. I suspected they would articulate the rhetorical forms they use instead. As expected, the developers described their oral communication as the rhetorical form that sustains their practice, in addition to all their other rhetorical forms.

CHAPTER II

LITERATURE REVIEW

EXAMPLE OF DOCUMENTATION THEORY AND PRACTICE

An important aspect of this literature review is the distinction between traditional and contemporary development. In addition, references to traditional documentation refer to the documentation standards and practices prescribed by traditional development. In the past, software developers used a documentation model that required they describe all of the steps the finished program would contain before the developers actually began the process of programming. This was the first step of the traditional programming model to which I refer—commonly referred to as the "waterfall model." Although much contemporary programming—models like Agile and Scrum—no longer involves the waterfall model, developers still use the dated and irrelevant documentation processes. In part, the reason they continue using the old model is because few have examined possible alternatives. This chapter presents that problem and posits alternatives.

A Short Narrative

In an effort to ground the literature review, I begin this chapter with a short narrative. Where traditional documentation standards and practices are not useful to contemporary developers, the following narrative describes a contemporary developer trying to make traditional documentation work in a contemporary work environment. I hope the narrative adds a level of relevance to the meta-language necessary to describe a documentation practice that incorporates rhetorical forms, in addition to written documents.

In 2009 I made a worksite visit to an Orem, Utah computer game development company I call Edgetech. I immediately noticed that the Edgetech used traditional methods of documentation, rather than the liquid documentation practices contemporary development demands. In fact, they used a template, downloaded from the Internet, to model their entire design documentation procedure. That template carefully listed every heading one might expect in a traditional design document and prescribed the content for each heading. That practice was completely inappropriate for their contemporary programming model.

Profile of edgetech. The company was a new start-up that began in 2005. In the four years before my site visit, it had developed its flagship PC computer game. The development studio continuously updated the game and administrated the servers that ran the game's persistent Massively Multiplayer Online Real-Time Strategy (MMORTS) game world.

The developers had converted a small home on the outskirts of Orem. The living room was their shipping room. The master bedroom was both the CEO's office and the storage room. The remaining two bedrooms were for the developers and the graphic designers respectively. There were wires, cables, and power strips hanging on the walls, along with large promotional posters for their game.

The market for the game was a cross between card gamers, battle strategy gamers, and online gamers. Troops, weapons, buildings, and resources were all cards the player acquired or bought in booster packs from card stores. Their game was available for digital download or mail order from retail web sites. The company was modestly successful but was looking to expand its business. When I met with the CEO, he was preparing a high concept document for presentation to investors in the Chinese market.

Disconnect Between CEO and Development Team

The bedroom in which the developers were set up had all the computer terminals in the middle of the room. Where the graphic designers each sat on their respective sides of a rectangle, the developers were sharing monitors and rolling their chairs back and forth between terminals. When I walked through, graphic designers were seated at their stations, whereas four developers were huddled together looking over the shoulder of a fifth programmer—they were collaborating about the code.

The company’s CEO was writing very thorough documentation, according to traditional industry standards. The CEO was even using a template to insure he met the industry standards. However, that documentation was incompatible with the contemporary development standards of the developers down the hall. They were on their feet problem-solving at a computer workstation. They were not studying the CEO’s documented design details. Nor were they following the comprehensive development plan. Rather, they made changes on the fly; yet while the changes necessitated changes to the documentation, they did not change the documentation.

Reasons for the Breakdown

Traditional development methods require a lot of pre-planning and need a great deal of forethought and oversight to stay on track. The Orem developers were not in a large-scale project and needed to work using an agile model with the flexibility to make quick enhancements. The developers used different standards from the CEO's because their needs constituted a completely different situation. However, even while the programming community and situation were different, they still sought to produce documents that facilitated pre-planning and rigorous organization.

In the end, the Orem developers were all clustered around a single monitor, still seeking to stuff their contemporary needs into traditional documentation. The developers were all standing there conversing about design decisions – in effect, ignoring the documentation so carefully prepared to guide them through the programming process.

Genre Theorists and Programming Documentation

Genre theorists efficiently describe traditional documentation in software development; however, when I use their meta-language to predict and describe documentation in contemporary practices, I do not find the documentation they describe. Instead, I find forms and genres that are no longer useful to contemporary developers.

Carolyn Miller (1994) argues genres are typified rhetorical responses in recursive situations; in the parlance of professional communication research, Miller's typified rhetorical responses are usually represented as written communication. In fact, although theorists might agree that oral communication is a typified rhetorical response, researchers tend to class oral communication as a genre of idle chitchat that is not serious communication. In the case of contemporary software development practices, failing to see the importance of oral communication makes it impossible to describe many rhetorical forms contemporary developers apply to their programming practices.

REVIEWING REAL IMPACT ON PROFESSIONALS

The Orem developer is not the only development shop in the world with documentation problems. In fact, contemporary developers at large experience similar frustrations and barriers with documentation practices. Those same frustrations are often expressed in articles published in issues of *Game Developer* magazine. The magazine targets game development professionals and every issue features a project postmortem article from a different professional development studio. Austin Grossman (2003) compiled 25 of *Game Developers* postmortems; his selections feature seminal games and key developers in the video game industry. Each of the 25 developers share the things that went both right and wrong in a short, summative article. For instance, many postmortems identify documentation as one thing that went wrong.

Impact Reported in Software Development Postmortems

A notable aspect of what these professionals count as successful documentation is the planning and vision good documentation seems to bring to a project. In the success stories, developers tout a tremendous amount of effort toward predefining the game and setting a course for the game's development. In contrast, *all of* the developers prize the organic way the design, direction, and team dynamics all adapt to the needs of the development project. For instance, Grossman features the postmortem of *Cel Damage* (Barrett, 2003). Of cutting assets from the game's design, Barrett states "If such changes created holes in the game's design, we would be flexible and design around the holes" (p. 46). So while on the one hand developers subscribe to a model of traditional documentation, when faced with the problem of programming in the real world, they ignore the documentation's prescriptions.

Successful professional documentation. Warren Spector writes his postmortem about the development of *Deus Ex* (2003). One of the strengths behind the project was a high level of vision, due to six months of preproduction. Spector reports: "we had 300 pages of documentation and thought we knew everything we'd needed to know to make a game. Were we ever wrong …that 300-page document mushroomed into more than 500 pages" (p. 198). The *Deus Ex* documentation is the spectacular result of focused preproduction efforts. Of course, Spector reports one of the project's key strengths was "recognizing that game design is an organic process" (p. 198). Spector lists nine design elements the team changed; "the game benefited, but this was a radical change from the original plan" (p. 200). Consequently, the two great strengths of the game were the comprehensive documentation and their willingness to adapt the design documentation in nine game changing ways.

Weak professional documentation. Grossman's book contains plenty of postmortems that detail weak documentation practices. These examples are the postmortems of very competent games made by very adaptive, creative developers; however the examples confess poor documentation. I highlight the competence of the game developers to suggest that there was an additional (not mentioned) rhetorical form that sustained meaning-making and stabilized the team's efforts, even if written documentation did not.

Toby Ragaini’s postmortem (2003) identifies the greatest weakness of *Asheron’s Call* was the “inexperienced development team” (p. 307). Their employees were often students on work-study at the local university. Where those inexperienced developers probably needed documentation more than others, Toby Ragaini reports: “The technical design document process and high-level feature overviews were basically skipped. This created severe problems, when it came to prioritizing which features were important. We constantly had to justify features, and we had no documentation to fall back on to resolve our discussions” (p. 309). Ragaini’s inexperienced students developed a successful game but Ragaini does not identify the rhetorical form that did hold the students together.

Brian Upton (2003) describes the development of a reliable game license in the industry by a mature development studio—*Rainbow Six* (1998). Upton’s postmortem demonstrates experience does not always equate sound documentation practices. Upton plainly states the chief weakness was the team “never had a proper design document, which meant that we generated a lot of code and art that we later had to scrap. What’s worse, because we did not have a detailed outline of what we were trying to build, we had no way to measure our progress” (p. 257). Upton’s admission identifies a development problem that seriously impacted the project. The reference to scrapped code is particularly expensive for developers because a few clear sentences written in a few short minutes could have saved hours of unneeded coding. Regardless of the suggested dysfunction and disorder, the team still shipped a competent and competitive computer game.

REVIEW USAGE OF META-LANGUAGES

When developers use the word “document” as a verb, they refer to a kind of activity or a phase in the development workflow. As a result, documentation is a development activity characterized by written documents; it is the only label for the activity. Consequently, a discussion about the methodological label “documentation,” without highlighting anything written, is nearly impossible. It is like talking about writing in a way that does not involve writing. Fortunately, there is another way to analyze documentation.

Genre theory provides a meta-language. That meta-language has it’s own set of rules and vocabulary. In addition to supporting the use of a meta-language, I detail the principles of genre theory. In addition to the activity of “documentation,” a meta-language highlights the language limitations of written communication in professional communication research. I want to demonstrate that a language about a language is the first step to understanding alternatives to writing documents in the activity of “documentation.”

Meta-Languages and Knowledge

The key element of a robust theoretical model is the vocabulary that it provides. Software developers produce plenty of industry guides that describe the rules of the documentation genres. However, they use the vocabulary of traditional documentation to describe traditional documentation. They need a vocabulary that permits them to examine their current language—a meta-language. A meta-language is simply an objectively positioned language above another language. While writing about language frameworks, Foss, K.A., Foss, S.K., and Trapp, R. state: (2003) "Thoughts and ideas are never free from the language that is used to frame them" (p. 199.) Foss, et. al suggest that writers who describe traditional documentation might never figure out why documentation does not work in contemporary development, as long as they use the same language to figure out the problem. Industry writers need a meta-language outside the community of software documentation.

Similar to the statement of Foss et al., Friedrich Nietzsche (1914) takes a strong position on the relationship between language and knowledge: "We cease from thinking if we do not wish to think under the control of language" (p. 38). Consequently, we can only think about things we have the language to describe—we have a language-limited knowledge. Thomas Kent (1986) sums up where Nietzsche believes people obtain their language to describe their knowledge.

> First Nietzsche understands language to be a thoroughly social and historical phenomenon; second, Nietzsche claims that rational thought derives from our use of language, and not the other way around; and third, he conceives of rhetoric as the process we employ in order to construct meaning and, consequently, to construct our knowledge of the world we share with others. (p. 9)

The relevance of Nietzsche to software development documentation lies in the industry writers who seek to describe traditional documentation for contemporary developers. As long as they continue to use the same traditional vocabulary to outline the purpose, situation, and community of software documentation they limit what they can know about documentation's application to contemporary situations and communities.

Researchers like Carolyn Miller and Russell Rutter agree that knowledge conforms to language; however, they also write a little more about what that kind of language-limited knowledge looks like. Carolyn Miller (1979) states: "Reality cannot be separated from our knowledge of it; knowledge cannot be separated from the knower; the knower cannot be separated from the community...facts are human constructions which presuppose theories" (p. 612). Industry guide recommendations for design documentation might be straightforward but such informational data emerges from a community knowledge system. Russell Rutter (1991) describes knowledge in terms of science. Rutter suggests that science is not merely a product of facts but is also a product of mindsets, expectations, and paradigms (p. 142). In other words, Rutter suggests the software industry's design documentation complies with mindsets, expectations, and paradigms.

Genre theory offers a rich meta-language with which I can describe documentation practices, without getting trapped in the same vocabulary mindsets, expectations, and paradigms as the industry developers.

Burke's Rhetorical Grammars

There have been traditional software development methodologies since the 1960s. It might seem unreasonable for five principles to retain relevance for 54 years of computer science history. Fortunately, Kenneth Burke demonstrates that he can use a set of terms to analyze and describe rhetorical motives in a broad variety of situations. Burke's *A Grammar of Motives* (1945) tackles the question of analyzing, describing, and understanding motive in his own meta-language system—Dramatism. For Burke, rhetors can use Dramaticism to describe motive and persuasive moments much like a viewer can describe a play or a movie. His grammar is also called Pentadic Criticism; the Pentad has five grammars: act, scene, agent, agency, and purpose. The pentad is five key descriptors that help rhetors code the drama-like communications between people. The grammar acts as a reliable system that rhetors can use to describe a communication and assess the motives.

> You must have some word that names the act (names what took place, in thought or deed), and another that names the scene (the background of the act, the situation in which it occurred); also you must indicate what person or kind of person (agent) performed the act, what means or instruments he used (agency), and the purpose. (p. xv)

Burke posits that motive is too complex for either a set of five discrete grammars or strict definitions. Rather, the power of the system is the way pair combinations (or ratios) make for a dynamic Pentad. For instance, Burke states the scene-agent ratio pairs people and things; a brutal scene indicates brutal agents (p. 7). Consequently, human motivation is a complex combination of the act, agent, agency, scene, and purpose.

Burke (1945) sets up the precision of his Pentad in the first pages of *A Grammar of Motives*. The remainder of the book's first part is mainly elaboration of the different ratios. However, part two of the book is where Burke divides the Pentad into schools of thought and demonstrates the application of the Pentad in each school (p. 128).

- Scene=materialism
- Agent=idealism

- Agency=pragmatism
- Purpose=mysticism
- Act=realism

Part two works through the schools of thought to demonstrate the versatility of the Pentad's ratios. Burke provides several applications of his Pentad in each school of thought. However, there are many more applications of the Pentad.

Adapting Burke's grammar to McAllister's gameworks. One application for Burke's grammar might be a video game. However, computer game motives pose a problem because in a computer game the audience is part of the drama, rather than simply detached spectators. When the audience is the agent, a different set of grammars, based on Burke's pentad, can be useful. Ken McAllister's (2004) grammar of gameworks is a set of five grammars that analyze the audience as part of the drama.

Yet again, a meta-language may seem inadequate to describe an entire industry and audience motives in general. However, in *Game Work* (2004), Ken McAllister seeks to describe computer games as a complex cultural system that is larger than issues of value systems, indoctrination or vacant stares. He uses Burke's pentad to develop a grammar of gameworks that lets scholars describe games without losing track of the entire system—much like the pentadic grammar. There are five grammars of gameworks: agents, functions, influences, manifestations, and transformative locales. Differences between the two grammars are apparent as McAllister's grammar becomes more focused on understanding the role of the player as both an audience and an agent. In fact, the McAllister's agent also includes the game avatar, in addition to the synthetic identity of the player/avatar. Consequently, the grammar of gamework is crafted to be a pentad-like critique of a complex computer game culture.

Grammar is a key notion for McAllister's gameworks; by grammar, McAllister refers to a set of terms that formulate an assessment matrix useful in understanding the complexity of the computer game culture. McAllister describes the grammar of gameworks as "one way to make meaning out of an artifact like a computer game" (p. 1). For McAllister, the grammar of gameworks is a cultural system with identifiable elements and can describe design practices, developer collaborations, fan sites, specification improvements, and even the onscreen avatar itself—all parts of a complex cultural system.

Much like McAllister and Burke, I will use a meta-language too. Burke's Dramatistic Pentad outlines persuasive motivations and McAllister's grammar of gameworks is a "multiperspectival" approach to studying a socioeconomically complex game industry. I use the principles (grammars) of genre theory as the meta-language of documentation practices in software development.

Tenets of Genre Theory as Meta-Language

Rather than the static boundaries of Sci-Fi Thrillers and Film Noire, genre theory posits genres as social action, rather than sets of categories. Carolyn Miller (1994) coined genre theory's seminal definition: it is the "connection between genre and recurrent situation and the way in which genre can be said to represent typified rhetorical action" (p. 151). Those recursive actions are rhetorical actions because human agents validate decisions again and again. Typified rhetorical actions are patterns of human decision and activity. In this way, genre rules are simply decisions that endure as communities comply with their genres.

Genres are not an immutable structures of letters and words; they are community actions. Thomas Kent (1993) advocates a shift in thinking away from structures of letters and words: “When we move away from a conception of communication grounded in the word and the sentence and move toward a conception of communication grounded in genre, both the production and the reception of discourse appear in an entirely new light” (p. 128). As opposed to the unique jargon, typical phrases, headings and content requirements, genre researchers suggest communication should be grounded in the purpose, situation, and community. Even more specifically, communication should be grounded in both the recurrent situation and typified rhetorical action.

Carolyn Miller (1994) was the first to define genre as typified rhetorical action and avoids restricting genres to written communication as well. In fact, she states, “I will be arguing that a rhetorically sound definition of genre must be centered not on the substance or the form of discourse but on the action it is used to accomplish” (p. 151). Instead of something like written software documentation, Miller prefers thinking of the rhetorical actions as the substance of a genre. Insofar as there is much more than written communication (i.e. the horror movie, pop music, first person shooter video games), those rhetorical actions that substantiate genres are a more flexible way to think about genre.

Meta-language of north american genre theory. Berkenkotter and Huckin (1995) reference “rhetorical forms,” rather than rhetorical actions. However, unlike Miller’s rhetorical actions, Berkenkotter and Huckin locate their rhetorical forms in five principles

Table 1

A Framework of Genre Theory

Theoretical Term	**Genre theory Definition**
Dynamism	"Genres are dynamic rhetorical forms that are developed from actors' responses to recurrent situations and that serve to stabilize experience and give it coherence and meaning. Genres change over time in response to their users' sociocognitive needs." (p. 4)
Situatedness	"Our knowledge of genres is derived from and embedded in our participation in the communicative activities of daily and professional life. As such, genre knowledge is a form of "situated cognition" that continues to develop as we participate in the activities of the ambient culture." (p.7)
Form and Content	"Genre knowledge embraces both form and content, including a sense of what content is appropriate to a particular purpose in a particular situation at a particular point in time." (p.13)
Duality of Structure	"As we draw on genre rules to engage in professional activities, we *constitute* social structures (in professional, institutional, and organizational contexts) and simultaneously *reproduce* these structures." (p.17)
Community Ownership	"Genre conventions signal a discourse community's norms, epistemology, ideology, and social ontology." (p. 21)

of genre, with clear labels. In fact, they identify five specific principles that define genre theory: dynamism, situatedness, form and content, duality of structure, and community ownership. Table 1 identifies the definitions Berkenkotter and Huckin gave for each of the principles. I prefer the ease of descriptive principles with which I can describe Miller's typified rhetorical action. Berkenkotter and Huckin's genre principles (Table 1) have a taxonomical benefit on the way I have come to understand internal documentation.

There is clear opposition to a taxonomy of genre (Askehave & Nielsen, 2005; Kent, 1986; Miller, 1994) in the research literature. For instance, Carolyn Miller (1994) argues against taxonomies because of the flexibility of genres; at the same time, Miller identifies that there is a function that taxonomies can serve. She argues: "The understanding of rhetorical genre . . . does not lend itself to taxonomy, for genres change, evolve and decay . . . [however,] it can provide guidance in the evaluation of genre claims" (pp. 36-37). At the same time, Miller does identify five genre features (p. 163) that are similar to those defined by Berkenkotter and Huckin:

- Genre refers to a conventional category of discourse based in large-scale typification of rhetorical action; as action, it acquires meaning from situation and from the social context in which that situation arose.
- As meaningful action, genre is interpretable by means of rules; genre rules occur at a relatively high level on a hierarchy of rules for symbolic interaction.
- Genre is distinct from form: form is the more general term used at all levels of the hierarchy. Genre is a form at one particular level that is a fusion of lower-level forms and characteristic substance.
- Genre serves as the substance of forms at higher levels; as recurrent patterns of language use, genres help constitute the substance of our cultural life.
- A genre is a rhetorical means for mediating private intentions and social exigency; it motivates by connecting the private with the public, the singular with the recurrent.

Miller neither labels nor elaborates her five features. They are characteristics or points of reference useful to those who evaluate genre claims. Where Miller offers guidance, Berkenkotter and Huckin suggest principles that assess and describe genres. I can use those principles of genre as a framework to describe documentation practices.

A meta-language of genre needs purpose. A notable theme in the genre theory literature is that "purpose" is a key element that constitutes a rhetorical form. In fact, Miller (1994), Berkenkotter and Huckin (1995) all identify purpose in their description of genres. Miller writes: “But at the level of the genre, motive becomes a conventionalized social purpose, or exigency, within the recurrent situation” (p. 162). Miller highlights the fact that nothing can be rhetorical without a motive—without a purpose. Berkenkotter and Huckin identify purpose as well: “analysts should pay attention to ways in which genre users manipulate genres for particular rhetorical purposes” (p. 2). The authors imply motive when they mention rhetorical manipulation; motives and manipulations are impossible without a purpose. In fact, Kenneth Burke (1950) suggests agents involved in something like a typified rhetorical action must have a purpose because persuasion is an attitude that involves free choice (p. 50).

Yates and Orlikowski (2002) take the importance another step and support a key formula for genre theory: “The purpose of a genre is not an individual's private motive for communicating, but a purpose socially constructed and recognized by the relevant organizational community for typical situations” (p. 15). In other words genres are purpose constructed by a community in a situation. Concepts like motive, persuasion, manipulation, choice, and typified rhetorical actions are empty without purpose, community, and a situation. Consequently, I have derived a genre theory formula:

Genre Theory=Purpose+Situation+Community

Five Meta-Language Principles to Describe Genres

While a traditional genre might be best known as a set of rules for a specific kind of text, genre theory suggests that communities of human agents invent and adapt those specific rules; therefore, written communication is a social act. Berkenkotter and Huckin

(1995) state: “[rhetorical forms] function within disciplinary cultures to facilitate the multiple social interactions that are instrumental in the production of knowledge” (p.1). In fact, communities become static without a document that facilitates social interaction. Consequently, rhetorical forms are an influential force in a community because “genres are the media through which [communities] communicate with their peers” (p.1). Berkenkotter and Huckin outline a framework that describes how communities invent and adapt rhetorical forms. The framework of genre theory has five principles: dynamism, situatedness, form and content, duality of structure, and community ownership.

The meta-language of dynamism. Genre possesses dynamism because of how the writing emerges from an “actor’s responses to recurrent situations” (p. 4). The responses are essential to dynamism because responses are part of a stabilizing struggle wherein actors and situations create “coherence and meaning” (p. 4). With each recursion, the actors validate meaning-making decisions. At some point, enough recursive actions reduce variance until here is a stable genre. Actors repeat situations until the pattern clearly marks the meaning—patterns of meaning emerge from the recursive acting.

In addition, the recurrent situation formulates specific vocabulary and style that lend to the form and content of the genre. However, the recurrent situation is a social adaptation to actors, rather than simply vocabulary, typical phrases, and style. The dynamic adaptation conforms to the responses and actions so that a rhetorical form matches the needs of the actors.

The meta-language of situatedness. Berkenkotter and Huckin (1995) suggest knowledge is the key element of situatedness; the community knowledge of which Nietzsche writes is a situated knowledge. In other words, every time actors transmit knowledge they situate both the knowledge and themselves within a meaning-making community. In short, actors have a situated cognition bound by their community's needs.

In addition, Berkenkotter and Huckin (1995) suggest situatedness refers to "sociocultural navigation" (p. 118). In other words, situations do not come with a script by which every actor understands the rules and strategies; on the contrary, situations are sites of negotiation between actors who form situated knowledge together. By this manner of persuasion do actors assimilate themselves into the situation and perform the enculturation of new entries. Rhetorical forms help stabilize the knowledge and situation to which new entries must assimilate themselves.

The meta-language of form and content. Form is the appearance, presentation, organization, and sequence of a rhetorical form. Content is the topic, details, and politically correct office phraseology of the rhetorical form. Form and content are not abstract principles like dynamism or duality of structure; they are tangible. Berkenkotter and Huckin (1995) claim the impact of form and content is tangible because "genre and genre knowledge are more sharply and richly defined to the extent that they are localized (in both time and space)" (pp. 13-14). Despite the meaning-making complications of dynamism and situatedness, the rhetorical forms have a physical form and content that stands as a material location in the community.

The form and content of a genre demands vocabulary, typical phrases, style, and even the organization of a rhetorical form. These conventions are the product of actor's recurring performances and their situated knowledge, rather than templates or categorical rules.

The meta-language of duality of structure. Duality of Structure is the category that conceptualizes how something like a design document can be both complete and yet not static. Berkenkotter and Huckin (1995) describe duality of structure in the following way: "we *constitute* social structures (in professional, institutional, and organizational contexts) and simultaneously *reproduce* these structures" (p. 17). Duality of structure relies on a close relationship between social structure and human action, rather than clear separations that make human action free of the social institutions that influence human agency. According to the authors: "human agency and social structure can be seen to be implicated in each other rather than being opposed" (p. 18). All told, the human participation simultaneously constitutes and reproduces social structure; the result is a document reflexive with situated knowledge and a meaning-making community.

Duality of structure is a key term for genres of software documentation; duality of structure describes the design document as a site that stabilizes the community. At the same time, the design document reconstitutes the community. Traditional and Contemporary software development might accept both stabilizing and reconstituting documents at the same time but only traditional developers have the documentation to show for it. Contemporary developers adapt quickly to new consumer requirements and test results. That means the project changes too often to document anything; either than or they employ other rhetorical forms to stabilize and reconstitute their community. They certainly do not let a document inform the decisions that drive the flexibility of their project organization.

The meta-language of community ownership. The discourse community owns the norms, epistemology, ideology, and social ontology of the community. With that ownership comes the actors who participate and sustain the recurrence of situations, as well as the constitution and reproduction of those situations. Actors draw on both situated knowledge and the consequent rhetorical forms to both support their value system and formulate their value system. The community discards localized conventions that do not meet the criteria for ownership. Berkenkotter and Huckin (1995) state ownership has to do with the "ways in which the genres of [writing] function to instantiate the norms, values, epistemologies, and ideological assumptions of [cultures]" (p. 22). This last description of community ownership clarifies the term's function. As actors respond to recurring situations, the community that constitutes and reproduces those situations also claims various conventions or values, as each recurrence requires; the community discards those that do not meet the criteria for ownership.

Verbal and Non-Verbal Dynamics of Oral Communication

Genre theory research may often focus specifically on written documentation but rhetorical forms are more than simply written forms. There are genres of movies, music, speeches, video games, etc. Consequently, the rhetorical form is a useful term that encapsulates all kinds of genres. At the same time, the rhetorical form is so often a reference to written communication. It should not be surprising that researchers who study writing would observe the writing, apply a robust theory to writing, and interpret their research results with the language of writing theory. However, a meta-language about documentation activities needs to escape the boundaries of written communication.

A meta-language could even benefit researchers. Words like Verbal and Non-verbal, oddly enough, do not refer to writing. A journal search supports my claim that "verbal" refers to written communication that would otherwise require words; "nonverbal" refers to communication that does not require words (i.e. charts, graphics, images, etc). Therefore, the term "written" refers to the product of pens, pencils, and paper. Simply put, once graphics became a reliable and accessible form of communication, researchers distinguished between verbal and nonverbal communication—particularly, scholars of visual rhetoric.

Verbal and non-verbal journal search. In the parlance of professional communication research, verbal and non-verbal communication mean something different. After a journal search of *Technical Communication* articles written between 1995 and 2010, I discovered verbal communication does not always refer to the spoken word. When *Technical Communication* researchers refer to verbal communication, they refer to written documents, as opposed to graphics or art. The following two examples are strong cases suggesting the way that genre theory trends towards written communication. In cases where I use the words "verbal" and "non-verbal," the authors in fact use verbal and non-verbal themselves.

- Dragga and Voss (2003). Hiding Humanity: Verbal and Visual Ethics in Accident Reports

  The article includes accident reports that feature both narrative text and photos. The author seeks to show that the text is articulated in a way that strips the humanity from traumatic accident reports. Interestingly, Dragga and Voss refer to the accident pictures as the non-verbal and suggest the narrative text of the report is the verbal.

- Doumont (2002). Verbal Versus Visual: A Word Is Worth a Thousand Pictures, Too

  Doumont details the difference between verbal and non-verbal. While the author defines verbal in a way that can include oral communication, the article is about the written communication that accompanies non-verbal communication. For instance, verbal displays text. Ultimately, the author recommends combining verbal and nonverbal communication for optimal effect in documents.

Other verbal and non-verbal examples. In addition to the articles by Dragga and Voss (2003) and Doumont (2002), there are other examples. These remaining examples date back to 1998. All told, the examples showcase how written communication and verbal communication are both something to read on paper.

- Brumberger (2004). The Rhetoric of Typography: Effects on Reading Time, Reading Comprehension, and Perceptions of Ethos

  Typography is one of the visual rhetorics of written communication—font, type size, etc. In particular, Brumberger separates non-verbal typography from verbal documents.

- Brumberger (2003). The Rhetoric of Typography: The Awareness and Impact of Typeface Appropriateness

Brumberger investigates whether clashes in typeface and text persona affect readers' perceptions of the text. The author defines verbal communication as the style and content of the text itself, rather than the visual (or the persona established by typeface).

- Johnson-Sheehan and Baehr (2001). Visual-spatial Thinking in Hypertexts

  The article distinguishes between hypertext "visual-spatial texts" and paper-based texts "verbal-linear."

- Qiuye (2000). A Cross-cultural Comparison of the Use of Graphics in Scientific and Technical Communication

  This article seeks to study the visuals in Chinese science magazines and instructional manuals, as opposed to American samples that put emphasis on the corresponding verbal explanations.

- Markel (1998). Testing Visual-based Modules for Teaching Writing

  This article posits that instructional materials that incorporate basic principles of visual design are more effective than those that are primarily verbal.

A meta-language about the rhetorical forms and rhetorical actions of genre theory should allow for more than what is written on paper, whether labeled verbal or non-verbal.

Reviewing Convergence of Meta-Language

Software documentation is traditional written communication. However, traditional theories of written communication do not seem to describe contemporary software documentation. However, not all "verbal" documentation is written anymore. I use the principles of genre theory (dynamism, situatedness, form and content, duality of structure, and community ownership) to highlight the rhetorical form that drives contemporary software development because limited vocabulary of written, verbal, and non-verbal communication cannot possibly describe a rhetorical form that is not in either the design documentation or evidenced in development decision-making activities. In other words, I cannot conceive verbal documentation without a new vocabulary.

There is more at stake than whether the meta-language can describe software documentation as verbal or non-verbal. The meta-language also describes how software developer communities interact with their rhetorical forms. Yates and Orlikowski (2002) describe the interaction of rhetorical forms and community: “A genre established within a particular community serves as an institutionalized template for social interaction—an organizing structure—that influences the ongoing communicative action of members through their use of it” (p. 15). Therefore, the rhetorical forms on which software developers rely are core decision-making assets. Yet, developers either have poor documentation practices or no documentation practices. Consequently, there must be another rhetorical form the community members use for ongoing communicative action.

Without an adequate meta-language to describe the invisible rhetorical form, professional developers are left with their own understanding and expectations, within the constraints of written documentation. Consequently, professional developers continue to kick at traditional documentation from their contemporary development workplaces. Yates and Orlikowski (2002) identify why developers cannot shake their expectations and their documentation: “the genre system…provides expectations about its socially recognized purpose” (p. 16). Therefore, software developers can expect their software documentation to serve their development projects in the prescribed way. As a community, they struggle to fit traditional documentation into contemporary practice.

TRADITIONAL SOFTWARE DEVELOPMENT AND DOCUMENTATION

With a meta-language of genre to describe new rhetorical forms in software development, the next step is to actually define what I specifically mean by “software documentation,” describe the purposes of software documentation and distinguish traditional development methods from contemporary development methods. Worth noting is the following details about documentation relate to traditional documents alone, as opposed to any contemporary rhetorical forms; in this way I can be clear about the standards of traditional documentation and how they connect so closely to traditional development practices.

External vs. Internal Documentation

There are two kinds of software documentation—external and internal. The difference between external documents and internal documents lies in the target reader. In the recommended practice of their own industry book about external software documentation, Denton and Kelly (Denton & Kelly, 1993) write: “This book is designed to help writers in the computer industry make their product documentation more useful, attractive, and accessible to their paying customers by building in quality from the beginning" (p. xi). In the case of the external documents Denton and Kelly describe, like software user documents or specialist manuals, the target reader is a customer and not on the development team (Barker, 1998). For instance, a grandmother editing photos and a studio animator making movies are both users who have different purposes for using the same software; however, the grandmother and the animator require different external documents to meet the complexity of their purposes. The varieties of external documentation serve consumers of software products.

On the other hand, internal documents are written for the development team; internal documents are design articulations that developers write to each other. Internal documents may include specification requirements, general design documents, feature elaborations, the art bible, specification documents, general design document, progress reports, etc. (Adams & Rollings, 2007; Rüping, 2003; Schultz, Bryant, & Langdell, 2005). The most useful documentation for developers is nested in the code itself; code comments are another form of software internal documentation (Rüping, 2003) because developers record the logic that justifies how and why they wrote specific code sequences. Other developers can pick up those sequences of code and contribute to the design without hours of puzzling over the original developer’s inscrutable logic.

In industry guidebooks that seek to identify how to write software documentation, the writers tend to focus on external users. Consequently, guides about external software documentation point to user manuals, promotional materials, and troubleshooting documents. The industry guides identify other user audiences as well: management, investors, and other developers in the company that require specialty instructions (Barker, 2003; Robinson & Etter, 2000). Developers who read these industry guides seek document design tips so that their user documentation can be designed in a way that both reduces unnecessary complexity and increases readability. The industry guides do not typically cover internal software documentation.

On the other hand, internal software design documentation references the supplementary design explanations teams of developers write to each other. In other words, those documented explanations are helpful when a dozen different teams are assigned to respective software features within the larger software architecture. Consequently, the design team, the software engine team and the GUI interface team can all access current documentation that identifies the documented details written by the respective teams, as well as individuals on teams.

When I refer to documentation activities and the written documents those activities produce, I refer to traditional documentation standards and practices. Consequently, the design documentation identified so frequently in this dissertation is internal, traditional documentation.

The Advantages of Documentation

Design documents have a specific purpose, follow conventional rules, and are written to fill particular needs. Whether developers work on software design or engineering, the development world loosely refers to both documentation activities and any written material a team generates to support their development work. Scott Berkun (2005) was a project manager at Microsoft for nine years and identifies the utility of what he calls vision documents; "a vision document is where all of the perspectives, research, and strategy are synthesized together. . .often contain requirements. . .anything the team (and client) agrees will be satisfied when the project is completed" (p. 59). The documentation consolidates the team's ideas, design choices, development plans, and consensus into one united vision. The more developers and designers there are the more a documented, united vision becomes critical. Among the many reasons to maintain documentation, there are three key advantages: a record of design decisions, a transmission of design decisions, and a reliable design resource.

Record of design decisions. A group of individuals cannot reliably remember the details of a detailed decision. The documentation preserves the details of those decisions; Berkun (2005) underscores the necessity for documenting decisions: “the reasons people had for listening to [decisions] today will be forgotten or ignored tomorrow” (p. 66). In fact, Berkun’s textbook also suggests that managers might forget the decisions and directions (p. 66) themselves. If only as a record of a team’s agreement the design document is essential. In addition, before a team gets mired down in the lines of code for a feature, they need to retain a perspective on the project’s vision and direction.

Transmission of design decisions. While Berkun (2005) mentions design documentation as a medium for communication “across a large organization” (p. 67), Ernest Adams and Andrew Rollings (2007), make communication a main feature of how they describe documentation. The authors claim, “the key part of [software] design is transmitting the design to other members of the team” (p. 62). The transmission of design decisions has multiple benefits for software development. The document is a record of oral agreements, turns vague ideas into explicit plans, and once written down is a “decision made, a conclusion reached” (p. 62). In addition, documentation ensures that teams do not overlook features; due to the strict specificity of code languages, the omission of a single feature necessitates rewriting thousands of lines of code to correct the omission. The authors point out that design errors are time consuming and costly; therefore, it is cheaper to thoroughly document the design before any code is written or artwork is created (p. 62). Design documentation's central objective is to broadcast a game's design to the development team.

Reliable resource of design decisions. The document Berkun (2005), Adams and Rollings (2007) describe is a source of information and the consolidation of a design plan. All the preplanning and consolidation makes documentation a fixed source of meaning on which teams can rely; Berkun writes: "When the vision is completed, the planning phase is over. The team should have the information needed to do good design work that satisfies the goals" (p. 77). No matter how displaced developers may be or how many development teams are on the same project, that one single resource keeps the entire project united. Yet, Berkun also advises: "Don't make the mistake of thinking that planning documents are fixed, rigid things: they're just documents" (p. 67). He indicates there is some kind of space between generation and completion where teams can situate their design document. While Berkun does not advocate endless revision, a space between generation and completion sustains the indispensible relevance of documentation to a team. That space is a prime example of the Duality of Structure; the document will always have a space in the community because the community is always sustaining and reproducing the document, at the same time.

Description of Governing Documentation

Adams and Rollings (2007) identify two important advantages to the maintenance of quality documentation: "[they play a part in] transmitting the design to other members of the team . . . [and] the processes of writing a document turns a vague idea into an explicit plan" (p. 62). Consequently, rather than a bunch of developers with vague, isolated ideas, the documentation ensures that everyone can understand the details of the team’s plan. At the same time, Adams and Rollings imply another purpose for development documents; they are guiding documents. Apparently, the most vital reason for documentation is to transmit concrete design plans to a distributed team and keep everyone on track. The documents are meant to rein in work done “on the fly” (p. 62), as well as restrain the development of great ideas that can delay the completion of development.

Adams and Rollings (2007) suggest that the documentation can act as a “paper trail” (p. 62) that can help teams resolve confusion about past agreements. While Adams and Rollings classify documentation as a predevelopment activity, they suggest that the documentation transmits those predevelopment decisions. However, that record of transmission no longer acts as a commanding resource because developers often rewrite the predevelopment documentation. In response, Alan Cooper (1999) suggests that documentation should act as strict blueprints (p. 237) that literally manage a project from the beginning; Cooper calls for developers to adhere to the predevelopment transmission.

Yet in opposition to Cooper (1999), the contemporary software developers to whom I have spoken were guided by their own decisions, rather than by what was written in a document. Contemporary developers were still generating the internal documents, even while retaining the purpose of internal documents; in other words, the importance of documentation remained, even as the importance of planning diminished. In fact, published industry guides presently continue to outline a planning methodology for the contemporary developers—the predevelopment transmissions Cooper advocates. Even while the discourse community and the situation of developers abandoned the preplanning of traditional methodologies; the purpose, form, and content of internal documentation did not.

Example: Edgetech's Governing Documentation

Edgetech is my name for an Orem, Utah computer game company. Edgetech is a classic example of a development studio that seeks to use a large, preplanned, governing document to manage contemporary practices. The purpose of this example is to show a traditional governing document that fails to adapt to the specific project and does not serve the contemporary developers anyway.

Edgetech has a single room in which the handful of developers can program and collaborate together in contemporary development fashion, even while the CEO writes the governing documents down the hall. In fact, the CEO uses legendary game designer Chris Taylor's generic game design document template—http://runawaystudios.com/articles/ctaylordesigntemplate.doc.

The template is an outline of feature and design sections that prompt game developers to thoroughly articulate their game design. For instance, one section relevant to the Edgetech might be a section about physical collectible cards interfacing with the game. The template is available for public use; Taylor gives the following permission: "for all of you who have ever wondered what they look like or need one for your own personal game project." Taylor's template is a blueprint with all the necessary section headings and sub headings; developers like Edgetech simply plug information into the appropriate fields. Table 2 lists Taylor's recommended sections:

Table 2

The 14 Recommended Sections of a Design Document—abbreviated presentation of the table of contents highlights the Design Document's topic areas.

1. Game Overview	8. Musical Score
2. Feature Set	9. Single Player Game
3. The Game World	10. Multi-Player Game
4. The World Layout	11. Character Rendering
5. Game Characters	12. World Editing
6. User Interface	13. Extra Miscellaneous Stuff
7. Weapons	14. Six Additional Appendices

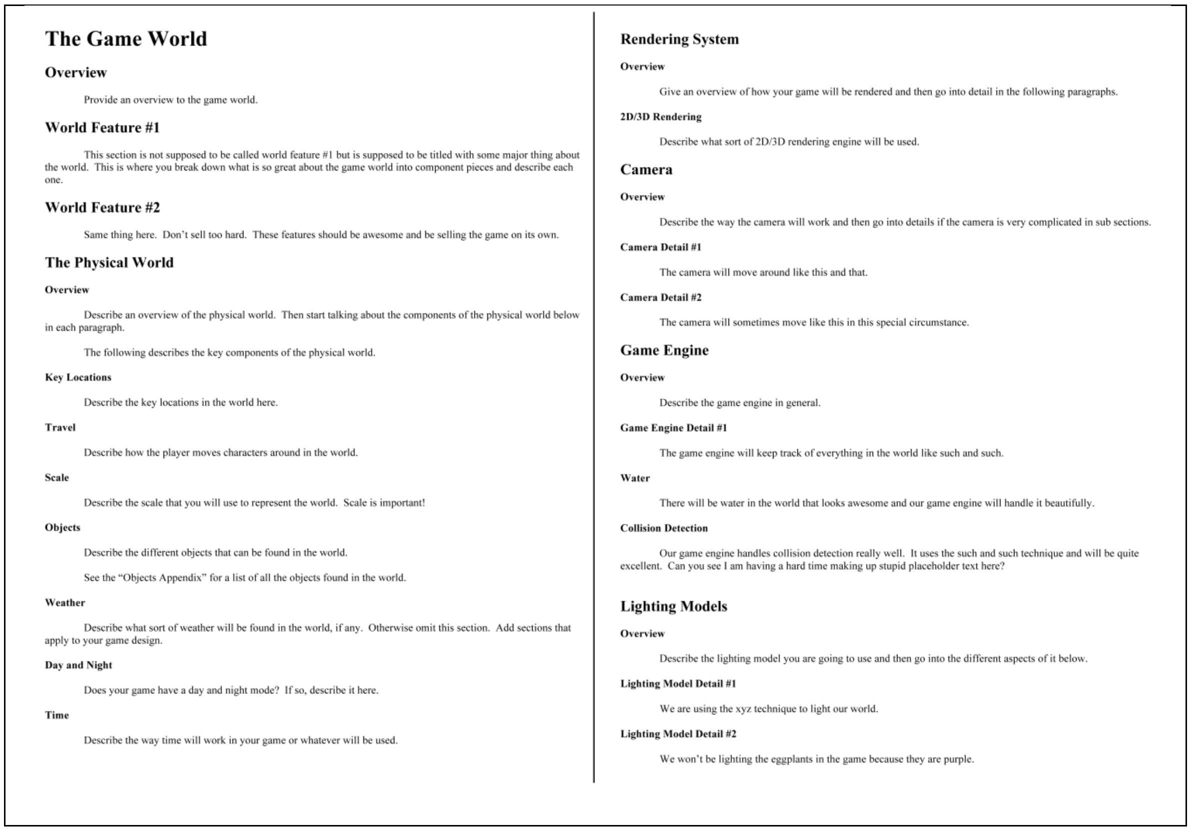
The governing document is Chris Taylor's template that includes a category for documenting the passage of day and night, among other things. The CEO complained to me about having to document day and night when that is not a feature in his company's games; therefore Edgetech used a static, templated document that recommended specific form and content. The purpose of Edgetech's documentation is preset for them so that they can have everything they need, as well as everything they do not need, thoroughly documented. While there is nothing inherently wrong about static templates, there is something wrong about canned preplanning that is completely disconnected from the contemporary methodologies of developers working down the hall.

Edgetech downloaded the general design document template from the website of Chris Taylor and was quite diligent with elaborating each of the recommended sections. Based upon my worksite visit, I know that Edgetech uses all the generic sections to detail their game software. Once their general design document is populated with information that details the design of each section, there is enough design information to meet the needs of investors and developers.

Within each section of the template, Edgetech finds specific recommendations for the form of the section, as well as content prompts. For instance, image 1 illustrates how the "Game World" section alone recommends the organization of form and content. The intention of image 1 is not to highlight the details of the template; rather, image 1 is meant to illustrate the clear organization and the quantity of prompts available to the prosumers of the template.

Image 1

Organization of the Game World Section of Chris Taylor's Design Document Template



The Game World

Overview

Provide an overview to the game world.

World Feature #1

This section is not supposed to be called world feature #1 but is supposed to be titled with some major thing about the world. This is where you break down what is so great about the game world into component pieces and describe each one.

World Feature #2

Same thing here. Don't sell too hard. These features should be awesome and be selling the game on its own.

The Physical World

Overview

Describe an overview of the physical world. Then start talking about the components of the physical world below in each paragraph.

The following describes the key components of the physical world.

Key Locations

Describe the key locations in the world here.

Travel

Describe how the player moves characters around in the world.

Scale

Describe the scale that you will use to represent the world. Scale is important!

Objects

Describe the different objects that can be found in the world.

See the "Objects Appendix" for a list of all the objects found in the world.

Weather

Describe what sort of weather will be found in the world, if any. Otherwise omit this section. Add sections that apply to your game design.

Day and Night

Does your game have a day and night mode? If so, describe it here.

Time

Describe the way time will work in your game or whatever will be used.

Rendering System

Overview

Give an overview of how your game will be rendered and then go into detail in the following paragraphs.

2D/3D Rendering

Describe what sort of 2D/3D rendering engine will be used.

Camera

Overview

Describe the way the camera will work and then go into details if the camera is very complicated in sub sections.

Camera Detail #1

The camera will move around like this and that.

Camera Detail #2

The camera will sometimes move like this in this special circumstance.

Game Engine

Overview

Describe the game engine in general.

Game Engine Detail #1

The game engine will keep track of everything in the world like such and such.

Water

There will be water in the world that looks awesome and our game engine will handle it beautifully.

Collision Detection

Our game engine handles collision detection really well. It uses the such and such technique and will be quite excellent. Can you see I am having a hard time making up stupid placeholder text here?

Lighting Models

Overview

Describe the lighting model you are going to use and then go into the different aspects of it below.

Lighting Model Detail #1

We are using the xyz technique to light our world.

Lighting Model Detail #2

We won't be lighting the eggplants in the game because they are purple.

In particular, the CEO of Edgetech confessed a frustration with the Game World section of the template. He mentioned his frustration with the Day and Night content recommendation: "Does your game have a day and night mode? If so, describe it here." Day and Night is under the Physical World subheading. Apparently, the CEO was tired of explaining, every time he filled in the template, that his game has no cycle of day and night. Rather than omit that part of the Physical World documentation, he was faithful to the prescribed form and content.

Two sentences about the absence of day and night does not seem like a big hang up for software development. However, the story does demonstrate that contemporary developers work really hard to comply with traditional documentation standards. The CEO's commitment to preplanning is completely disconnected from the contemporary methods of the developers who simply cannot wait around for the documentation of day and night cycles. The purpose of the documentation was the same, even if the community and situation were completely different.

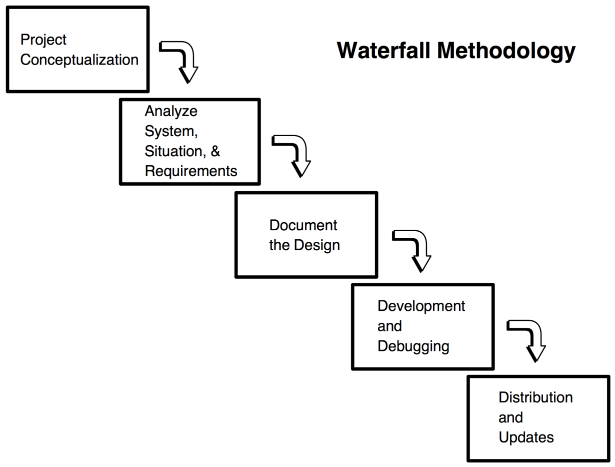
Reviewing Two Development and Documentation Methods

I often refer to traditional and contemporary software development methodologies. Contemporary developers revolutionized traditional methodologies so that Community Ownership—mindsets, philosophies, priorities, deadline management, and client relations—was radically different. Insofar as the community and situation are different, the purpose of documentation should be significantly different too. However, the purpose of documentation is unchanged.

Traditional development and documentation. Software developers who use the traditional method of software development are familiar with the purpose, discourse community, and situation of internal documentation. In fact, they believe thorough internal documentation is necessary for the team to have a firm grasp on purpose, discourse community, and situation. In other words, traditional developers seek to plan, design, and document everything they can before they begin development (Barker, 1998). The traditional method is pictured in image 2. Not only does traditional methodology suggest an ordered sequence of steps, the method also suggests a flow of progress that can be credited to preplanning and documentation.

Image 2

Model of Traditional Methodology


Waterfall Methodology
Project Conceptualization
Analyze System, Situation, & Requirements
Document the Design
Development and Debugging
Distribution and Updates

The traditional method was borrowed from engineering methods. After all, if engineers can succeed with only one shot at making a bridge function safely, then software developers can also rely on the same principles for similar success. Subsequently, the significance of internal documentation in software development required qualified writers. In fact, a 1985 *Technical Communication* article (Antoine, 1985) identified the need for technical writers in software development. That article also outlined how technical writers could serve the computer industry by clearly outlining both the kinds of documents and kinds of literacies the software industry needed. The article demonstrates how perfectly technical writers fit into the purpose, community, and situation. However, contemporary developers utilize more rhetorical forms than simply written documentation so that both the community and situation are different than Antoine described in 1985.

Writers like Kent Beck (2000) look down on oral communication as the absence of documentation and the rejection of communication. This hard position is because industry experts see documentation as a concrete artifact of communication to which developers can look for what might be a difficult design.

The concrete artifact is what makes documentation so vital in software development. Scott Berkun (2005) illustrates this vitality with the drastic consequences of an engineer forgetting the design of a nuclear reactor (pp. 66-67). Engineers need complex, concrete instructions to build a nuclear reactor. A software program can be as complex as a nuclear reactor, even if there are not the risks. Without a concrete design, a team of software developers will struggle to redesign the software.

At the same time, the genre rules for a traditional document's form and content are very different from a contemporary rhetorical form's structure and content. Specifically, the form of contemporary rhetorical forms is not concrete and includes oral communication. Yet, contemporary developers and software industry guides all expect contemporary developers to write traditional documents. Adams and Rollings (2003, 2007), Berkun (2005), Cooper (1999), and Brown (2007) all suggest traditional documents are the model by which contemporary developers should write documentation. In addition, Genre Theorists like Spinuzzi (2002, 2003), Williams (2003a, 2003b), and Winsor (1990) trace the roles of Traditional documents in decision making.

Reviewing contemporary development and documentation. In the mid-90s, developers began to tire of the battleship methodologies they used—they needed more flexible ways to turn the boat around. As programs became smaller and release deadlines became shorter, developers realized the methodological hulks would no longer serve. They changed their value systems and set new gauges for progress. They streamlined a cumbersome process into tight cycles of development. They formed a whole new philosophy of nimble, user-driven development practices.

Developers abandoned traditional methods—Agile, Object-oriented, Use-case, rapid-prototyping, extreme-programming, etc —and they formulated various contemporary methodologies. Many of those methods are classed under the category of what I term Contemporary development. The commonality between all these methods is the quick, responsive development practice. In fact, contemporary developers hold to the Agile Manifesto, which outlines three distinct values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Most notable is when there is a choice between completing a prototype before the deadline and documenting design decisions; contemporary developers abandon the documentation in favor of completing development cycles on time. Consequently, developers cannot comply with the documentation standards they seek to maintain. Most interesting, contemporary developers reject traditional development methodologies, even while they figure out how to perform traditional documentation activities.

Contemporary development described by developers. A contemporary developer I know in Newbury Port, Massachusetts once told me: “You want to know how I document? I don’t” (Professional Conversation). In fact, I have met many contemporary developers. I have performed on-site visits. Where I had wanted to see genre theory in action, see samples of contemporary documentation and have conversations about successful documentation practice, I instead heard frustration. Much of that interaction was unrecorded and all of it was bound by nondisclosure contracts I signed. However, I did take notes during a series of casual conversations with two developers in Logan, Utah on March 14, 2008. We had a very conceptual conversation about programming methods and they helped me understand the constellation of methodologies. Essentially, they told me there was an entire spectrum of methodologies distinguished by the amount of preplanning required. One of the two developers drew a diagram on a napkin; I have recreated the diagram for Image 3.

Image 3

The Spectrum of Programming Methodologies Identified by Logan, Utah Developers

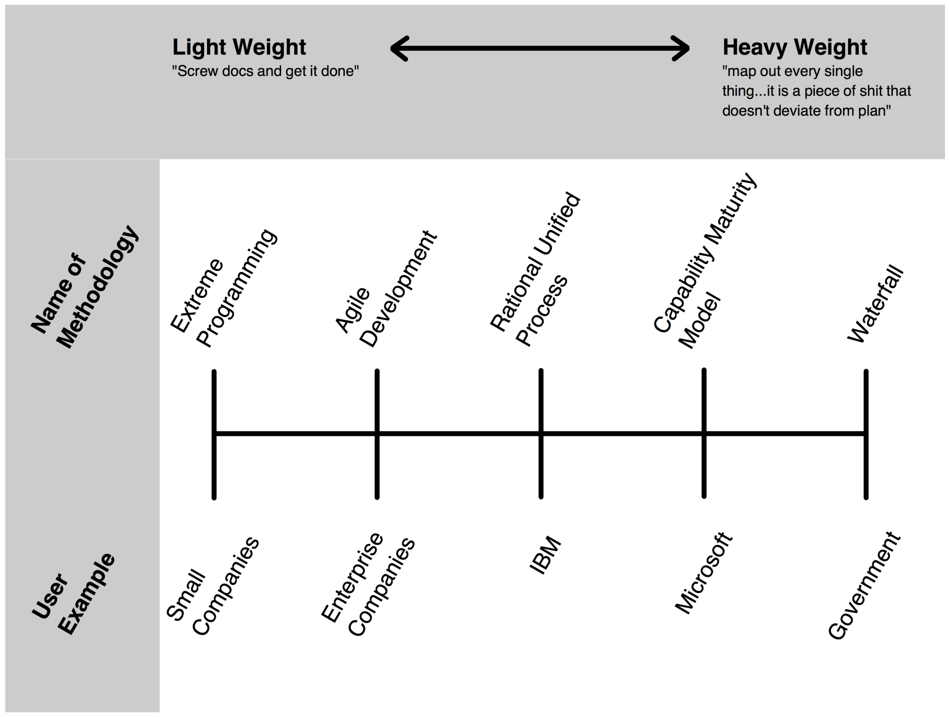


Image 3 represents the spectrum of software methodology titles on the top and examples of company-types that use them. The two developers had a lot to say about the spectrum but their most poignant thoughts identify what they think of heavy weight documentation: "map out every single thing…it is a piece of shit that doesn't deviate from the plan." While traditional methods still worked for extremely large software architectures, with a large number of program teams, there were other software projects that were not nearly so large and not nearly so slow to require change.

An easy contrast between traditional and contemporary. The two Logan developers organized the spectrum left to right—contemporary to traditional. Where the two Logan developers helped me understand contemporary development, I knew another programmer that helped me understand traditional development. In direct opposition to the far left of image 3 "Screw docs and get it done" is the traditional development.

The traditional programmer had previously worked for a government software contractor in St. Paul, Minnesota. The programmer joked that for every one line of code he had to write 20 pages of documentation. While this was an exaggeration, the life of a government software product, the scale of development, and the need for frequent audits made traditional development the best option. Contrary to traditional values, contemporary developers are smaller companies, with changing needs, and products with a short lifetime.

HIGHLIGHTING PURPOSE IN GENRE THEORY

I have presented a meta-language of genre theory and have outlined the chief purpose of traditional documentation. However, that “Purpose” does not match the needs of contemporary developers, even if they still try to meet that “Purpose” in their contemporary practice. In terms of my research, as soon as developers formed new values to support more contemporary methods of software development, developers changed their situation and their community. However, they did not change the purpose of documentation to meet the needs of their situation or community.

There are clear examples of how the industry has not converted the purpose of documentation from traditional to contemporary. One example is in the *Forward of Communicating Design: Developing Web Site Documentation for Design and Planning*. Daniel Brown identifies the target audience of his book and explains how to read his book; he also identifies his assumption that the situation does not impact the document's content: "Although this book tries to remain agnostic about process, it does make some assumptions about how you operate and what your working environment is like. You may find yourself in a different situation, but it should not affect the content of your documentation" (Brown, 2007). Brown suggests the traditional documentation is static and always serves the same purpose no matter the situation. In fact, how developers operate and how their environment works has everything to do with the purpose of a document. By extension, the document serves a different purpose for each situation, and community.

Genre theory indicates a reason why traditional documents fail in contemporary practice: genre theory is the combination of purpose, situation, and community. Situation and Community are key elements in genre theory. Consequently, the following sections elaborate the rhetorical situation and the meaning-making of communities. In order to showcase the centrality of purpose for genre theory, I review three different schools of genre to elaborate how purpose fits into each one.

Purpose in Rhetorical Situations

The situation might be an important element but the trick is that with an organic model like genre theory, the situation is necessarily different every time, even though a repeating situation seems to suggest comparable or similar material features. This is the particular conundrum Miller (1994) tackles in her own research:

> What is particularly important about rhetorical situations for a theory of genres is that they recur…What recurs cannot be a material configuration of objects, events, and people, nor can it be a subjective configuration, a 'perception,' for these, too, are unique from moment to moment and person to person. Recurrence is an intersubjective phenomenon, a social occurrence, and cannot be understood on materialist terms. (p. 156)

Miller points to the persuasion in a community as an intersubjective phenomenon that morphs an otherwise immutable, repeating situation.

The rhetorical situation is a moment on which forces of persuasion have a great deal of influence. Keith Grant-Davie (1997) states the meta-definition for the rhetorical situation is: "a set of related factors whose interaction creates and controls a discourse"

(p. 265). The interaction of those related factors necessarilly involves rhetors—people who seek to persuade an audience. Whether intersubjective phenomenon or related factors, either one suggests rhetors can evaluate the results, or consequences, because a rhetorical situation is a historical event with causality (Grant-Davie, p. 264). A real-time, causal approach to rhetorical situations allows rhetors to analyze, "why decisions were made and why things turned out as they did. . .that some events might easily have turned out otherwise, while the outcomes of other events seem all but inevitable when seen in light of the situations in which they occurred" (Grant-Davie, p. 264). Rhetors can clearly delineate between cause and effect on historical events because the outcomes are indisputable, as when the American colonists won the Revolutionary War in 1783; the outcome was clear because the United States is no longer a British colony. Yet, rhetors implement symbolic strategies of persuasion to project what kind of situation generates a specific outcome. Historians use both evidence and rhetoric to interpret the cause of the Revolutionary War. The extrapolation of cause can then demonstrate how one persuasive strategy, or sequence of events, over another can produce different effects in the same rhetorical situation.

In terms of software documentation, the historic event is clear to everyone on the team. For instance, a team meets every Monday morning to showcase prototypes; that weekly meeting is a historic event. However, the outcome of that event might be new design directions and new prototype requirements; that outcome is the result of the intersubjective phenomenon Miller described and the symbolic strategies of persuasion Grant-Davie described.

Purpose and Community Meaning-Making

Like Carolyn Miller suggests, the intersubjective phenomenon are the features of a rhetorical situation, rather than materialistic features. That intersubjectivity is resolved by the rhetors involved in the rhetorical situation. The resolution those rhetors create is an act of meaning-making that defines the community. Consequently, there are many social theories that define “community” and that research can highlight the value of meaning-making communities in genre theory.

Social theory varies from genre theory insofar as community is the primary transmission of meaning-making, rather than written communication. While Nancy Roundy Blyler and Charlotte Thralls (1992) suggest that all social theories might agree on the “centrality of socially mediated meaning” (p. 125), they use four categories to trace differences in the key tenets—community, knowledge and consensus, discourse conventions, and collaboration. Rather than tackle the impossible task of reviewing all social theories, I plan to showcase community, knowledge and consensus, discourse conventions, and collaboration with only two social theories: social constructionism and paralogic hermeneutic theory. Of the two, I believe paralogic hermeneutic theory does the most to highlight how a community of contemporary developers can employ new rhetorical forms to communicate in development projects.

A brief summary of paralogic hermeneutic theory. Paralogic Hermeneutic theory focuses on the unlikely coherence of two different minds in the same unique moment of communication. Thomas Kent (1993) defines Paralogy in the following manner: “paralogy refers to the uncodifiable moves we make when we communicate with others, and ontologically, the term describes the unpredictable, elusive, and tenuous decisions or strategies we employ when we actually put language to use" (p. 3). The uncodifiable nature of communication is key to the theory because the community must negotiate unpredictable communication moves. Another key term is coherence because meaning-making happens when the rhetors in a paralogic situation align their understanding. Mark Zachry (2005) clearly identifies exactly why paralogy literally defies logic: "human communication is paralogical rather than logical because in any given example of communicative interaction, a superfluity of potential understandings is only temporarily and locally arrested when the participants involved in the communication 'come to terms' over meaning so that they can interact" (p. 179). The former social theories rely on a community that can arrest understanding and codify that knowledge in a meaning-making process. However, paralogic hermeneutic researchers insist that understanding is uncodifiable. Therefore, a document can be complete but the community is too uncodifiable to surrender meaning-making to that document.

Social theory tenet: community. For the social constructionist, *Community* is the source of knowledge. In fact, social constructionist researcher Kenneth Bruffee (1986) posits that knowledge is social in nature, rather than individual, internal, or mental (p. 775). Yet according to Blyler and Thralls (1992), ideologists argue that social constructionists like Bruffee "separated the social conditions of writing from questions of power and control" (p. 131); rather than a benign and apolitical process that presupposes the communal membership of everything, the ideologist sees "the authority structures that enable communities. . .to maintain and legitimize social orders and practices under the auspice of tradition and custom" (p. 132). Ideologic communities situate meaning-making in the power struggle.

Paralogic hermeneutic theory posits that *Community* is notably impermanent and insubstantial. Superfluous communities have no structure because such constructions or authoritative forms require stable, codifiable definitions. Instead, Blyler and Thralls (1992) argue that paralogic hermeneutic theory builds uncodifiable community on the temporary "rapport experienced by communicants as they interact" (p. 137).

Social theory tenet: knowledge and consensus. Socially justified beliefs are forms of *Knowledge and Consensus* that benefit from the agreement of an entire community; Blyler and Thralls (1992) detail how social construction goes on to suggest that community interaction enables consensus (pp. 128-9). However, Ideologic social theory believes knowledge and consensus is an instrument of power and exclusion (p. 133). Consequently, while social construction minimizes and silences discourses, the ideologic approach makes meaning as "some interests are suppressed while others dominate. Consensus is not so much an index of agreement as an exercise of power" (p. 133). In other words, the ideologic meaning is in the struggle for power.

Zachry (2005) states that paralogic hermeneutic theory focuses *Knowledge and Consensus* on interpretation and the "primacy of communicative interaction" (p. 137). Interpretative achievements are still uncodifiable and temporary because each additional interaction shifts understanding once again. Zachry states: "meaning is never predictably constrained because communicative interaction is always defined by ongoing interpretive acts" (p. 180). Therefore, knowledge and consensus is an ongoing interpretation, rather than static truth.

Social theory tenet: discourse conventions. The social construction approach to *Discourse Conventions* suggests that communities are constituted by language.

Consequently, language is a tool or an intermediary that communities use to understand and identify communal agreements and truth. On the other hand, ideologic researchers focus on the signs and symbols that mark ideologic knowledge and support the dominant consensus. Ideologic researchers therefore seek to examine those dominant discourse conventions in the hopes of breaking down their power.

Discourse Conventions and paralogic hermeneutic meaning-making are actually incompatible; discourse convention is the definition of codified, planned, and reliable communication, whereas Paralogic Hermaneutic theory is uncodifiable and unplanned. Zachry (2005) articulates the place of discourse convention in paralogy: "communication always occurs in the presence of an 'other' via communicative interaction. That is to say, meaning does not reside 'in' texts" (p. 180). Rather, the meaning is always in the interaction. Zachry goes on to suggest exactly why discourse conventions are inadequate as either signs, symbols, or markers: "any given rule, guideline, or strategy—regardless of its complexity—cannot offer a fail-proof way for moving ideas from one mind to another" (p. 180). The only remedy for communication is to simply create temporary communities within which communicants can parley for meaning.

Social theory tenet: collaboration. Blyler and Thralls (1992) indicate that social constructionists believe (p. 130) *Collaboration* is a participation in conversations with knowledgeable peers; the effect de-centers authority and acculturates communicants. On the other hand, the ideologic approach argues that the collaboration is a site of struggle, where power and acculturation is suspect (p. 135).

Paralogic hermeneutic is a social theory of purely *Collaboration*; consequently, Blyler and Thralls (1992) indicate the theory is collaborative by nature (p. 139). Zachry

(2005) outlines exactly what role tools and preparations play in paralogic collaboration: "words are introduced and consumed, meanings are assigned, new words are exchanged, meanings are remade to accommodate this exchange, and the process continues on as long as the communicators are engaged with each other. Communication, therefore, is never static" (p. 181). Collaboration means design documents are moments of exchange and constant negotiation.

The uncodifiable nature of paralogic documentation is worrisome to some developers who rely on what they see as the predictability of more traditional development methods. Traditional documentation simply utilizes discourse conventions negotiated by a community of meaning-makers. On the other hand, contemporary developers already practice an uncodifiable development practice. They embrace new rhetorical forms in their communication; contemporary developers bring meaning to the documentation, rather than look to the documentation for the discourse conventions.

Purpose and the Three Schools of Genre Theory

Not all genre theorists trace documents in the same way. In fact, there are three different schools of genre theory. There is classical genre theory, Sydney Australia school of genre theory and genre theory. The nature of the rhetorical forms is where the different schools vary—i.e. prescribed forms versus organic forms. These variations are important to specifically identify what I mean by genre theory.

Classical genre theory. Classical genre theory precedes both Sydney and North American genre theories. I term the theory "classical" to distinguish it from the newer theories. Classical genre theory is made up of static categories like romance novels, haiku poetry, and software industry guides. In the case of software development documentation, there are specification documents, the user manual, specialist instructions, and the art bible. The categories are constrained by clearly set rules. William Harmon and C. Hugh Holman (1996) state classical genres, like the User Acceptance Test Plan document, designate: “the types or categories into which literary works are grouped according to form, technique, or, sometimes, subject matter . . . [and] implies that there are groups of formal or technical characteristics among works of the same generic kind regardless of time or place of composition, author, or subject matter” (p. 231).

These categories have specific rules that make it easy to find them in different parts of a bookstore. In addition, the genre categories make it easy for people to talk about their favorite horror novel author with a common idea of which authors qualify for that kind of conversation. Consequently, the formal rules identify the form a genre should take and the content a genre should include. Of course, artists push the boundaries of genres but that does not mean that “pushing the envelope” entails transgressing the actual boundary. Instead, an artist seeks to play within the boundary in original ways that keeps the genre fresh and exciting.

Sydney australia genre theory. The Sydney School of genre theory acknowledges fairly inflexible standards and rules; however, the theory conceives of genre structures as always imbedded in political and economic situations (Freedman & Medway, 1994). In other words, the Sydney researchers are not so interested in what makes a good office memo; rather, the Sydney Australia genre researchers extrapolate the impact of an office memo in a political or economic milieu (Freedman & Medway, 1994). The rules are static so that something like a memo might have had a controversial impact and heighten office tensions. Another example might be a specialist instruction written for a cadre of multi-lingual and illiterate factory workers who must tangle with international issues that might conflict with company values.

North american school. Genre theory proposes that genre categories actually shift the temporary genre rules each time the community participates and repeats a situation. Rather than static, self-evident categories; genres are social, community-defined phenomenon. In other words, genres like software documentation have form and content set by organic social needs and unstable community standards. Genre theory highlights recursive interactions between people and the rhetorical forms they use to communicate.

Two key distinctions. There are two ways that the Sydney school and the North American school blur together. Whether an office memo in a political milieu or an office memo in a community of meaning-makers, there is still a document that imposes stability and meaning on a situation. In addition, social structures are key elements in both theories; a structured office memo genre has a meaning-making impact in both schools.

Situatedness in the Sydney school and Situatedness in the North American school are distinguishable because of the situated knowledge and enculturation specific to the North American school. In other words, genre theory posits a Nietzsche-like situatedness—in which knowledge and agents form a strict context of meaning-making. Therefore, the situation and the community organically form a specific epistemology.

Social structures are distinguishable because genre theory features a Duality of Structure—genres reproduce and sustain themselves at the same time. Duality of Structure makes rhetorical forms like an office memo both change to the needs of the community and stabilize meaning-making activities in the community. That means rhetorical forms are mutable structures in genre theory.

PROFESSIONALS AND THEIR DOCUMENTATION

I have previously suggested that genre equals purpose, community, and situation. Shifts in this triad substantiate changes to the documentation of the software industry. Consequently, a shift from traditional to contemporary was necessitated by the rapid rate of change any single software development project undergoes. The traditional method was meant for titanic-sized software implementations and was not agile enough to work for more enterprise software projects. The contemporary developers changed the situation by which they document and the community within which they document but they did not realize the purpose of their documents changed too. They no longer need comprehensive, forecasting documents that clearly elaborate features and govern the path of development. However, they still try to meet the governing purpose of their documents, even while they already employ newly purposed rhetorical forms in their new situation, and community.

Part of the reason for the oversight is the word "document" itself. The word naturally lends itself to written communication. In addition, the word is both a noun and a verb at the same time: I like to document my work; I like to work on my document. Even more complex is the label "Documentation" because it is a classified activity in development methodology. Without more vocabulary words, it is difficult to demonstrate how the purpose of documentation changed but developers did not stop documenting in the same way. In fact, I nearly did not see the oversight myself. The meta-language of genre theory reduces that difficulty.

Purpose as Central to Software Documentation

I made the same oversight while meeting with two Logan, Utah developers (personal communication, March 26, 2008). I was looking for samples of documentation that would illustrate genre theory in the software development industry. I had read the industry guides and had my head full of theories; I could not wait to see something like the practical application of "Duality of Structure."

The senior programmer walked me step-by-step through his contemporary development methods. He described a recursive, rapid development cycle of implementation, testing, client feedback, design, and documentation. Documentation was actually situated between every step in his development cycle as each cycle ended with documentation. In fact, the two developers' workflow included 13 different proprietary, internal documents and five distinct documentation stages.

With a process that was clearly marked by a thorough documentation procedure, I concluded that the two developers' documents were the milestones that marked progress on their development projects. However, the senior programmer did not agree with my conclusion about document milestones; after all, he did not think of his process as a documentation heavy, preplanning-oriented, traditionally modeled practice. In his words, traditional development is a "piece of shit" and his practice was not a piece of shit, even if he produced documentation.

Insight about purpose in software documentation. I still perpetuated the isolation of purpose from both meaning-making communities and rhetorical situations. In other words, even though his situation, and community was not traditional, I imposed traditional purpose on his documentation. I interpreted the presence of adequate documentation as a practice that met the standards of genre theory. Instead, I should have seen that two developers' documentation transmitted design but was not the dynamic rhetorical form in which they situated their knowledge, owned their conventions and both structured and reconstituted their community.

The two developers knew the documentation was not governing their workflow, even if they called them "governing documents." They understood something I could not see and that they did not have a meta-language to describe. They understood their discourse community, as well as their situation. While their documents served traditional purposes, the actual role they played in the discourse community was less important. Where I concluded that documents were the milestones in the workflow, the developers saw only a bunch of required documents. They knew there was something else—another rhetorical form—that they used to structure and reconstitute their practice.

There is always another traditional document that records legacy decision making; however, there is no documentation for the decision work that happened between cubicles or in the hall near the drinking fountain. In the case of the two Logan, Utah developers, there was no documentation when they surreptitiously collaborated to resolve design problems. They talked and then they implemented; when the step in their workflow was done they would document what they ended up with but did not document the meaning-making process, rather, they conversed about meaning-making. If there was a milestone marking each stage of their development cycle, the developers frequently described their ongoing contact with their users.

The Edgetech Narrative—in the Meta-Language of Genre Theory

With the intent to come full circle back to Edgetech, I want to refer again to that developer's narrative—this time with the meta-language. There are two ways to interpret what I know of Edgetech's documentation practices. In the first case, there is what genre theory predicts. In the second case, there is what I observe when I can work directly with developers. The two means of interpretation yield very different results.

Narrating genre predictions. Genre theory predicts that the community recycles through written documents and that each cycle creates the form and content of the documentation—one rhetorical situation at a time. Communities act on documents based upon needs. Those needs determine the form and content of the document. In the case of Edgetech and Chris Taylor's generic design document template, the community simply omits the "Day and Night" entry. Consequently, the form and content are very liquid conventions. Genre theory also predicts that the community responds to what is already written; consequently, the writing still guides the community. For instance, the omission of "Day and Night" might prompt a design discussion about its absence from the documentation if the team becomes aware of a need for Day and Night transitions in the game.

Therefore, I would expect to see Edgetech's CEO writing and using documentation among the developers and graphic artists—among those who make the design decisions. I would expect to see the general design document organized according to the needs of the community. In addition, I would expect the developers and graphic artists to draw on the documentation as they develop their game; after all, genre theory suggests that documents have authority when the community makes decisions. Finally, I would expect the developers and graphic designers to play key roles in the authorship of the community's written communication. However, none of this is what I found at Edgetech's studio.

Narrating what was not predicted. Even if the Edgetech developers and artists do not use the CEOs general design document, genre theory still predicts that the discourse community generates communication. That communication does not have to be something long and detailed like a general design document. Particularly, many software developers write much smaller documents, instead of the general design document. Most importantly, they write what matters when it matters in the way that it matters.

Genre theory does not predict Edgetech's useless, template-like general design document. The theory does not predict the company's developers and artists cut off from the production of a guiding document. Neither does genre theory predict that decisions made by the community might never be documented; Edgetech developers clustered around the computer monitor might implement a decision without writing a single thing. Genre theory does not predict a successful discourse community if the community does not generate written communication; however, there was a new rhetorical form that stabilized the meaning-making community, even while the CEO plugged information into the cookie cutter template.

ORAL COMMUNICATION AS A RHETORICAL FORM

I have frequently referred to rhetorical forms. While an ecosystem of rhetorical forms is key to stabilize a development project, I want to highlight oral communication as a rhetorical form. I explain what I mean by oral communication, apply oral communication to documentation failure and plug oral communication into genre theory as a rhetorical form.

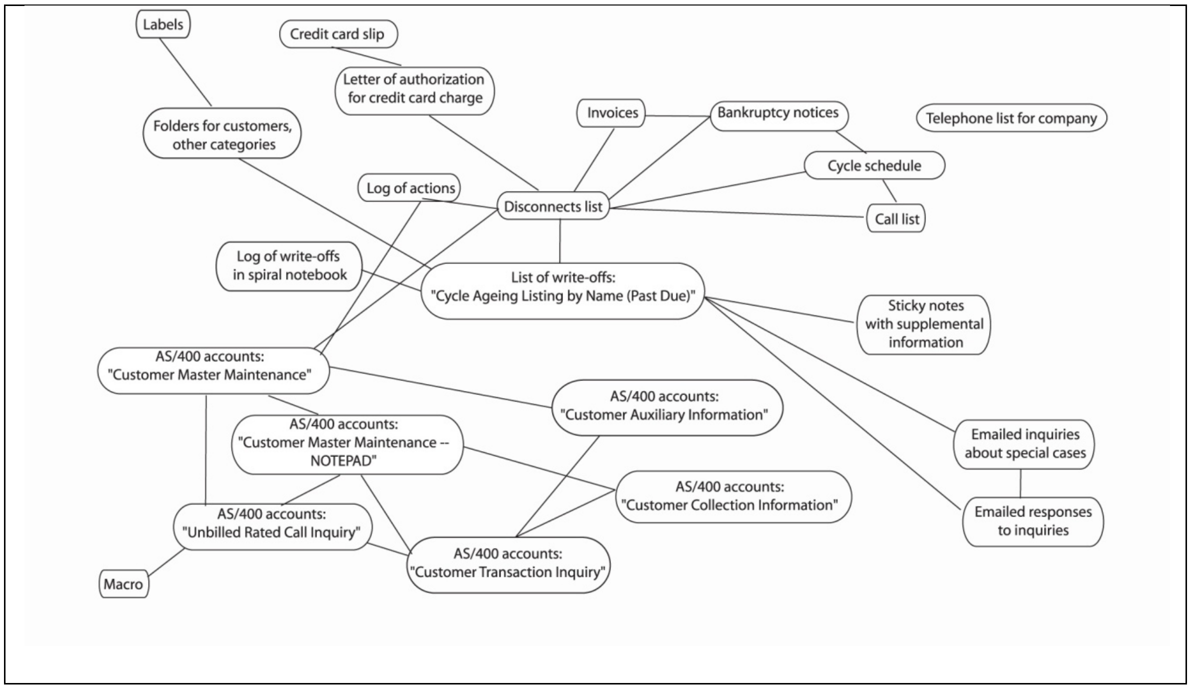
Utterances as Oral Communication

While the utterance is a core unit in genre theory (Berkenkotter & Huckin, 1995; Kent, 1993; Spinuzzi, 2003) and while researchers do identify a speech genre, it is actually a very novel observation that developers "speak to each other." In fact, M. M. Bakhtin (2007) gives as examples of the speech genre: "novels, dramas, all kinds of scientific research, major genres of commentary, and . . . organized cultural communication (primarily written)" (p. 62). Bakhtin specifically identifies that speech genres are primarily written. Actual, literal speech in genre theory is novel because so often speech signifies meeting minutes, rather than the meeting.

Genre ecologies, without utterances. Clay Spinuzzi is a genre theorist who researches the web of internal and external documentation used at workplace sites. His research highlights the concept of genre ecologies (Spinuzzi, 2002, 2003). In other words, document genres are not isolated pieces of writing; rather, document genres are parts of a constellation of co-constituted (Spinuzzi, 2003) genres—a genre ecology. Spinuzzi maps out what ecologies look like in the various communities he researches; the maps detail the connections between documents and highlights the clusters of activity at work sites. Image 4 is one such map taken from *Modeling Genre Ecologies* (Spinuzzi, 2002).

Image 4

Spinuzzi's Ecology Map from Modeling Genre Ecologies.

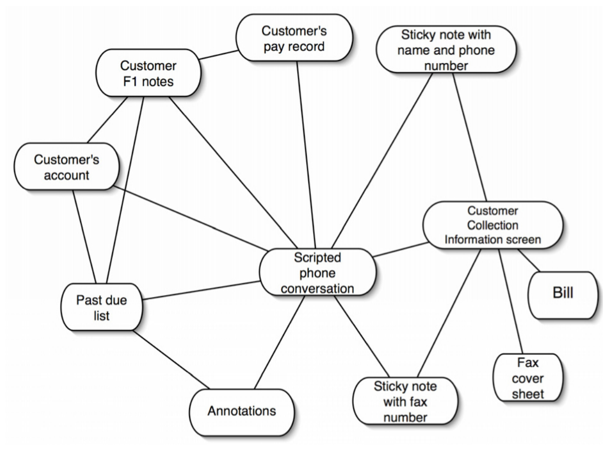

Labels
Credit card slip
Letter of authorization for credit card charge
Invoices
Bankruptcy notices
Telephone list for company
Folders for customers, other categories
Cycle schedule
Log of actions
Disconnects list
Call list
Log of write-offs in spiral notebook
List of write-offs: "Cycle Ageing Listing by Name (Past Due)"
Sticky notes with supplemental information
AS/400 accounts: "Customer Master Maintenance"
AS/400 accounts: "Customer Auxiliary Information"
AS/400 accounts: "Customer Master Maintenance -- NOTEPAD"
Emailed inquiries about special cases
AS/400 accounts: "Customer Collection Information"
AS/400 accounts: "Unbilled Rated Call Inquiry"
Emailed responses to inquiries
AS/400 accounts: "Customer Transaction Inquiry"
Macro

The map details the genre ecology (Spinuzzi, 2002) used by credit and collections workers at a "medium-sized regional telecommunications company" (p. 200). The map outlines the co-constitution of genres like bankruptcy notices, labels, log of actions, sticky notes, and email. However, the map does not include workers talking to each other. Spinuzzi interviewed and observed workers at the telecommunications company; he was aware that the workers spoke to one another and regretted how he simply could not record spontaneous speech acts; however, the genre ecology omits speech. The missing bubbles from this ecology map are chitchat at the drinking fountain, the soda machine, and any other location where oral communication impacts the project.

This is not to say that Spinuzzi does not acknowledge speech acts in other ecologies. For instance, Image 5 showcases how oral communication is a central component of the ecology. The ecology showcases the work of a single telecommunications professional named Ralph (Spinuzzi, Hart-Davidson, & Zachry, 2006). Ralph's ecology of genres all connect to his phone conversations.

Image 5:

Spinuzzi's Ecology Map from Chains and Ecologies.


Customer's pay record
Sticky note with name and phone number
Customer F1 notes
Customer's account
Customer Collection Information screen
Scripted phone conversation
Bill
Past due list
Fax cover sheet
Sticky note with fax number
Annotations

Telecommunication companies sustain meaning-making with or without oral communication but a software development company cannot omit speech without neglecting to account for how the community makes meaning-making decisions.

Documentation goals and documentation failures. Many developers are like the Orem developer; they produce traditional documents that do not really serve their needs, and some developers do not document at all; yet, they are still communicating. In both cases, developers employ oral strategies as their principle rhetorical form.

Unfortunately, both professional communicators and the authors of industry guides agree that an absence of documentation is poor practice—whether or not there are new rhetorical forms of communication. In fact, even the father of Extreme Programming—Kent Beck—does not recommend the absence of documentation: "saying 'we don't have to write documentation because we're extreme,' shows contempt for communication, not an embracing of communication as a value" (Beck, 2000). Consequently, Beck and many developers in the field declare a failure to maintain traditional documents is a failure of communication. The Orem developer is not exempt because poor traditional documentation is also a failure. However, all these judgments of failure rely on the traditional purpose of traditional documentation practices, whereas the discourse community and situation changed to contemporary development.

Oral Communication in Genre Theory

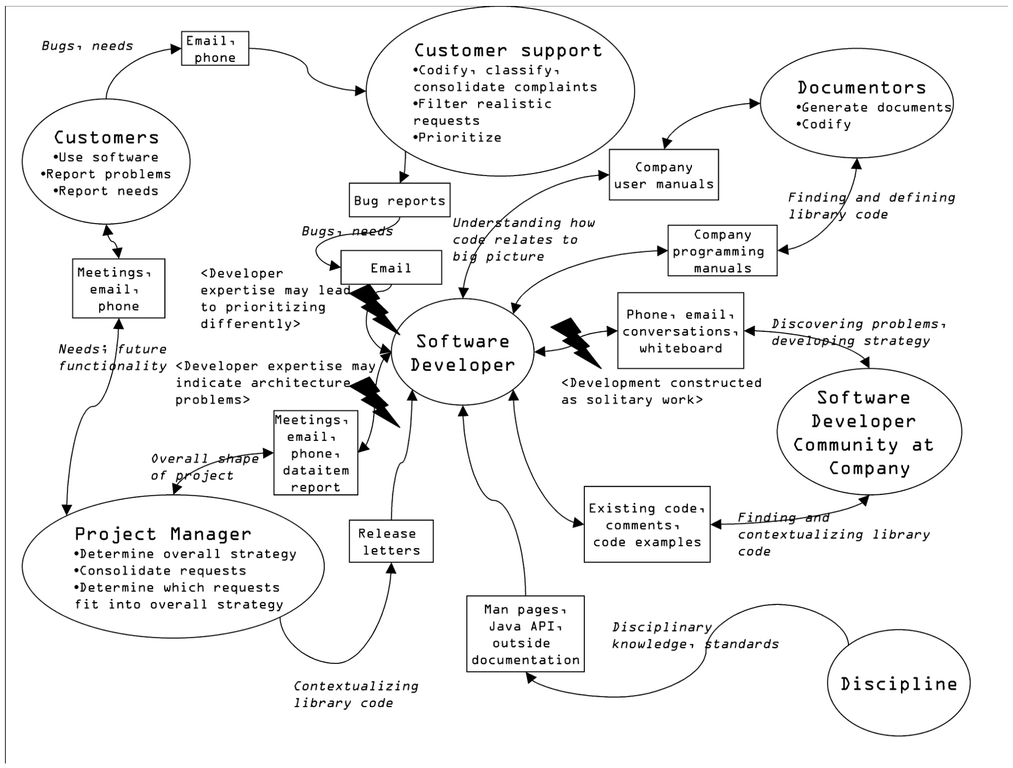
Genre theory gives me a meta-language with which I can describe software documentation in Contemporary development. Yet, there are still things for which the meta-language does not account. All the design decisions made by developers in the elevator or while playing golf are rhetorical forms of oral communication. At the same time, by the standards of industry guides, the idle chitchat is contempt for communication. genre theory is useful to describe oral communication in terms that validate its role in Contemporary development. Whereas my dissertation seeks to identify a need for that validation, genre theory still overlooks oral communication. The following uses the meta-language to demonstrate the omission of oral communication in each of Berkenkotter and Huckin's five principles.

Principle of dynamism. Dynamism states that the rhetorical forms develop from the actor's responses to situations. In other words, Traditional developers write documents that guide development; in addition, those developers adapt the documentation to fit into the constraints imposed by their preplanning work. However, Contemporary developers document while they develop and only document what is necessary. Consequently, the apparent absence of rhetorical forms—documentation—means no genres are formed from their agility. The three most important features of Berkenkotter and Huckin's Dynamism are (1) Forms develop from responses (2) stabilize development experience and (3) give meaning to what might be a chaotic mess with no commonality to focus a community of developers. Traditional documentation is precisely like that. On the other hand, while Contemporary documents are still generated because of Contemporary flexibility, those documents do not stabilize anything and are not what give meaning. The oral communication of software development is the rhetorical form to which Berkenkotter and Huckin refer: "Genres, therefore, are always sites of contention between stability and change. They are inherently dynamic, constantly (if gradually) changing over time in response to the sociocognitive needs of individual users" (p. 6). Oral communication fills those requirements set by genre theory. The office chitchat of the cubicle jungle drives development.

Principle of situatedness. Situated knowledge is the cornerstone of documentation activities in both traditional and contemporary software development practices. However, without additional rhetorical forms, the developers are reliant on simply documentation activities to record that situatedness. They look for code comments in the code of other developers; they review code libraries to acquire team standards; they write up requirements; they report progress to key stakeholders; they email each other to justify coding logic. However, the design decisions that should form from community responses to recurrent situations are not often documented. They are mediated and sustained as community knowledge via oral communication and other rhetorical forms.

Image 6.

Spinuzzi's Ecology Map from Software Development as Mediated Activity.


Bugs, needs
Email, phone
Customer support
•Codify, classify, consolidate complaints
•Filter realistic requests
•Prioritize
Documentors
•Generate documents
•Codify
Customers
•Use software
•Report problems
•Report needs
Company user manuals
Bug reports
Finding and defining library code
Bugs, needs
Understanding how code relates to big picture
Company programming manuals
Email
Meetings, email, phone
<Developer expertise may lead to prioritizing differently>
Phone, email, conversations, whiteboard
Discovering problems, developing strategy
Software Developer
Needs; future functionality
<Developer expertise may indicate architecture problems>
<Development constructed as solitary work>
Software Developer Community at Company
Meetings, email, phone, dataitem report
Overall shape of project
Existing code, comments, code examples
Finding and contextualizing library code
Project Manager
•Determine overall strategy
•Consolidate requests
•Determine which requests fit into overall strategy
Release letters
Man pages, Java API, outside documentation
Disciplinary knowledge, standards
Discipline
Contextualizing library code

Clay Spinuzzi's ecology maps demonstrate situatedness is a web of documentation and activity. Image 6 showcases an ecology map taken from *Software Development as Mediated Activity* (Spinuzzi, 2001). In his article, Spinuzzi focuses "on how artifacts (such as manuals, code comments, and the code itself) collectively mediated the developers' production and comprehension of code at three units of the same global corporation" (p. 58). In order to work in this environment, a worker must accept the situatedness and become a participant. Even though image 6 identifies "conversations," the ecology's collective mediation focuses on written communication. While oral communication is represented in the ecology as a form of situated mediation, the "conversations" are not included among the written communication Spinuzzi targets.

Principle of form and content. Rather than rhetorical forms like documentation that have conventions like headings, page numbers, and appendixes, the only conventions of oral communication are the developers themselves. In other words, the developer becomes the convention or agent of the genre. Clay Spinuzzi (2004) clarifies the "Agent" in genre theory: "the agent is 'genre' or 'genre ecologies' rather than human beings" (p. 114). While the traditional agent might have been the genre, the contemporary agent is the developers making design decisions on napkins at lunch time. Industry writer Alan Cooper (Cooper, 1999) takes a strong position against developer-agents generating form and content, without the appropriate documentation; Cooper suggests that developers should never deviate from the "blueprint" preset by the documentation. However, the very principle of office chitchat is a daily deviation from any documentation contemporary developers write.

Principle of duality of structure. Development documents are communication tools that outline guidelines, detail policies, and record a team's unified vision; teams produce these documents during pre-design stages of development. They are meant to be authoritative and reliable sources of information that guide development stages. However, projects rarely go according to plan and the documents are rewritten as fast as the project changes. At the same time, even while teams of writers, designers, and managers write these documents to act as development bibles (Adams & Rollings, 2003, 2007), the team actually writes all the changes to their own bible. In other words, the team is constantly rewriting their rules. Because of redundancy, many teams do not even keep their development documents. My dissertation seeks to bridge the paradox of document authority and author agency by highlighting oral communication as the bridge.

Principle of community ownership. Oral communication is clearly the Community Ownership of Contemporary developers; oral communication is a convention that signals Contemporary norms and ideology because Contemporary development values face-to-face communication. Ruping echoes the Contemporary Manifesto (p. 1) valued by Contemporary developers:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

These community values favor flexibility and an oral communication that stabilizes and gives meaning to Contemporary development practices.

Genre theory is perfect for the Traditional documents in Spinuzzi's genre ecologies. However, a similar map of Contemporary documents is incomplete without the oral communication.

VALIDATION WITH TWO THEORETICAL MODELS

I use two very different models to analyze software development documentation genres. My intent is to capture what rhetorical forms I expected to see in past development projects and assess the appropriateness of oral communication with professional developers. The following supports the two models I use.

The Model of Expectations and David Hailey's EUPARS (Exigency, Urgency, Purpose, Audience, Rhetorical stance, and Structure) model of genre evaluation serve different parts of the dissertation. I use the Model of Expectations to establish my genre expectations in the two postmortems. I use EUPARS to evaluate the new rhetorical forms in contemporary development workplaces. Where the Model of Expectations is more appropriate for postmortems, the EUPARS model is more useful for assessing whether the new rhetorical forms in contemporary development are actually genres that efficiently meet the needs of the community. The methodology section explains how I use the two models and the following is the theoretical support of the models.

The Model of Expectations

I encountered genre theory before I became familiar with software documentation. Consequently, I had the meta-language at my disposal when I began reading about documentation goals and standards. When I sought out professionals to compare their methods to the industry standards, I had specific expectations. Rather than simply look for the documentation industry guidebooks described, I was looking for the theoretical practice I could describe with the meta-language. Therefore, my procedures are modeled after specific expectations I have formed over the years.

I wanted my postmortems to be as close to my original expectations as possible. Therefore, I used research papers I wrote at the time to create the Model of Expectations. Fortunately, I wrote many papers about comparisons between my own documentation I wrote for my student projects and the documentation professionals described from their practice. Invariably, those research papers each utilize the meta-language of genre theory. I adapted those original research papers into my postmortems. I wanted to accurately draw on my expectations and conclusions.

The greatest commonality between all the research papers is the utilization of Berkenkotter and Huckin (1995). Their five principles of genre theory feature in all my early work: dynamism, situatedness, form and content, duality of structure, and community ownership. I broke up the commas and semi-colons in the descriptions of Berkenkotter and Huckin's five genre principles into 11 phrases. These are the 11 value statements on which I base my interview questions.

Of course, professional developers would not respond to questions like: "How is your community knowledge bound by recursive activities?" I needed my meta-language to breach the lexical walls of industry vocabulary because no one in industry would know what I was talking about without the industry vocabulary. The solution was to translate the 11 value statements without loading my dice—so to speak. Once again, those research papers I wrote were indispensible; I had already done the translation from meta-language to industry vocabulary. Neither my documentation practices nor those of developers met my expectations and my research papers articulated why.

The following table includes the 11 value statements in my Model of Expectations. I translate those value statements from the meta-language of genre theory into the language of traditional industry documentation recommendations.

Table 3

Model of Expectations Matched to Practitioner Wording

Model of Expectations	Worded for Practitioners
Dynamism • Genres are developed from actors' responses to recurrent situations. • Genres serve to stabilize experience and give it coherence and meaning. • Genres change over time in response to their users' sociocognitive needs.	Dynamism • The document forms as the team uses it and responds to it. • The document is a common resource for teams and is a foundation for the innovative solutions teams require. • The document changes with team decisions so that it is a decision making tool.
Situatedness • Genre knowledge is derived from and embedded in our participation in the communicative activities. • Genre knowledge is a form of "situated cognition." • Genre knowledge continues to develop as we participate in the activities of the ambient culture.	Situatedness • If it isn't documented then it didn't happen. The act of documentation is the formation of common knowledge. • Project productivity and long-term goals originate from documented knowledge. • Documents become more comprehensive and typified as the community collaborates and tests the document's relevance.
Form and Content • Genre knowledge embraces both form and content. • Genre knowledge is a sense of what content is appropriate to a particular purpose, situation, and time.	Form and Content • Form and Content conform to documented knowledge. • Document-based knowledge prompts decisions about relevant rhetorical content.
Duality of Structure • Genre rules inform activities that constitute social structures • Genre rules inform activities that simultaneously reproduce these structures.	Duality of Structure • Project planning, roles, and responsibilities match procedures established in the documentation. • The documented rules guide procedural and organizational decision-making.
Community Ownership • Genre conventions signal a discourse community's norms, epistemology, ideology, and social ontology.	Community Ownership • Philosophy of development methodology expressed through documentation and practice.

The EUPARS Model

EUPARS stands for Exigency, Urgency, Purpose, Audience, Rhetorical stance, and Structure. The model is best applied to text moved between Internet sites; insofar as all moves are not appropriate to the respective purposes, Hailey (2013) endeavors to use EUPARS as an assessment tool that measures the appropriateness of a move. He simplifies the objective of EUPARS: “figure out what the page is supposed to do and see if it does that” (p. 167). Where Hailey uses EUPARS on texts that already exist, I apply EUPARS to show traditional documentation is inappropriate in contemporary development and show that new rhetorical forms meet the needs of the situation.

I have been very careful to identify new rhetorical forms, rather than specify methods of oral communication. While I believe the latter is supportable, I adhere to the reasoning of David Hailey (2014) and cautiously avoid naming the new rhetorical forms: “naming the genre will not get us what we need for evaluating. Instead, we need to dissect the genres. Only when we are looking at the genres’ parts are we able to see if the individual parts work” (p. 167). Consequently, the Model of Expectations works well for documents I thought I understood and the EUPARS works well for rhetorical forms I seek to assess.

Table 4

Hailey's EUPARS model Defined

EUPARS term	**Definition**	**Core Question**
Exigency	A pressing requirement (p. 174)	Why is the text needed? (p. 168)
Urgency	The reason for the pressing nature of the requirement	How badly is the text needed? (p. 168)
Purpose	Seeing whether it does what it is supposed to for the right audience (p. 181)	What is the text supposed to do? (p. 168)
Audience	The audience is not named. It is described…Audiences are made up of collectives of different groups with different needs and expectations (p. 185) It isn't so much that you want to know who your audience is as you need to know *what they want or expect* and *how you can meet those expectations* (p. 193)	With whom are you trying to communicate? (p. 168)
Rhetorical stance	Rhetoric can be described as that part of communication intended to persuade, and all communication has a persuasive component. This persuasion occurs when you effectively meet the needs and expectations of the audience (p. 199), in the way the audience needs and expects	What are the audience needs and expectations? What do you hope to do with the reader? What is the appropriate stance to accomplish these goals? (pp. 168-169)
Structure	The structures of [communication] range from the document's physicality through real and conceptual metaphors to the document's ambiance or emotional tone (p. 202)	Should the text be bulleted, numbers, narrative? (p. 169)

Hailey does not see the table as a template or worksheet; however, Hailey does indicate that a table formatted like table 4 might be helpful. In fact, Hailey (2013) claims that after diligent inquiry into each of the six EUPARS terms, "you have enough information to instantly tell whether the text is appropriate for the situation" (p. 171).

CHAPTER III

METHODOLOGY

METHODOLOGY AND PROCEDURES

I used the following methods to discover and verify the rhetorical forms in contemporary development. In addition, I implemented genre theory as a theoretical framework to describe those rhetorical forms. However, it is not enough to use genre theory to describe my own development experiences; I also used the theoretical framework and a set of interviews to describe development in professional practice. In other words, I used a schematized meta-language to describe several different documentation practices. The following three research situations are key practices to showcase alternative rhetorical forms in contemporary development.

1. Development of an educational simulation for the American West Heritage Center in Logan, Utah. I preserved document samples and captured peer perspectives.
2. Coordination of learning-modules to supplement Department of Engineering online courses in Logan, Utah. I preserved a broad sampling of materials to demonstrate rhetorical forms in contemporary development.
3. Organization of interviews with seven professional developers in Sandy, Utah. I recorded interviews with seven local developers.

In addition to a meta-language and some development experiences, I employed two theoretical models—Model of Expectations and EUPARS model. The models insured that I deployed the meta-language the same way in all three research situations. In addition, they highlight common findings so I could triangulate my data.

Strategies of Inquiry

I used three strategies of inquiry in each of the three research situations. First, I used a theoretical framework to consistently apply the same lens to all the research situations. Second, I used well-documented project postmortems to recapture the path of discovery that led to my research conclusions. Finally, I used semi-structured interviews with seven professional developers to foster a frank conversation about efficient documentation practices. The framework supported the meta-language I used to both describe the postmortems and formulate the interview questions.

Inquiring with a theoretical framework. I used genre theory as a conceptual framework and meta-language with which I could describe the research situations I have selected. Of the many authors who describe genre theory, I find Berkenkotter and Huckin's (1995) five categories are the most useful in describing oral communication. While genre theory can be summed up in a single sentence by Carolyn Miller (1994), the five categories lend themselves to a more methodical analysis of rhetorical forms. Berkenkotter and Huckin break genres down into five component parts. The five principles create a meta-language by which I describe genre features and community actions.

I use two genre models: Model of Expectations and the EUPARS Model. I use the Model of Expectations to methodically describe the documentation I sought in each of my research situations. I use EUPARS to verify oral communication is a rhetorical form appropriately situated in contemporary development practices.

Inquiring with postmortems and interviews. Developers produce postmortems at the end of a project to process any lessons-learned. Scott Berkun (Berkun, 2005) explains:

> Team leaders must be committed to investing in the postmortem process. As things wind down, leaders should be asking people to start thinking about what went well and what didn't, even if it's just in the form of their private lists. A plan should be made for team leaders to collect these lists and build a postmortem report. The report should have two things: an analysis and summary of lessons learned, and a commitment to address a very small number of them in the next project. (p. 316)

Berkun (Berkun, 2005) also suggests what should be expected from a postmortem: "build a report based on what was learned, filtered through the consultant's expertise" (p. 316). As the documentation leader in each of the two research situations, I had all the communications, lists, documents, and reflections I preserved. These samples were sufficient information to capture an accurate postmortem. Most importantly, I rely on reflections I preserved, to build my conclusions in this dissertation.

My postmortems might relay trouble areas that I found but I also interviewed professionals to demonstrate that the postmortems and document samples supported what I claimed. I had a theoretical framework but interviews would prove I was not simply plugging aspects of the three research situations into Berkenkotter and Huckin's five categories. I used semi-structured interviews because, while I have a set of interview questions, I wanted to hear what the developers had to say about rhetorical forms, without guiding them in that direction with a series of structured questions.

Inquiring after pilot research. I have previously worked with professional developers. I have done site visits, reviewed proprietary documentation, and interviewed developers. However, where my extensive experience with professional developers reveals weaknesses in their documentation practices, I quickly discovered my potential participants were all constrained by standard industry disclosure contracts. Software development companies make non-disclosure a condition of employment; consequently, disclosure is grounds for disciplinary action. I have also signed non-disclosure statements in order to do the site visits and interviews I have done for the past few years. In fact, security would confiscate my electronic devices and some of those companies literally own my observations. It would not be appropriate to expose my participants to such legal and employment risk with their employers.

At the same time, those interactions served to help me feel out the extent of the problem and get a sense of direction as I explored documentation practices. I was also able to learn the parameters of a development project. Therefore, in an attempt to be sensitive to industry professionals with whom I have had interaction, I developed research situations that replicated industry practices and revealed similar weaknesses. I also conducted offsite semi-structured interviews so that my interview subjects could control the open-ended discussion.

Experimental Design

My data comes from both documentation samples and participant interviews. I have collected this data in three research situations—two postmortems and a set of interviews. Whereas, I gathered a healthy corpus of documentation samples for the two postmortems, I used theoretically-based interview questions to prompt data collection in the interviews with professional developers. In each of the three research situations, I used my Model of Expectations to assess the situation's match with genre theory and industry standards.

The following are the specific research situations I used in my research. I briefly describe the situation; I identify my role in the situation; I articulate how I implemented the situation in my methods.

Research situation one: american west heritage center (AWHC). Spring 2009. Under the faculty supervision of Dr. Brett Shelton, I served on one of three Agile teams developing a tour simulation of Logan Utah's historic American West Heritage Center. Development was done under the name of Dr. Shelton's Interdisciplinary Media Research Consortium (IMRC). The IMRC gathers funding to create student development projects that serve community organizations. My team produced documentation much like industry recommends.

I was one of four graduate students on one of three teams. Dr. Shelton closely directed the aggressive, contemporary project. My team maintained all documents on Google Docs. In this way, we could update the same document in real-time, even if we were each working in remote locations. I have extensive documentation samples to represent real-time development. I asked my three team members to share their perspectives with me. They each had very different previous experience with documentation and I was interested in how they perceived the documentation we wrote.

This situation will be a chapter that illustrates what I thought qualified as successful, contemporary documentation. However, the meta-language of genre theory helps me articulate why this was not actually the case after all.

Research situation two: USU engineering department. Summer 2009. Under the faculty supervision of Dr. David Hailey, I coordinated filming and development of an online, modular interface for engineering students at remote Utah State University campus locations. The team worked in such close proximity in a fast paced contemporary development process so that the absence of documentation was insufficient to account for the success of the project.

As the project coordinator, I was responsible for coordinating a film crew with four engineering courses. I was not project manager and was not involved with development. I coordinated the participation of six Engineering faculty, scheduling recording sessions, and making sure the interns were where they needed to be for filming, converting, and editing film footage to Flash learning modules.

Strictly speaking, the project did not generate sufficient internal software documentation. However, the student developers were successfully producing Flash files and coding the files into a website, despite the absence of traditional documentation. I could use the meta-language of genre theory to articulate why there was another rhetorical form involved and that the rhetorical form was oral communication.

Research situation three: software developer interviews. Spring 2012. I worked for a local Sandy, Utah software company. My workstation was in close proximity to a team of experienced, senior developers. The project manager led a very contemporary style of software development. The project manager often expressed to me his interest in hiring a technical writer to manage the documentation his team could not complete. I was able to secure permission to interview him and his team.

I prompted open discussion with a set of semi-structured interview questions, based on the Model of Expectations. I took each team member to lunch; we could discuss documentation in general and offsite, rather than in the context of their employment. This meant we did not discuss their specific employer's development and I did not need to see any of their work. Consequently, we did not need a non-disclosure statement to simply discuss documentation at a restaurant. In addition, the time I required would not cost the employer or impact project timelines because the interviews were during lunch break.

After the interviews, I reviewed the notes against my Model of Expectations. In addition, I included Dr. David Hailey's (2013) EUPRAS Model to assess the appropriateness of oral communication as a rhetorical form, based on the semi-structured interview results.

METHODOLOGY AND PARTICIPANTS

There were 11 participants in total between all three research situations. The participants play an important part in my methodology because the document samples all come from my own research projects. Consequently, my interpretation is suspect because my claim is that my documents meet my expectations. The inclusion of research participants necessarily offers 11 different perspectives about rhetorical forms in contemporary development. This balances the objectiveness of my analysis.

Participant Selection

There are two types of participants split between only two of my three research situations. The first four participants were involved in the first research situation: the American West Heritage Center simulation development. The remaining seven participants are the professional developers in the third research situation: the lunch hour developer interviews. The seven developers in the third research situation are meant to verify conclusions made at the end of the second research situation.

While there were four student developers in the second research situation, they did not engage them as participants. After all, they did not document anything and would not have relevant perspective on both documentation and oral communication. There are no participants in the second research situation because there was no documentation.

Participant assumptions. My interview questions initially posed a problem when I sought to instruct my participants; I did not want to tell my participants to explain how they use oral communication as a rhetorical form. Rather, I wanted participants to discuss their documentation practices, without knowing I expected to hear more about oral communication as a rhetorical form.

Therefore, I instructed participants about my interest in software documentation standards. This emphasized traditional, written communication and set expectations for the topic of discussion. While the participants were experts on industry practices for documentation, based on pilot research, I predicted the participants did not have documentation practices that matched industry standards. In fact, the participants discussed their documentation or their lack of it. When they had documentation they accounted for how their documentation differed from the standard.

Participant briefing. I was not interested in data that only showed what participants did not do so I guided the participants away from confessions about their poor documentation practices. I briefed developers about my specialty and my project. I did not want to be absolutely clear about oral communication in the beginning because I expected that information to emerge from the interview. The central assumption of my research was that developers with poor documentation do not work in a vacuum of communication; rather, they do something else. My semi-structured questions elicited open-ended discussions about either the role of traditional documentation or the role of another rhetorical form.

The briefing explained my interest in internal software documentation. I repeated the following to give developers context for my research:

> Software developers are frequently frustrated because traditional documentation rules do not work well in contemporary development

> methodologies. I do not mean to imply that it cannot work or that many of the values are not worth preserving. However, what exactly is “Agile Documentation” and how exactly does it work?

Participant protection and IRB. The Utah State University Internal Review Board (IRB) determined that this research is IRB Exempt. I only asked questions about participants’ documentation and development practices. There were no personal questions, no mention of real names, no disclosure of other participants, and no reference to identifying information. The IRB approval letter is in appendix 1.

Participant Perspectives from American West Project

The participants in the American West Heritage Center simulation development project were three graduate students from a class in which I was also a student. I selected these three students because they were on my own development team and were the most familiar with the documentation they wrote with me. The conversation I had with each of them yielded a perspective about documentation that was meant to offer some credibility to my own conclusions about the project documentation.

I have ascribed each of my graduate student peers a name to sustain their anonymity. I used the first three letters of the NATO phonetic alphabet—Alpha, Bravo and Charlie.

Alpha—mobile unit programmer. From the beginning of the project, Alpha had a lot of energy and a lot to offer. Alpha actively contributed to the design documentation. Alpha generated his own documents, rather than only using documents I generated. Alpha was responsible for most of the game art for all three game modules on both platforms. In addition, Alpha was responsible for the GPS mobile unit coding. In fact, Alpha was really motivated to shoulder the GPS mobile unit programming all alone because of the speed with which Alpha could do it and because of the pride Alpha felt in doing it well.

Bravo—PC software programmer. Bravo was quick to pick up on the design tool we used to develop the tour game for the PC platform. Bravo completed a lot of work on both the general inventory interface and the avatar navigation controls for all three project teams. Consequently, Bravo became the PC programmer for our team. Bravo tended to work in a hermit-like style so that there was no communication; then Bravo would emerge from isolation with an enormous amount of work completed. Bravo confided in me that he had been required to maintain documentation in the past and that it was a frustrating chore. Bravo simply did not want to waste time documenting when Bravo could be developing. Not surprisingly, Bravo asked for dialogs, object descriptions and narrative scripts, without relying on the documentation. Bravo could have benefited a great deal from reviewing the design documentation. However, despite Bravo's key role developing the PC platform of the game, Bravo apparently did not look at the design documentation until I directed him to it in April 2009.

Charlie—PC software designer. Charlie recorded research and design work in the documents. While Charlie expressed unfamiliarity with the development technology, Charlie was very valuable with research, with supporting Bravo's programming and a lot of little careful things that made the documentation better and the development easier. Charlie was the least familiar with documentation practices and goals.

Utilizing three conversations. My peer students offered valuable perspectives in a postmortem that would otherwise be introspective. However, I was not objective or rigorous in the selection of my three participants; they were the only sample that could answer for documentation on my team because they were the only other people on my team. Fortunately, I only wanted perspectives that would counter my own interpretation of my own documentation.

I recognize three limitations with the conversations: (1) these are more like brief conversations than they are interviews, (2) my own presence on the project must have necessarily made my peer-students sensitive to the importance of documentation and (3) the articulation from each of my peer-students was not equal due to differences in both experience and background.

After Bravo requested the design information that was already documented, I became very interested in what my peers thought documentation was and whether they thought differently after experiencing what I felt was a best practice situation. I prepared three very general questions and arranged to meet with my peers; I wanted them to have the chance to answer the questions without trapping them on the way out of class. My objective was simply to discover if specialists unfamiliar with documentation would define documentation the way I did—without the meta-language and imposing genre theory everywhere I look.

I let them choose where they wanted to discuss documentation with me. I met Bravo and Charlie each at the same university lab. Alpha was busy and asked to discuss documentation at Alpha's home. I limited each conversation to approximately 20 minutes. The small timeframe was sufficient to elicit clear definitions during a formal conversation. The following are the three conversation points I planned.

1. What did you think what a General Design Document was before we started the semester?
2. Can you describe for me the role of a General Design Document in a development process?
3. Now that we are 9 weeks into development, what do you think it is now?

I was confident I would get very significant answers that would support my impression that the American West tour game documentation was a best practice example of how traditional documentation practices can be as dynamic as contemporary workplaces.

Participant Interviews with Lunch Time Software Developers

I interviewed seven professional, contemporary software developers. They are senior developers, knowledge experts, and very competent with their job responsibilities. The seven developers are all on the same team and consist of software engineers, database developers, and web developers. Their specializations are the strength of their comprehensive project strategy.

Interview participant inclusion criteria. I wanted a developer that was a small team with very contemporary methods. In addition, the team must be too small to afford a professional writer so that the developers needed to manage the documentation themselves. Teams that have a professional writer do not experience the documentation problems I have identified in my research. After all, a documentation expert does really well at generating documentation, no matter what the standards are.

I tapped into my network of developers to find the right subjects. I had a pool of four teams: two did not qualify, one did not cooperate, and the last is the group I chose. I included the following inclusion criteria when I recruited developers:

- I need a flexible, contemporary team of software developers. There is some latitude with the amount of methods by which I label "contemporary"; however, developers who use more traditional methodologies will not meet my requirements. I need contemporary developers with very quick, flexible methods.
- I need 5-10 developers; that number can include the project manager. I do require the project manager. It would be best if the subjects engage each other during the course of the workweek. This would be the best way to learn about the communication they employ.
- I need a contemporary team that does not employ a technical writer. A writer would do the documentation and probably does the documentation very well. Rather, I need a team that is responsible for writing the documentation. My expectations are that such developers have resolved their documentation methods, even if they do not match traditional documentation standards.
- I need local developers who work together in the same office. The alternative is remote developers and based on experience with the American West Heritage Center simulation project, I believe that a remote developer confounds my results; remote developers must write documentation in order to develop together—otherwise they would have to stay on the phone all day with each other.

These requirements set clear expectations so I could rely on using the information once I acquired information.

Participant payment. Developers track their time to specific revenue-driven projects and they capitalize time spent on operations that do not generate product. Consequently, their willingness to cooperate with me costs them money. I did not have grant funding so I was limited in how much I could compensate developers for their interviews. Consequently, the solution must not cost the developer time and money.

I scheduled time with the developers during their respective lunch breaks. In addition, I paid for their meal at a quiet, local restaurant. We talked offsite, without costing the company any money or costing the team project time.

Participants and recording. I used a Livescribe Pen to record my interview notes. The Livescribe pen technology uses a special notebook to capture, use, and share audio lectures and interviews. The Livescribe pen captured an audio recording of the interviews.

After the interview, I used the pen to cue the audio of each session by tapping the respective notes with the digital pen. Therefore, I could search the audio recording by simply reviewing my notes.

I transferred the audio notes to a Livescribe software program. I deleted the notes from the pen. My computer is a password protected Macintosh that uses an encrypted backup system. The notes would be securely stored on my computer. Finally, I could use the Livescribe software program to annotate my notes.

I could make the interview notes available to interviewees on a secure Livescribe community website, if the participants requested a copy. They received official consent paperwork (see Appendix 2) that indicated the availability of the recording. The community website is the property of Livescribe and the security is managed by Livescribe. The default setting for any recordings I upload to the website is "private." Consequently, I have no plans to mark the recordings as public. I can set independent passwords to each recording so I can restrict access to only the interviewee.

Procedures: Semi-Structured Interview Questions. I matched an interview question to each of the 11 value statements in the Model of Expectations. The questions assume that I will have access to documentation during the interview, even if we did not actually use specific documentation during the interviews. We wanted to decrease the proprietary disclosure required by the interviews and documentation would have necessarily increased risk of proprietary disclosure.

Each of the 11 questions was a starting point for an open-ended discussion. I wanted participants to answer questions about documentation and to elaborate as necessary. There were four additional icebreaker questions to get some general information and to set the participant at ease. Table 5 showcases my interview questions, organized according to the five principles of genre theory.

Table 5

The Interview Questions Organized According to Genre Principles

Interview Questions
Ice Breakers 1. How long have you been operating? Can you explain how you are agile and why you choose agile methods? 2. What is your development philosophy? What development standards do you value most? 3. What does a typical day look like for your own role and responsibilities? 4. If it is not documented it didn't happen. The absence of documentation is a failure to communicate. What kind of response do you have for these two statements?
Dynamism 1. I'm looking at these two documents. Describe how they fit into your workflow--both writing and using them. 2. Describe a situation in which you needed the documentation to resolve the team's confusion about the design. 3. Some professionals refer to live documents or organic documents. Describe a situation in which an organic document a) evolved with the development cycle and b) informed the development cycle?
Situatedness 1. What changes when design concepts are written down, rather than merely "known" by the team. 2. Industry writers suggest documentation is a bible or blueprint or governing document. Vision documents. Guiding documents. What kind of governance does documentation have in your shop? 3. Aggregation. Compounding. Synthesis. These are all words that suggest a whole is formed by the sum of its parts. Can you explain how a document is the sum of development activities?
Form and Content 1. Books have recommended outlines and there are templates available online. What kind of adaptations do you make when you measure your work against industry samples? Describe an experience when the recommendations didn't fit right. 2. In addition to standards, there are other things that don't fit right. You make decisions about direction, design, procedure, and operational details that don't always fit right. I'm looking at this document sample; can you tell me some tough decisions that involved this document?
Duality of Structure 1. I've observed that your workplace is organized to meet specific needs. In what way did you use documentation to identify those needs and record your business solutions? Can you tell me how well the documented business solution works for everyone else in the company? 2. How much do you draw on documented business solutions when you have a meeting? Can you describe what that would/should look like in a perfect world?
Community Ownership 1. You told me about your development philosophy. Now that we have discussed documentation so much I wonder how you connect your documentation to your philosophy.

METHODOLOGY AND ANALYSIS

I used a theoretical framework for analysis, as well as some advanced technology for my interview notes/recording. I used two models formed from genre theory.

Central to their framework of genre are Berkenkotter and Huckin's five key principles: Dynamism, Situatedness, Form and Content, Duality of Structure, and Community Ownership. I broke down the five principles into 11 value statements. These value statements constitute a Model of Expectations with which I described each of the three research situations.

I also used a second model to assess how well the documentation I found in the interviews actually worked. In his forthcoming book, David E. Hailey (2013) uses genre theory to identify the Exigencies, Urgency, Purpose, Audience, Rhetorical stance, and Structure of online genres. His EUPARS Model assesses how appropriate a particular document is for the particular situation. The EUPARS model was useful in the final research situation to highlight how appropriate oral communication was as a rhetorical form in the contemporary development workplace.

Data Analysis and Interpretation

While my research methods are not ethnographic, my analysis and interpretation borrow a great deal from ethnographic research. Specifically, I use the methodology detailed by Emerson, Fretz, and Shaw (1995) to not only systematically turn my notes into conclusions but also guard against my conclusions speaking for the participants.

Method that identifies participant voice. Emerson et al. (1995) outline three steps in their methodology. Their objective goes beyond a system that organizes ethnographic data but is a procedure for making objective conclusions that originate in the voice of the participants. They encourage researchers to write jottings on the spot, detail the jottings immediately after and code the field notes later.

- Jottings: jottings translate to-be-remembered observations into writing on paper as quickly rendered scribbles about actions and dialogue…will jog the memory later in the day and enable the field worker to catch significant actions and to construct evocative descriptions of the scene. (p. 20)
- Field notes: write up their observations into full field notes…turning recollections and jottings into detailed written accounts that will preserve as much as possible. (p. 39)
- Coding Field Notes: sift systematically through the many pages of field note accounts…the ultimate goal is to produce a coherent, focused analysis of some aspect of the social life that has been observed and recorded. (p. 142)

An important feature of this methodology is delaying the insertion of personal interpretation for as long as possible. So while the goal of field notes is "looking to identify threads that can be woven together to tell a story about the observed social world" (p. 142), the authors discourage anything other than privileging the voice of the participants. The authors write: "even seemingly straightforward descriptive writing, is a construction. Through his choice of words and method of organization, a writer presents a version of the world" (p. 66). There is always a lens in ethnographic research. The researcher's interpretation is always in the way. Consequently, the authors divide up the formation of conclusions into the three steps to reserve analysis for as long as possible.

Even when the researcher finally begins to code, the authors still encourage a coding practice that highlights the participants' meanings. In other words, Emerson et al. (1995) believe that any code words are derived from the words of the participants. Consequently, the field notes are a "data set" (p. 144) that presents the key words the researcher must identify before tracing any patterns in what the participants say themselves.

Using ethnographic values with interviews and developers. Jottings and field notes are how I managed my own data. I used my Livescribe pen and notepaper during my interviews. Consequently, my notes are the jottings to which Emerson, Fretz, and Shaw refer. The audio recording captured by the pen is directly synced to the jottings but I still took the time to write field notes after each of the interviews. I wanted to be sure that I preserved any thoughts or insights I had during the interviews, rather than project back or forget what could turn into useful findings. The pen made it very easy to write useful field notes that were closely synced to what participants actually said in the interviews.

Where Emerson et al. (1995) suggest I identify threads and tell the participants' stories, I was looking to see what threads I could find about documentation specifically. I expected to find other rhetorical forms and coded those as well. The end result was a picture that included oral communication in contemporary development workplaces but involved a lot of other little rhetorical forms.

American West Postmortem Document Sampling

While we developed the American West Heritage Center tour game, Alpha, Bravo, Charlie, and I wrote all our documentation in a browser-based software called Google Docs. Google is an online corporation that released a revolutionary web search engine in 1998. The Google web search engine uses a proprietary algorithm to return search results for nearly any information a person might seek. Since 1998, Google has extended its mission of open access and free information to communication tools, media portals, storage space, and business solutions. The Google Doc is a free word processor document to which the owner can invite several contributors. The shared document is a site of co-authorship where every contributor has equal editorial powers.

Rather than email a single document and wait to see changes until the document circulates through the team, a Google Doc is not circulated and all changes are visible in real-time. Consequently, four graduate students can write the same document together, at the same time, no matter how remote they are when they do it. In fact, the absence of a common workspace necessitated a documentation solution like Google Docs.

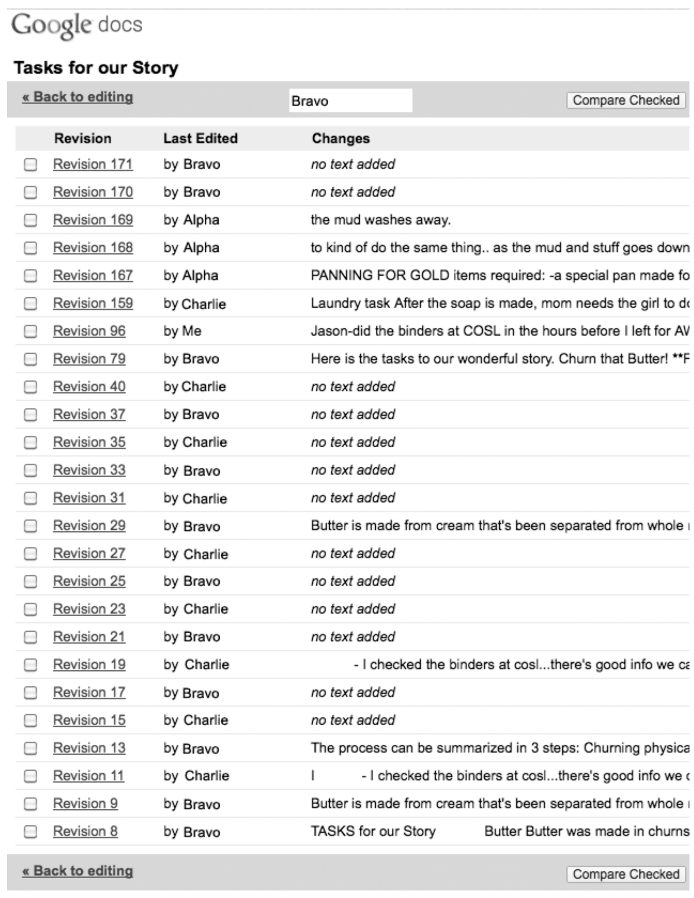
The bi-product of real-time co-authorship is a comprehensive awareness of project progress. Consequently, the Google Docs we all shared are a body of evidence that demonstrates how we worked so closely together, even though our development activities took place remotely.

Example of remote collaboration. The "Tasks for our Story" document was written at the end of January and represents some original design work. We each researched 1917 farm chores we could incorporate into our module of the tour game. We recorded our findings in the same document. We updated the farm chores with not only greater detail but also broke the chores down into programmable steps.

Image 7 showcases the 171 revisions made to the document. We did not take turns editing the document. Rather, we simply turned it on and wrote. The Google doc identifies which contributor is editing the document at the same time and it tracks all changes in a revision history.

Image 7.

A Record of 171 Revisions.



Google docs

Tasks for our Story

« Back to editing | Bravo | Compare Checked

Revision	Last Edited	Changes
Revision 171	by Bravo	*no text added*
Revision 170	by Bravo	*no text added*
Revision 169	by Alpha	the mud washes away.
Revision 168	by Alpha	to kind of do the same thing.. as the mud and stuff goes down
Revision 167	by Alpha	PANNING FOR GOLD items required: -a special pan made fo
Revision 159	by Charlie	Laundry task After the soap is made, mom needs the girl to dc
Revision 96	by Me	Jason-did the binders at COSL in the hours before I left for AV
Revision 79	by Bravo	Here is the tasks to our wonderful story. Churn that Butter! **F
Revision 40	by Charlie	*no text added*
Revision 37	by Bravo	*no text added*
Revision 35	by Charlie	*no text added*
Revision 33	by Bravo	*no text added*
Revision 31	by Charlie	*no text added*
Revision 29	by Bravo	Butter is made from cream that's been separated from whole
Revision 27	by Charlie	*no text added*
Revision 25	by Bravo	*no text added*
Revision 23	by Charlie	*no text added*
Revision 21	by Bravo	*no text added*
Revision 19	by Charlie	- I checked the binders at cosl...there's good info we ca
Revision 17	by Bravo	*no text added*
Revision 15	by Charlie	*no text added*
Revision 13	by Bravo	The process can be summarized in 3 steps: Churning physica
Revision 11	by Charlie	I - I checked the binders at cosl...there's good info we c
Revision 9	by Bravo	Butter is made from cream that's been separated from whole
Revision 8	by Bravo	TASKS for our Story Butter Butter was made in churns

« Back to editing | Compare Checked

The revision history shows that Alpha, Bravo, and "Me" each contributed 1917 farm chores. In addition, the revision history shows the numerous revisions Charlie made to Alpha's contribution. Charlie deleted text, even while Alpha was editing; they worked in tandem as if they were co-authors in the same office.

Gathering american west documentation samples. I chose my documentation samples from my Google Doc collection. Every Google Doc is stored in the Documents tab of an individual's Google user account. From that account, a person can check email, manage a calendar, and share documents. The list of Documents includes both documents I own and documents I share with others.

I wanted to select three kinds of document to showcase the dynamism, situatedness, form and content, duality of structure, and community ownership of the game tour documentation. In addition, I wanted the samples to link to what Alpha, Bravo, and Charlie told me about the project documentation.

1. I wanted a document that met industry standards and simply transmitted knowledge. The document should have changed the least over the course of the project. This would be the document that fails to meet my expectations.
2. I wanted a document that organized our remote team and provided clarity. The document would showcase real-time changes by a remote team.
3. I wanted a document that would capsulate the success of the entire project. The document should have a strong edit history to show the level of collaboration.

I want to show that the American West Heritage Center documentation met my expectations for how the genre of traditional software documentation should work in contemporary workplaces. I wanted to demonstrate that with the right community ownership, the traditional documents can perform as recommended by industry guides. I wanted documents that showcased this success; however, my student peers articulated the success I sought to demonstrate with the selected samples.

Online Engineering Modules Projects Document Sampling

The Engineering Modules project followed the American West project by mere days. I needed to see if I could repeat the documentation success of the American West project. My team of graduate student tour game developers had documented so much and if I could do it again then the interns of the online module team could write a similarly rich quantity of documentation.

To my surprise, after three months of online module development, the interns had not produced any documentation. As in the case of my pilot research, they were too busy keeping pace with their development cycles to spend time documenting. Consequently, I thought I was left without any documentation but I discovered that I was wrong. The interns did documentation activities with different rhetorical forms. None of the rhetorical forms they used matched any industry standards; sticky notes, emails, schedules, scraps of paper, and little memo files on the computer are definitely insufficient forms of traditional documentation. In fact, they constitute a documentation failure, according to industry standards.

If the success of the project was an indicator of efficient communication then there was a rhetorical form that I clearly missed when I gathered the samples. Communication was so simultaneous with development that the only missing rhetorical form was the oral communication that bound the participants, the little samples of documentation, and the recursive situation.

Example of diverse rhetorical forms. As the interns walked from the Department of Engineering building back to the Department of English building, they would discuss what went wrong or they would identify a good practice that made a great deal of difference. They would make decisions about the editing and identified software features that would speed up the editing.

Unfortunately, I did not plan on capturing oral communication samples; I was looking for written documentation. Therefore, the only samples I have from the project are the sticky notes, emails, schedules, scraps of paper, and little memo files. However, these minor rhetorical forms are still artifacts of the oral communication between the interns. A sticky note affixed to a cassette tape would indicate the editorial status of that tape. And a memo file in the same folder with a video file would be the meta data that described why the video was stored but not implemented.

The following is a procedural checklist the interns used to make sure they had all the necessary equipment with them before they went to film the live courses. This inventory was on a sheet of paper the interns left in the camera bag.

Table 6

The Film Equipment Inventory Used for Engineering Courses

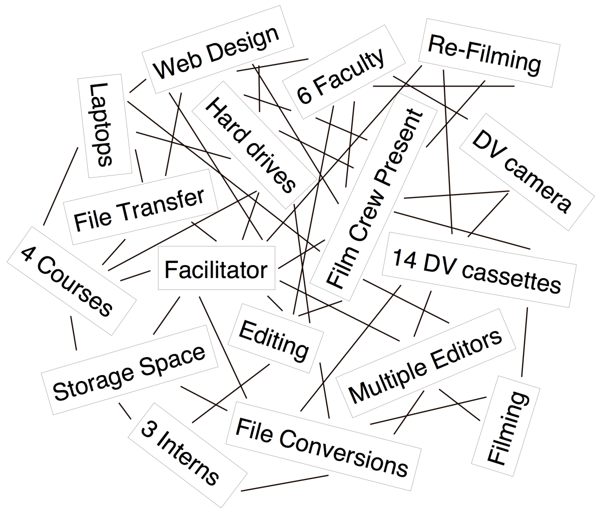
Film Equipment Transport Inventory
□ Camera Bag
• Camera
• Battery
• Power cord
• Tapes
□ Microphone Case
• Receiver
• Transmitter
• Microphone
• Power cord
• Feed Cable
• Extra Battery
□ Laptop Bag
• Laptop
• Power cable
• Mouse
□ Tri-pod
□ Extension Cord
□ Power Strip
□ Headphones
□ External Hard drive and cable

An inventory list is an artifact of lengthy conversations about what was needed for each recording session. The list is an artifact of debates about whether the interns had all the right equipment before they left for each recording session. Consequently, even while rhetorical forms like the inventory list are not samples of the actual oral communication, the sticky notes, emails, schedules, scraps of paper, and little memo files were all simultaneous rhetorical forms, along with the oral communication.

Gathering online engineering modules documentation samples. I retained all the rhetorical forms I could find. When the interns were done for the day, I would go into the development room and grab any written document I could find. I scanned documents and returned the documents to their appropriate location in the room.

Image 8

Visual Portrayal of the Complex Recursive Situation



The documents I selected were meant to showcase the simultaneousness of communication in that project. There were so many moving pieces and the little pieces of documentation were far too insufficient to keep things from falling apart. Image 8 portrays the complexity of the recursive situation I coordinated. There were too many important intersections for a sticky note to stabilize the experience. A little text file hidden the file architecture of the computer was too obscure to foster any kind of coherence or meaning. The samples I choose are meant to highlight the rhetorical form that actually turned potential chaos into a fabulous success.

CHAPTER IV

POSTMORTEM: AMERICAN WEST HERITAGE CENTER

CONTEXT FOR AMERICAN WEST GAME POSTMORTEM

The American West Heritage Center is a historic farm in Logan, Utah. In spring 2009, graduate students from both Dr. Brett Shelton's IMRC initiative and Instructional Technology course worked in teams to develop educational tour games for the farm. The students were not experienced developers; with the exception of myself, the students were not experienced with software documentation either. The three other graduate students on my team—I call them Alpha, Bravo, and Charlie—worked with me to both develop and generate documentation. The documentation of that project is perhaps a best practice example of documentation for contemporary development methods. In fact, I was thrilled to mentor my three graduate student peers in the dynamic documentation practice that facilitated our project's success. We created a small community that was literally situated in the documentation. At the time, I was extremely committed to the traditional documentation rules and quite earnest to implement them for the American West Heritage Center project. I was confident that with the right community ownership I could generate documentation predicted by genre theory.

I did not disappoint myself. We generated extensive documentation. More to the point, we wrote documentation that was not merely an archive of design elements or documents irrelevant to design decisions.

This is a Postmortem

This chapter is a postmortem of the American West Heritage Center development project. During the student project, I was still looking for written communication that met traditional rules, yet worked in contemporary practice. I was particularly interested in identifying what the "duality of structure" looked like in documentation practice. In theory, each time an agent rewrites a document, there is an exchange between genre and writer that both sustains and reproduces the written communication. However, based on my experience and the developer with whom I had spoken, I did not know what duality of structure was supposed to look like—if it was at all possible in the first place.

This postmortem seeks to demonstrate what dynamism, situatedness, form and content, duality of structure, and community ownership all looked like in practice. While we were students, rather than professionals, we created a series of documents that met industry standards and adapted to our needs at the same time. Most importantly, while Alpha, Bravo, and Charlie understood design documentation as merely a method to communicate and archive design concepts, they came to see documentation as a dynamic hub of communication that plays an active role on the team.

Postmortem as a snapshot in time. In an attempt to preserve the expectations I had and the conclusions I made, much of the text for this postmortem was taken from a report I wrote at the end of Spring 2009. I have since learned the project is not a best practice of documentation and subsequent chapters will explain why. The purpose of this chapter is to accurately capture documentation from Spring 2009 and present samples that showcase how I met my expectations. I include remarks from my student peers to support the objectivity of the conclusions I made at the time.

CONTEXT OF THE AMERICAN WEST GAME

While Alpha, Bravo, Charlie, and I worked on the tour game, Bravo made a request that prompted me to think about the project documentation and what my peers thought of the documentation. While the game was fully elaborated in the design document, Bravo asked for verbal explanations of elements already detailed in the general design document. In the first week of April, Bravo asked for item descriptions that he could input into the PC version of the game. He wanted narrative dialogs and descriptions for the characters. He wanted to know about recent modifications to the plot, dialog, and code I had recently made; I had added a plot element, including garden seeds. Finally, Bravo wanted an account of how the conclusion had been resolved in the mobile unit. Of course, all these things were already detailed in the design documentation.

A Context of Documents

The purpose of documentation is so that developers do not use their time answering the long string of questions Bravo asked. Rather, Bravo should review the documentation. It is worth noting that I could have answered his questions without deferring to the documentation like I did; however, that would have consumed much more time than was necessary. There was enough documentation for enough work that a quick conversation was not possible. In fact, best practice suggests that rather than use precious development time verifying, confirming, and repeating elaborations at every inquiry throughout a week, teams should be able to access any of several central documents, without necessitating the verbal repeat of elaborations, reflections, and solutions.

Our team documentation was not nearly as elaborate as many professional development companies but it was still sufficient for Bravo's needs. Specifically, a Salt Lake City developer I once interviewed boasted over 500 pages of documentation for each development project. They had multiple teams, over 200 employees and simultaneously produced two or three development projects every year. On the other hand, we created approximately 18 documents in total; many of the documents were collaborative workspaces for the team to hammer out game problems or fill out narrative details. All told, the documents totaled a little over 100 pages. There was all the information Bravo needed to complete the game for the PC platform.

The Context of the Actual Game

The development for the American West Heritage Center tour game began in early February 2009. The American West Heritage Center is a historic farm located in Logan, Utah; the farm provides local elementary schools the opportunity for field trips and a seasonally open farm for people who seek to explore the historical interpretations represented on the farm. The game permits children to take GPS-guided tours of the farm while playing three different farm adventures.

The farm site is the target of the four historical interpretations: 1917 farmstead, Pioneer settlement, Cache Valley Trappers, and Shoshone Native Americans. Different parts of the American West Heritage Center showcase different features of these four interpretations. The 1917 farmstead is by far the strongest interpretation; the farmstead features a farmhouse, blacksmith, chicken coop, etc. The student team to which I was assigned was also responsible for the 1917 farmstead interpretation.

There were several communication activities used to develop the game. As graduate students in a classroom, there was no common office space in which we collaborated every day. We worked remotely at asynchronous times of the day (and night)—from home or from computers in Dr. Brett Shelton's IMRC research lab. The only time many of us engaged face-to-face was in the classroom once a week. Consequently, communication was oral, email, phone, web chat, and shared online documents. Bravo kept up with much of the communication activities; however, Bravo neglected the shared online documents. Therefore, Bravo neglected the remotely situated team's key communication activities.

Team organization. The team divided up into discrete roles as we made progress towards development. Insofar as Alpha and Bravo were so comfortable and confident with programming, they naturally took on programming responsibilities. Of course, the majority of my time was spent documenting, even if I worked closely with Alpha on coding and bugtesting. However, there are advantages and disadvantages to the discreet roles the team used. On one hand, the team members were very efficient as everyone focused on their own skill sets. In addition, the team was able to trust each other as the team met deadlines in the contemporary development cycle.

At the same time, there were disadvantages too. Chiefly, the team was not able to adapt to the absence of one member; in fact, I was out of contact with the team for a week and they had to scramble to complete tasks they were not prepared to complete without me. However, the team came to an agreement that the disadvantages were worth the high level of efficiency afforded by discrete roles. In other words, dividing up coding responsibilities quickly became difficult; as inexperienced developers, we were not able to find a natural way to distribute programming tasks with the same efficiency as we found with specialization.

The Context in Three Phases

There were three phases on two different platforms—PC computers and mobile handheld units—in the space of four weeks.

- Design and Development
- Quality Assurance Testing
- Refinement and Unification

In an attempt to demonstrate the scale of the project and the difficulty of meeting Bravo's requests, I will describe more about the project's scale.

Design and development. The three teams were assigned to three of the four historic interpretations. Each team had four or five graduate students. On February 10, the teams were all instructed to have a design document from which teams could manage their ongoing development. Every class—once a week—after February 10 required a prototype that could showcase weekly progress. The weekly prototyping did not mean that the design work was complete and that there was nothing to document; rather, weekly prototyping happened in tandem with new implementations and revised designs. The documentation was constantly changing and we were constantly challenged to stay in sync. We relied on the documentation as a hub of communication. Throughout that time period, my team met once a week to get our bearings and set goals but the work we did was always remote and always communicated through the documentation.

Quality assurance testing. By mid-March, the teams started their own quality testing. Teams went to the historic farm with their GPS units and tried to break their game. I started a bug tracking spreadsheet online. We filled that document with bugs and logged the solutions, as well as dates when we implemented the solutions. After the third week of beta testing, the teams all exchanged games so that a second team conducted third party quality assurance tests. The fourth week involved a second round of third party testing so that all three groups had tested all three modules of the game. Not only did this fill up the bug tracking document but the activity in the design document increased as changes were made to implement the feedback.

Refinement and unification. At the beginning of April, all three teams were making final changes. In addition, the three teams began the awkward process of merging the different modules together for a unified game experience. This required interface standards, art standards, and compression standards. Any independent work was subsumed by group collaboration; the importance of documentation increased in value as the teams adapted their design to the standards set by large-scale consensus.

PARTICIPATION IN THE AMERICAN WEST GAME

The American West Heritage Center tour game was not a research study. Rather, it was a student development project. Through the course of that project, the primary goal was a tour game on two platforms. Towards the end of the project, Bravo had a series of questions about the game, specific narrative, and character details; he did not know that everything was documented. At that point, we suggested he look at the documentation and then ask questions. He had no further questions but only registered his amazement at the work we had done. Only then did I begin to think about the documentation we had produced. I was confident I could describe our documentation practices with genre theory; however, I wondered whether my peers noticed anything different about the documentation they had written with me.

I kept every version of every document. I kept all project assets. I preserved our email record. I saved online chats. Finally, I asked each of my three peers about our documentation practices and noted their responses. Consequently, not only was the project's documentation both successful and thoroughly preserved but my three student peers confirmed our practice was as unique as I thought.

My Role in Both the Development Project and the Methods

Three teams, with four or five graduate students, developed the tour game. I was on a team of four. At the beginning of the semester, my own team revised the narrative of the game multiple times. In each case, the narrative became simpler. The team left me with the responsibility of writing the narratives, as well as the revisions. On February 3rd the class was given the task of presenting a design document to capsulate the three weeks of rapid planning. I produced 20-single spaced pages of design documentation before February 10th. By the end of the semester, Alpha, Bravo, Charlie, and myself increased the design document to 47-single spaced pages.

Other communication responsibilities. In addition to the documentation, I managed the communication of the team. At the very beginning of the project I set up an online project site to share documents. At the beginning of March, we discovered that our Photoshop and Illustrator files exceeded file transfer size limits for the project site. Consequently, I quickly generated an FTP alternative through my own Internet provider. Our team had our own FTP server, with our own password; in addition, the team had one gigabyte to fill, without file size limits.

I played a larger role than simply the documentation specialist. For instance, I spent time in the Utah State University Library's special collections researching the Wyatt family from Jeannie Thomas's folklore thesis (1987). The American West Heritage Center actually relied on an interview in that thesis paper for their historical interpretations of the Wyatt family's 1917 farm life. There were many details about the family—names, dates, and history—as well as interesting vignettes that we used to enrich the game.

Coding responsibilities. I have limited programming experience that was adequate for the development of the American West Heritage Center's tour game. However, I did feel I could serve my team better by focusing on the generation of documentation, as opposed to slowing down my team with my amateur scripting and coding. I generated a bugtracking document as I started conducting tests and recording the results of tests conducted by other teams. I managed the bugtracking and completed 54 out of the 76 bugs on my own. Admittedly, the 54 bugs were not code intensive—Alpha was so familiar with his own coding methodology that he took hard coding; however, I still had to learn to find my way around Alpha's coding methodology and learn the coding logic he had implemented.

I was solely responsible for coding GPS zones for the tour game's mobile platform. Unlike the game's modules built by the other teams, the 1917 farm featured replica structures that were clustered close together. Consequently, I had to code very specific GPS zones that matched physical structures on the planet's surface. I returned to the farm several times to test my latitude and longitude coordinates with the farm structures.

THREE SAMPLES FROM AMERICAN WEST GAME

Each of the following three samples showcase the documentation I expected to create during the tour game project. In addition, I transcribed the documentation definitions my student peers gave me during our formal conversations. While I chose the first sample to represent standard industry documentation, I chose the remaining two samples to illustrate the documentation I expected.

A Standard Sample of Traditional Documentation

Dr. Shelton started the student project with the task of finding activities for the game tour. My group of four graduate students researched activities relevant to 1917 farm life in Logan, Utah. Research materials were in the Utah State University Merrill library special collections and in Dr. Shelton's IMRC research lab. We identified the precise steps to churn butter, wash the laundry, shop at the general store, and make soap. Precision was key because each step would become a coded, executable action in the game tour. The next assignment was to think of the objects necessary to perform these activities and the locations in which the activities would be performed. Our deliverable for our Monday morning class was a document identifying all the objects and locations.

The document for our objects and locations was a very traditional document, insofar as we designed and documented before we did any coding. Not only did we identify the objects and locations for respective activities, but we also (1) identified the learning objectives for game tour activities (2) wrote the narrative descriptions for each object and location and (3) identified the ways we wanted users to interface with objects and locations. I use this particular predevelopment document to demonstrate not all traditional documentation is out of place, unusable or broken in contemporary development practices. This particular document served the team very well and is a great way to showcase the traditional documentation my peers and I expected.

Table 7

The Original Horse Barn Documentation Version

Scene Name:	**Horse Barn**
Description:	The Horse barn was meant to house the horses, as well as cows and pigs. There is hay in there. There are partitioned spaces for the different animals.
Scene Links to:	Aerial Access
Characters	**Interactive Items/Objects**
Father (moving hay around for the animals)	Water Bucket
Learning Objective:	Care of animals.
Activity:	The father is moving hay around so that the animals have fresh hay to eat. After a long night, the hay bales are a mess and need to be cleaned up.
Learning Objective:	Cows and Horses eat all the time. Horses need to drink after pulling a wagon.
Activity:	When a horse pulls a wagon between farms, it needs time to recover energy after exercise just like everybody else.

A sample of traditional documentation. The original document detailed twelve farm locations and nine farm objects. As four remote graduate students worked with the code and developed the tour game, the thoroughly documented assets were easy to manage. Rather than showcase all locations and objects, I have selected the Horse Barn location and Water Bucket object. The Water Bucket is the only object associated with the Horse Barn in the original document. Table 7 is the exact template used in the original document.

Even while the original document was completed so close to the beginning of the semester on February 10, 2009, the location entries were still ready to support development. The location entries detail the description used in the game, as well as the characters and objects associated with the scene. The learning objectives are clearly articulated; insofar, as the tour game is a learning experience on a historic farm, we decided to clearly identify what players should learn with each activity.

In addition to locations, the original document detailed usable objects in the game. The Water Bucket is the only object identified in the Horse Barn documentation and is therefore a natural object to showcase the traditional strengths of the document.

> *Water Bucket for Horse*
>
> *Description* – (Phase 1) There isn't much to say about a bucket of water. In phase 1, the bucket of water must be taken from the barn to outside of the farmhouse. The neighbor's horse will drink without any prompting.
>
> *Examine* – The water looks cool and refreshing.
>
> *Take/Use* – The player can take the bucket and use it on a horse. The bucket will disappear from inventory and reappear at the pick-up location. The player can use the bucket to give the daughter a drink.
>
> *Other Wherigo Functions* – Water the horse
>
> *Reusable* – The player can return to water horses as much as the player wants.

Unlike the location entry, the object description is more oriented towards development and implementation. The object entry for the Water Bucket identifies the interface options and the instructions for implementation. The water bucket description is flippant but the description is still a reflection about the purpose of the Water Bucket.

The original document was a great place to start on February 10, 2009. However, an engaging development process necessitates change and the document was a site of remarkable changes before the end of the semester.

Three months later and the same traditional documentation. Later in the semester, on April 29, 2009, our team completed the final version of the documentation. The document went from 21 pages to 47 pages in length. The finalized version of the document still included locations and objects; however, there were three more locations and thirteen more objects added between February and April. Table 8 showcases those changes to the horse barn location.

Table 8

The Final Horse Barn Documentation Version

Scene Name:	**Horse Barn**
Description:	The Horse barn was meant to house the horses, as well as cows and pigs. There is hay in there. There are partitioned spaces for the different animals. 41.659673, -111.900651 41.659794, -111.900622 41.659771, -111.900457 41.659653, -111.900434
Scene Links to:	Aerial Access
Characters	**Interactive Items/Objects**
Father (moving hay around for the animals)	Water Bucket **non-takeable items/objects:** Horses: they were used to pull the wagons for people transportation hay: it is used to feed the horses and cows in the barn.
Learning Objective:	Care of animals.
Activity:	The father is moving hay around so that the animals have fresh hay to eat. After a long night, the hay bales are a mess and need to be cleaned up.
Learning Objective:	Cows and Horses eat all the time. Horses need to drink after pulling a wagon.
Activity:	When a horse pulls a wagon between farms, it needs time to recover energy after exercise just like everybody else.

There are several changes to the Horse Barn location entry. There was a subtle change to the formatting; Google docs enable co-authorship but three months revision by four editors left a toll. Second, we added GPS coordinates to the descriptions because the coordinates were hardcoded into the mobile unit tour game. The document identifies a new "Non-takable Items/Objects" class and identifies two additional Horse Barn objects in the new class: Horses and Hay.

We altered the water bucket entry between February and April. The flippant reflection was not changed but the Examine text used in the game was changed significantly.

> *Water Bucket for Horse*
>
> *Description* – (Phase 1) There isn't much to say about a bucket of water. In phase 1, the bucket of water must be taken from the barn to outside of the farmhouse. The neighbor's horse will drink without any prompting.
>
> *Examine* – This bucket holds about THREE gallons of water. Usually the horses will drink about FOUR buckets worth of water in a single day.
>
> *Take/Use* – The player can take the bucket and use it on a horse. The bucket will disappear from inventory and reappear at the pick-up location. The player can use the bucket to give the daughter a drink.
>
> *Other Wherigo Functions* – Water the horse
>
> *Reusable* – The player can return to water horses as much as the player wants.

The radical transformation of the Examine text is evidence that the tour game evolved a great deal between February and April. The Examine text became key to a mathematical puzzle added to the Horse Barn scene; this puzzle did not exist at the time of the original document.

Peer expectations and traditional documentation I asked each of my three peers: "What did you think design documentation was, before you took this class?" At the time, I was still seeking a contemporary development practice in which traditional documentation standards would work. Insofar as I thought the tour game was that practice, I thought my peers could confirm my belief that we did not write standard documentation. I intended the question to highlight my peers' expectations. On one hand, the question highlighted the disparate experience levels of the team. On the other hand, the question demonstrated that students with enormous variance in documentation experience still fixated on a document's purpose and organization.

- Alpha: It is a document that shows the design of the product.
- Bravo: A layout of our learning objectives, our story I guess, an outline of the story…there is the predevelopment design document and the post-development but the final I guess would have the story in detail and every scene in the game and every object just broken down with enough detail for someone else to recreate virtually the same thing without having to fabricate responses, interactions, objects, things like that…with notes about what we changed and why we changed it. Also, probably a break down of hours that were spent…the purpose of the design document is two things is one is to understand the existing project so that if there is things you want to change or debug or workout you easily how everything works when you go in to fix it and the other is someone I think it should be good enough to recreate…as well as a list of all the resources we used to create everything for instance our graphics the Wherigo program, Visionaire, Photoshop, Illustrator, those kinds of things.
- Charlie: A general design document is a document with general design…I would choose to give the answer out of the name.

Bravo was the team member who asked for dialogues and object descriptions; he was the team member I directed to the documentation. Even though Bravo was the team member who used the documentation the least, he was apparently the team member who knew the most about documentation. Whereas, both Alpha and Charlie knew very little but still quickly caught on to the purpose of the documentation.

The locations and objects document showcases the rudimentary concept of documentation. Consequently, the traditional document seemed to work in the American West Heritage Center project. My peers understood documentation at that traditional level. The next sample showcases how documents should work in a contemporary development project.

Sample Full of Collaboration and Meaning

The four teams of graduate students developed the tour game for two different platforms. In order to train the class on the software design tool for each respective platform, Dr. Shelton required a team assignment for each design tool. This involved making a first prototype of the game—a crude, working version of the game. We had a set of files necessary for the prototype to work. Insofar as we were students working independently and asynchronously, we made plans to trade the game files between us as we took turns developing. I suggested we track our time and document our work so that we could control versioning, the trade-offs could be easier to manage, and the progress could be more unified.

I chose the Source Control Log document for my sample because I wanted to show how a document could organize a team and guide a team. The document would be a living document and would be maintained by the team all throughout the course of development. It was in fact that kind of document. We were updating it with our activity.

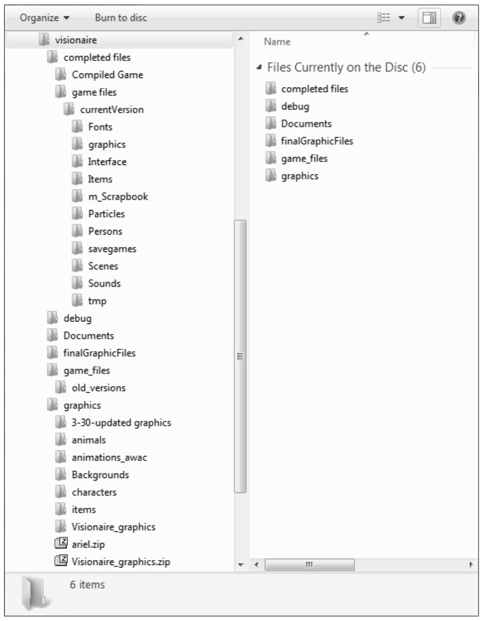
Therefore, the document showcases how the document was a site of collaboration by which the team stabilized the meaning of the project.

Project problems without collaboration and meaning. With four asynchronous, remote developers sharing the same set of files, the potential for disaster was inevitable. Invariably, someone would write over someone else's work, move a file, misspell a word in the code or spend a day developing something that was already completed by someone else. In addition, there was nothing to prevent all four of us making changes to the files at the same time so that there were four different versions of the same set of files.

To illustrate how easily four student developers could have broken the prototype, I want to highlight that the errors do not even need to be big. As long as a file is not where the code says it should be then the program is broken.

Image 9

The File Tree Used in the Code.



```
visionaire
  completed files
    Compiled Game
    game files
      currentVersion
        Fonts
        graphics
        Interface
        Items
        m_Scrapbook
        Particles
        Persons
        savegames
        Scenes
        Sounds
        tmp
  debug
  Documents
  finalGraphicFiles
  game_files
    old_versions
  graphics
    3-30-updated graphics
    animals
    animations_awac
    Backgrounds
    characters
    items
    Visionaire_graphics
    ariel.zip
    Visionaire_graphics.zip
```

```
Files Currently on the Disc (6)
  completed files
  debug
  Documents
  finalGraphicFiles
  game_files
  graphics
```

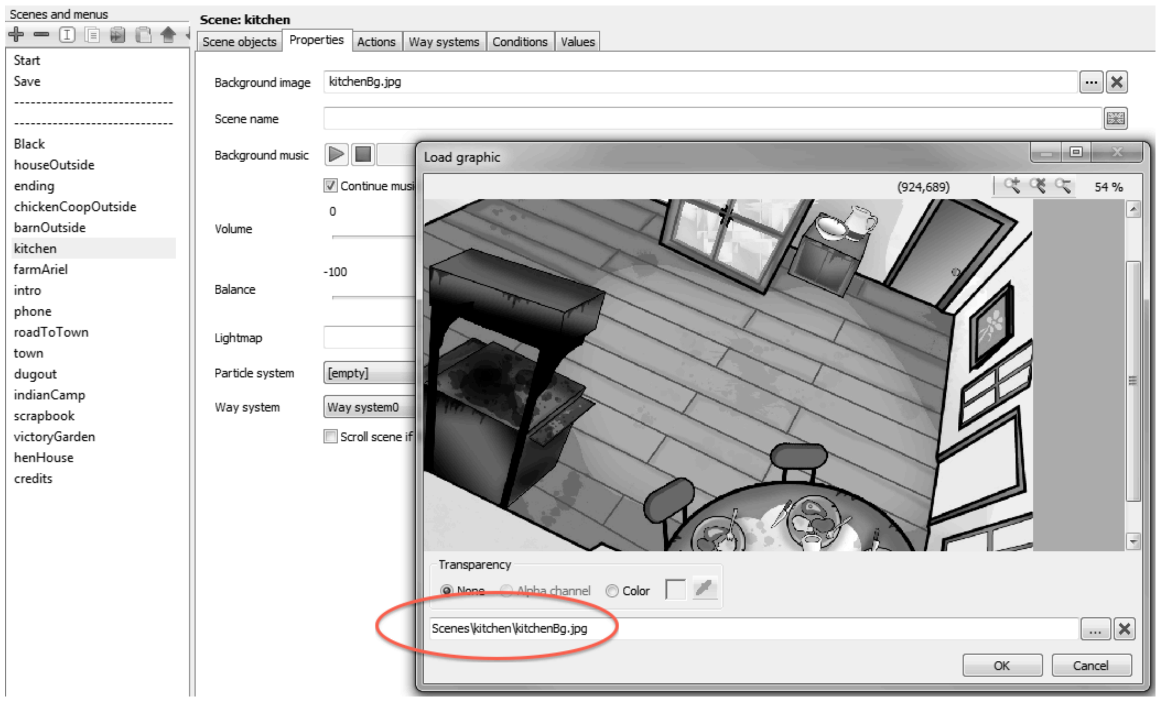
6 items

Image 9 portrays the program's file architecture as a file tree. Each file is nested in a folder. That folder may or may not be nested in another folder. As soon as one file is out of place then the code cannot find the file path. The same is true if the filename is changed or misspelled.

The computer code looks for a specific asset along a specific file path. If the specified file is not at the coded location then the program produces an error report. At that point, the program would leave us scrambling to identify some obscure error when we could be making progress.

Image 10

The KitcheBg.jpg had a Unique in the File Structure.



Scene: kitchen

Background image: kitchenBg.jpg

Scene name:

Background music

[✓] Continue musi

Volume: 0

Balance: -100

Lightmap

Particle system: [empty]

Way system: Way system0

[ ] Scroll scene if

Transparency: [✓ None] Alpha channel Color

Scenes\kitchen\kitchenBg.jpg

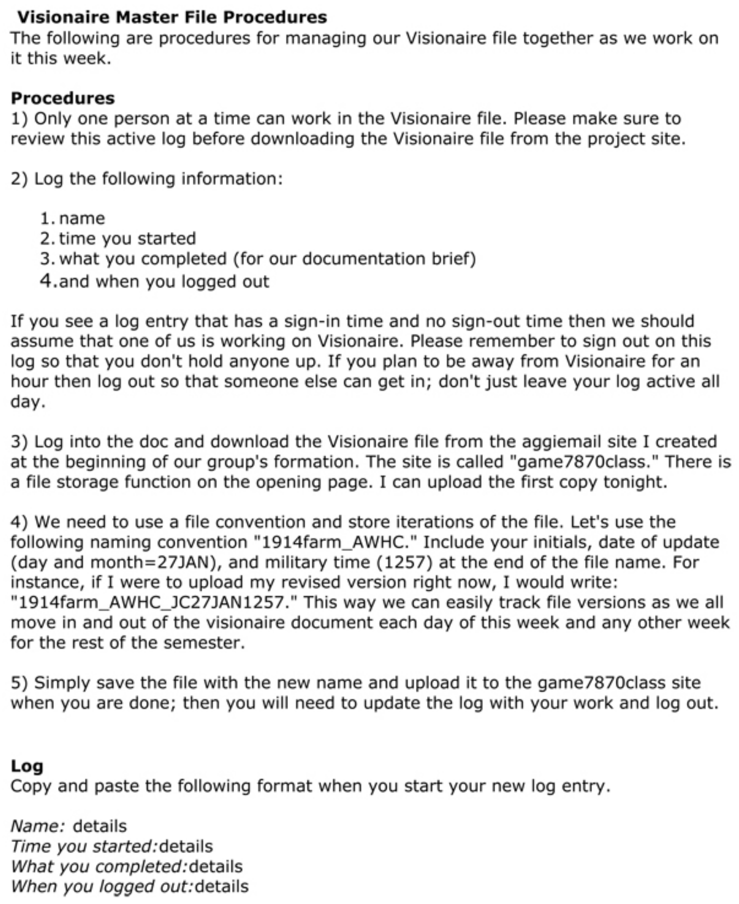
Image 10 shows the KitchenBg.jpg image file. The interface in the background is how we set the instructions for how the program utilized the KitchenBg.jpg image. Fortunately, the design program did not completely necessitate comprehensive coding skills; the design program did all the hard stuff. The interface directs the program where to put the file, when to use the file, under what conditions it should not use the file, etc. Of note, is the file path: Scenes\kitchen\kitchenBg.jp. The Kitchen is only one among many scenes in the game and kitchenBg.jpg is one of many assets important to the kitchen scene. Any code that depended on the image or even depended on a condition that required the image, would break if kitchenBg.jpg was moved or misspelled.

If we planned to email a package of files back and forth to one another then we were going to break our program. We stood to make all kinds of mistakes, to overlap our work, to unnecessarily redo work, developing work that was already obsolete, saving over each other's work and losing track of the most current version. We needed a solution to improve collaboration and stabilize the meaning of our efforts together.

The source control log and collaboration and meaning. I created a procedure by which we would transfer the files (retaining a version history), track changes made to the files, and signal that the files were "checked out" for the use of a team member. The document outlined the standards, along with instructions.

Image 11

The Documented Procedure for Versioning Control.



Visionaire Master File Procedures
The following are procedures for managing our Visionaire file together as we work on it this week.

Procedures
1) Only one person at a time can work in the Visionaire file. Please make sure to review this active log before downloading the Visionaire file from the project site.

2) Log the following information:

1. name
2. time you started
3. what you completed (for our documentation brief)
4. and when you logged out

If you see a log entry that has a sign-in time and no sign-out time then we should assume that one of us is working on Visionaire. Please remember to sign out on this log so that you don't hold anyone up. If you plan to be away from Visionaire for an hour then log out so that someone else can get in; don't just leave your log active all day.

3) Log into the doc and download the Visionaire file from the aggiemail site I created at the beginning of our group's formation. The site is called "game7870class." There is a file storage function on the opening page. I can upload the first copy tonight.

4) We need to use a file convention and store iterations of the file. Let's use the following naming convention "1914farm_AWHC." Include your initials, date of update (day and month=27JAN), and military time (1257) at the end of the file name. For instance, if I were to upload my revised version right now, I would write: "1914farm_AWHC_JC27JAN1257." This way we can easily track file versions as we all move in and out of the visionaire document each day of this week and any other week for the rest of the semester.

5) Simply save the file with the new name and upload it to the game7870class site when you are done; then you will need to update the log with your work and log out.

Log
Copy and paste the following format when you start your new log entry.

Name: details
*Time you started:*details
*What you completed:*details
*When you logged out:*details

Image 11 presents the documented procedure. The actual document was four pages in length. After that first assignment, we adopted pretty specific roles and maintaining the log was no longer necessary. However, the first assignment necessitated the procedure.

The document included a log in which the team would write progress notes. There were four parts of the log: the name of one of the four teammates, the time started, articulation of changes made to the files and the time a person checked the files back in for the next person.

We shared the document online via the Google Docs service. Any changes any one of us made to the document were immediately available to any other member of the team. In this way, we had real-time updates on the status of the file. I could literally log in to Google, review the log, and know that the files were currently checked out or not.

I wrote the first entry in to the log at the time I presented it to my teammates. The following is a subsequent log entry I made. The entry is a good example of the Source Control Log's utility. I was running into problems with my coding competence and wanted to research "offline" so that I did not interfere with progress:

> *Name:* Jason
>
> *Time you started: 10:57 am 02FEB*
>
> *What you completed:* I've run into 184.9 problems. So I'm going to log out of the document and let someone else in while I problem solve. I'll just update the document with my stuff later, once I've figured out what I'm doing. So I'm done for now.
>
> *When you logged out: 1:29pm 02FEB*

The log entry identifies that I had the document for only three hours in the afternoon. Another team member, without the need for explanatory handoffs, checked out the files merely 71 minutes later. In fact, the files were checked out twice before I checked them out again at 11:43pm that night. I was able to read up on the changes that had been made since I checked the files out earlier in the day; consequently, I knew my revision work had not already been done and that the revisions were still necessary. This method of communication orchestrated three revisions by three different people within 12 hours. Four remote graduate student developers could not have had near that efficiency without a communication tool like the Source Control Log.

While the collaborative benefits of the document are evident, the Source Control Log also stabilized the meaning of the project. In other words, the prototype was the byproduct of the team collaboration so that the prototype ended up being pretty close to what we expected. However, when the meaning is not stable, any chain of mistakes determines the final result. Consequently, unstable meaning is not the result of collaboration; an unstable project is whatever it ends up being, without any decision-driven direction. Fortunately, the Source Control Log is also an example of how documentation stabilizes meaning.

The source control document showcases collaboration and meaning. The Source Control Log was a successful document because it was a shared space where remote student developers could track ownership of project files and record progress notes. However, the document did much more for the project because the prototype meant the same for each of the four student developers throughout the prototyping phase.

There is a situation when the meaning of the project could have splintered into four distorted directions. Incredibly, the single situation involved two incidents within 24 hours of the other so that without the Source Control Log the prototype would have been crippled. Responsive communication and adherence to the procedure saved the project.

Charlie checked out the files twice in five hours but uploaded neither changed files nor new assets. Charlie made significant additions to the prototype that Charlie detailed in the Source Control Log. Consequently, the omission of the actual files meant that on the first day of development the team could have had two versions with significant changes.

> *Name:* Charlie
>
> *Time you started:*5:30 pm 30JAN
>
> *What you completed:* created characters (girl, grandpa, mother), and scenes
>
> *When you logged out:* 6:00 30JAN
>
> *Name:* Charlie
>
> *Time you started:*8:50 pm 30JAN
>
> *What you completed:* I worked a little bit on the "sheep place" scene, created the path ways, tried to adjust the girl's size. I tried to follow the tutorial, creating the inventories before creating, and placing the items. For some reason, they don't show up when I run the game. I wanted to finish that scene today, but I think I have to give my brain a break and recharge the neurons.
>
> *When you logged out: 10:30 pm 30JAN*

Charlie logged significant changes that would have gone unnoticed without the files Charlie developed. Charlie’s work was completely invisible to the other three student developers, without the Source Control Log. In addition, the logs suggest another problem, even if Charlie had uploaded the files. Charlie experimented a great deal and after five hours of development work Charlie checked in a broken prototype; Charlie created and replaced files until the prototype no longer worked.

I recorded an entry identifying the omission just over two hours later. Bravo recorded an entry identifying the files were still missing 15 hours later. Both Bravo and I could have proceeded with 15 hours of development progress but we would have created a version control failure. We did not proceed with development because we were able to track progress with the document. One hour after Bravo's entry, Charlie uploaded the files.

Bravo checked out the files for another five hours, once the files were updated. Bravo logged progress notes:

> Name: Bravo
>
> Login: 1Feb - 3:40pm
>
> Read the [design program's] readme file.....
>
> sorry I reorganized files and renamed them also.
>
> Logout: 1Feb - 8:12pm

Bravo both moved all the files and changed all the names of the files. Bravo effected this change throughout the prototype's entire file architecture—as to conform to the design program's standards. Changes to file names can have drastic consequences without proper communication. Without proper versioning and collaboration, the prototype would have been hopelessly broken.

Peer expectations and collaborative documents. Of Alpha, Bravo, and Charlie, I asked: "Can you describe for me the role of a GDD in a development process?" In the previous question, I wanted to know what my peers thought of documentation before our development experience together. This question was meant to prompt my peer-students to articulate the purpose of documentation. My head was full of genre theory so that I was very interested in how my peers would define documentation, without all the theory.

Alpha: It gives us something to follow by it gives everyone allow everyone to be on the same page to know where we are where we need to go and why we need to be there. It gives clarity. Also prevents confusion…Mapped it out and created a blueprint to follow. It is necessary because otherwise you're just shooting in the dark.

Bravo: I mean I think that until we had the design document at least had all those ideas solidified in the document we as a group were disorganized.

Charlie: Well, I guess that before put all the information the narrative our game every information concerning the what we were going to do every step every scene every face the face is very important the faces were detailed in the design document as well as the every addition we made because I remember we never deleted anything we just added the new information and took the other to the appendix. So everything is in there.

The peer responses clearly show the peer-students understood the importance of documentation in transmitting design information. More importantly, whereas Alpha and Bravo were able to merely identify the collaborative aspect of that transmission, Charlie described the documentation as a detailed record of design decisions. Charlie refers to decisions and changes as "additions"; whenever we added something we documented the addition and moved out-of-date information to the appendix. Consequently, we were very consistent and stable in our collaborations.

While the Source Control Log structured collaborative activities, the document also managed the development of meaning. In other words, the prototype was the byproduct of development and that byproduct could have altered significantly depending on the design choices the four student developers made during the project. Whether collaborative decisions that stabilized meaning or a string of communication failures that splintered meaning, the Source Control Log impacted the meaning of the prototype.

A Dynamic Documentation Sample

Every Monday morning at 8:00 AM, the various teams showcased their weekly prototype. According to contemporary development methods, a team needs to demo a working-prototype every development cycle. The reporting keeps teams accountable, the regular cycle keeps teams productive, the shortness of the cycle ensures a very organic development process and the working-prototype means progress. The teams would all collaborate together and try to agree on standards and the teams would receive direction for the next week of development. The teams would break up and make more specific assignments. My team would work out the big decisions together and then we would disperse until the following Monday.

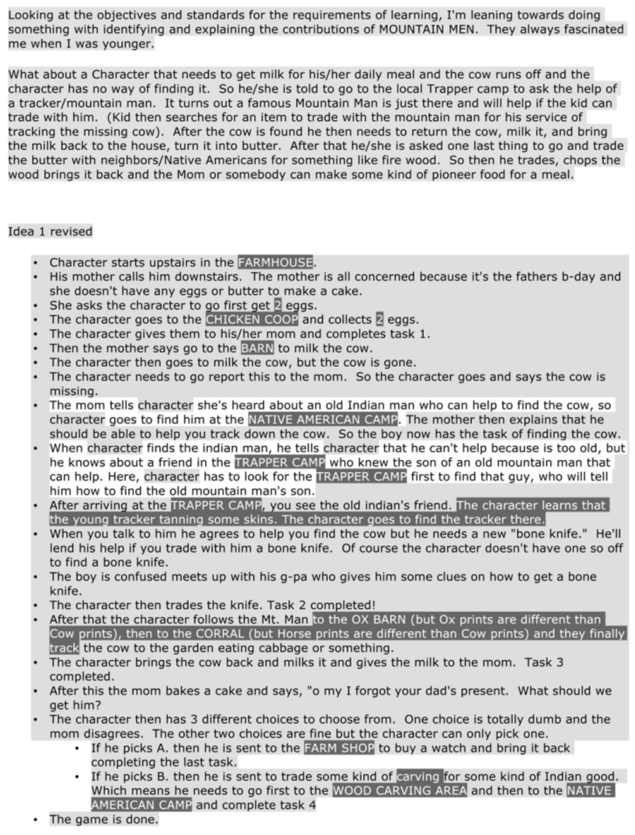
We were four graduate students whose lives only crossed during that Monday morning at 8:00 AM. Unless we planned to meet at some other random time, we worked remotely with whatever computer resources we could find. Whether from university computer labs or from home computers, we developed alone for 7 days. However, after seven days, we would showcase our working-prototype—evidence of how well we worked together. In the absence of a common workspace and daily interaction, we had an online document sharing solution. That hub of communication was a library of documents hosted by Google. The Google Docs were where we recorded our progress, posted our questions, looked for updates, compared effort, and documented our unified direction. By Monday morning, our team’s working-prototype had been through a development process about as organic as some professional development teams.

I chose the Dynamic Documentation Sample from the most collaborative of our documentation. When we were developing the tour game's narrative, we had accessory documents and a larger, central document into which we all contributed our efforts. Even more important than the fact that we actively documented is the fact that we all revised together. Consequently, any page in the documentation was the result of multiple edits from multiple authors. We were a dynamic community that literally shaped the documentation, even while it helped keep us unified and on course. I chose this sample because it shows a dynamic, relevant document sustains a morphing team, even while the team morphs and reforms the development goals and standards.

Dynamic Hub of Communication. If not for common documentation, I was one of four graduate students who would have splintered far from any common path. We needed a hub of communication so that we could keep posted on the tour game's evolution. Before I did anything, I would log in to the Google document and check for the most current decisions. We could not make those course-altering decisions together but we could watch the development of those decisions in the document changes. I would describe some aspect of the design and within hours text highlights would mark changes made by another developer.

Image 12

Sample Section Revised by Team Members.



Looking at the objectives and standards for the requirements of learning, I'm leaning towards doing something with identifying and explaining the contributions of MOUNTAIN MEN. They always fascinated me when I was younger.

What about a Character that needs to get milk for his/her daily meal and the cow runs off and the character has no way of finding it. So he/she is told to go to the local Trapper camp to ask the help of a tracker/mountain man. It turns out a famous Mountain Man is just there and will help if the kid can trade with him. (Kid then searches for an item to trade with the mountain man for his service of tracking the missing cow). After the cow is found he then needs to return the cow, milk it, and bring the milk back to the house, turn it into butter. After that he/she is asked one last thing to go and trade the butter with neighbors/Native Americans for something like fire wood. So then he trades, chops the wood brings it back and the Mom or somebody can make some kind of pioneer food for a meal.

Idea 1 revised

- Character starts upstairs in the FARMHOUSE.
- His mother calls him downstairs. The mother is all concerned because it's the fathers b-day and she doesn't have any eggs or butter to make a cake.
- She asks the character to go first get 2 eggs.
- The character goes to the CHICKEN COOP and collects 2 eggs.
- The character gives them to his/her mom and completes task 1.
- Then the mother says go to the BARN to milk the cow.
- The character then goes to milk the cow, but the cow is gone.
- The character needs to go report this to the mom. So the character goes and says the cow is missing.
- The mom tells character she's heard about an old Indian man who can help to find the cow, so character goes to find him at the NATIVE AMERICAN CAMP. The mother then explains that he should be able to help you track down the cow. So the boy now has the task of finding the cow.
- When character finds the indian man, he tells character that he can't help because is too old, but he knows about a friend in the TRAPPER CAMP who knew the son of an old mountain man that can help. Here, character has to look for the TRAPPER CAMP first to find that guy, who will tell him how to find the old mountain man's son.
- After arriving at the TRAPPER CAMP, you see the old indian's friend. The character learns that the young tracker tanning some skins. The character goes to find the tracker there.
- When you talk to him he agrees to help you find the cow but he needs a new "bone knife." He'll lend his help if you trade with him a bone knife. Of course the character doesn't have one so off to find a bone knife.
- The boy is confused meets up with his g-pa who gives him some clues on how to get a bone knife.
- The character then trades the knife. Task 2 completed!
- After that the character follows the Mt. Man to the OX BARN (but Ox prints are different than Cow prints), then to the CORRAL (but Horse prints are different than Cow prints) and they finally track the cow to the garden eating cabbage or something.
- The character brings the cow back and milks it and gives the milk to the mom. Task 3 completed.
- After this the mom bakes a cake and says, "o my I forgot your dad's present. What should we get him?
- The character then has 3 different choices to choose from. One choice is totally dumb and the mom disagrees. The other two choices are fine but the character can only pick one.
    - If he picks A. then he is sent to the FARM SHOP to buy a watch and bring it back completing the last task.
    - If he picks B. then he is sent to trade some kind of carving for some kind of Indian good. Which means he needs to go first to the WOOD CARVING AREA and then to the NATIVE AMERICAN CAMP and complete task 4
- The game is done.

Image 12 has three colors, representing the contributions of each graduate student on my team. The colors distinguish contributions so that words are highlighted and phrases are highlighted. In some cases, there are breaks in a highlight, indicating another contribution made on top of the first contribution.

Image 12 showcases the first draft of the story. The American West Heritage Center Tour game was split into three storylines; I was on the team responsible for the 1917 interpretation of the historic farm. Beth is the main character of our tour game;

however, we really wanted to have a trickster in the story so that the tour becomes more of a game. We selected the Grandfather as the trickster that would pop up throughout the tour; the grandfather would interfere with player progress.

Dynamic, without doubling or diverging. The role of the grandfather altered significantly over the course of development. Consequently, developing a tour game around the morphing role of the grandfather was a challenge. There was a version on the story in which the grandfather was in every scene. There was a version of the story in which the grandfather was the antagonist through which narrative progress was possible. There was a version where Beth was a co-conspirator who had to report back to the grandfather. There was a version where the grandfather stayed at the farmhouse and provided misinformation. There was a version of the story where the grandfather hid from Beth's mother in the horse barn. There was even a version of the story where we gave up on the grandfather and had him sleep in the kitchen for the entire farm tour. The final version of the story is where the grandfather simply hangs out in the kitchen and slips Beth clues; the clues help Beth make her own birthday cake surprise.

In fact, one version of the story involved an entire section of the American West Heritage Center that was not part of the 1917 historic farm interpretation. At that time, the student development teams had not yet decided to restrict each of the three storylines to their specific area of the Heritage Center properties. Consequently, the grandfather's role was useful in bridging to other parts of the farm.

> The character seeks out her Grandfather at the WINDMILL. Grandpa hangs out there because keeping the thing functional is an all-day job. Grandpa is also a clever old fool that Beth trusts. She helps her Grandpa

> make some repairs at the WINDMILL and they take some grain or flour (isn't that what they produce at windmill's?) to the FARMHOUSE (for the cake) and then hatch a devious plan. They pick some wild flowers at the SMALL BRIDGE, go to the OPERA HOUSE, and sell the wild flowers to house for the evening's performances. Now that they have some capital, they head off to the NATIVE AMERICAN CAMP to get their hands on a bone and to the SMITHY (a smith is not on the list but I'm positive there is one) to forge a bone knife. Beth leaves her Grandpa at the FARM EQUIPMENT SHED before Grandpa can return to the WINDMILL.

With the exception of the Farmhouse, the locations (in all caps) were not on the 1917 historic farm interpretation, even if they were still on the Heritage Center property. For instance, the Opera House was on the far side of the Heritage Center and was more appropriately part of the Center's Utah Pioneer historic interpretation. Regardless, the example showcases the extent to which we revised the role of the grandfather in the tour game.

Dynamic and real-time—not simply multiple authors. The grandfather posed specific problems to four remote graduate students who were making prototypes on a weekly basis. Even while we were each building our respective parts of the tour game, we were also rewriting the grandfather's role in the design document. Insofar as the grandfather, at one time or another, touched every scene of the tour game, it is a miracle that we were working on the same conception of the grandfather at the same time; yet, that is the case. If we were simply four authors writing in the same document we would have had a mess; however, the writing and development happened simultaneously so the document sustained the meaning of the project.

Even while Alpha would be coding the grandfather, I would make a change to the grandfather in the design document. Meanwhile, Charlie would have discovered some anomaly in the historical interpretation and changed the grandpa yet again. Yet, Alpha attended to the design document so that grandfather was coded accordingly. In addition, Bravo was coding the same tour game on another platform. Perhaps there was an element of the grandfather's role that did not work in Bravo's platform; Bravo might mention the constraint in an email. In such a case, Alpha and Charlie would have feedback. The consequent decision would be copied from the email string into the design document. In the end, Bravo's presentation of the grandfather was identical to that coded by Alpha. Come every Monday morning, we would stand in front of the class and showcase unified prototypes for each platform.

The tour game design document is 47 single-spaced pages. The grandfather's role was a work in progress that touched items, scenes, characters, and the code itself. The fact that four remote, student developers used a hub of communication to make dynamic, real-time, coordinated changes is an example of how a dynamic, relevant document sustains a morphing team, even while the team morphs and reforms the development goals and standards.

Peer expectations of dynamic documentation. I was excited about the team's dynamic hub of communication. I was full of genre theory but I wanted to know if my peers felt the same way. I asked Alpha, Bravo, and Charlie: "Now that we are 9 weeks into development, what do you think [documentation] is now?" The first question I had asked prompted my peers to reflect back on their original reaction to the design documentation I wanted to maintain. The second question I had asked was simply to see how my peers defined documentation. I saw this third and last question as their opportunity to articulate the differences they saw. The responses to the question are not surprising for Alpha and Charlie; I knew they were impressed simply because I had been working so closely with them. However, while Bravo's actual response is not long, Bravo's story highlights the central problem of developing and documenting at the same time.

Alpha: I was like wow. So that is a design document that is nice.

Bravo: I guess I got a lot of respect of it before you start developing a product whereas before in other classes I did it because I was told to.

Charlie: Well, I looked at it a couple of days ago and I thought oh my gosh there are many things in here because I know that the first section is the most recent information and I side scrolled and saw all the old stuff I said these guys have done a lot…We changed a lot of things…All of us had access to it we all had options to edit…[Charlie can see version histories in the Google doc design document] I think it is very useful.

Alpha was simply impressed by all the work. Charlie saw the 47 pages of collaborative design and writing; Charlie's insight is exciting because I'm not the only one who noticed how the hub of communication sustained the meaning of the project. However, Bravo had the most rewarding insight; Bravo had confided in me that documentation was a chore and Bravo preferred to simply develop in isolation. In other words, despite Bravo's familiarity with documentation, Bravo resisted documenting his work. Yet, Bravo was the one who did not know what was going on when it was time to fit consistent, current information about the Grandfather into the code.

DISCUSSION ABOUT AMERICAN WEST GAME

From the beginning of American West Heritage Center tour game development, I knew I had an opportunity to conduct documentation practices as close to the textbook as possible. I was armed with genre theory and had clear expectations for what I knew I should see. The Model of Expectations translates the meta-language of genre theory into the industry parlance that describes my expectations. I expected dynamic, relevant documentation that would sustain a morphing team, even while the team morphs and reforms the development goals and standards. I sought to show how the tour game development documentation met my expectations—how it matches the prediction of genre theory

American West and Development Expectations

Bravo's story highlights a developer's need to stay in constant connection with a strong, reliable hub of communication. However, before the American West Heritage Center project, I was unsure whether the hub of communications I expected was even possible in contemporary development practices. By the end of Spring 2009, I felt the project did meet my expectations.

Model of expectations. The following discussion breaks the model of expectations into the five component parts. Of the three documentation samples, the third sample is an example of dynamic documentation and is the most relevant. Consequently, the following discussion focuses on that specific example.

Dynamism	Dynamism for Practitioners
• Genres are developed from actors' responses to recurrent situations. • Genres serve to stabilize experience and give it coherence and meaning. • Genres change over time in response to their users' sociocognitive needs.	• The document forms as the team uses it and responds to it. • The document is a common resource for teams and is a foundation for the innovative solutions teams require. • The document changes with team decisions so that it is a decision making tool.

I was on a team of only four graduate students. We were remote and relied on the documentation to organize ourselves and guide ourselves. The sample showcases how the role of the Grandfather was the result of collaboration and the documentation was the sole medium of communication. At the same time, while we were constantly adhering to the document, we were also writing and rewriting the document. Therefore, the actors' responses were core to the recurrent situation every Monday morning.

Alpha, Charlie, and I actively worked to keep the documentation current and relevant. More often than not, the document was the only source of the project's current status and the only expression of what the project's deliverable would look like. Even Bravo had to seek out the document for meaning when Bravo required essential details about the game's development.

Whether I was responding to emails or attending an independently scheduled team meeting, I relied on the document to control the scope of discussion, the assignment of tasks and the resolution of differences. Often, disputes would simply resolve because the most recent revision to the document offered the solution. Consequently, the document was core to the decision-making, meaning-making, and problem solving we would work on together.

Situatedness	Situatedness for Practitioners
• Genre knowledge is derived from and embedded in our participation in the communicative activities. • Genre knowledge is a form of "situated cognition." • Genre knowledge continues to develop as we participate in the activities of the ambient culture.	• If it isn't documented then it didn't happen. The act of documentation is the formation of common knowledge. • Project productivity and long-term goals originate from documented knowledge. • Documents become more comprehensive and typified as the community collaborates and tests the document's relevance.

Without the core hub of communication, our team would have had no anchor. We were remote students who developed a software program for a single class that met for three hours on Monday morning. We had that common anchor from which we drew our knowledge about the project and it was to that common source that we would add value. Simply by needing information from the document, we participated in the document's relevance to the project.

Anything I knew about the Grandfather or the farm was either from the document or added to the document. Therefore, while the document was not the fount of all knowledge, the document was the only place for community-knowledge; the team's knowledge of the project was literally situated within the documentation.

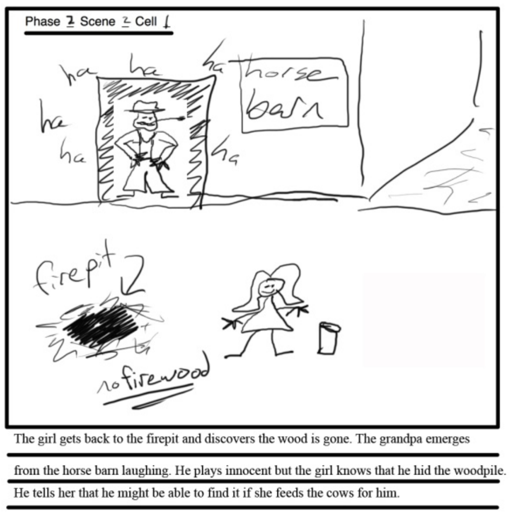
Form and Content	Form and Content for Practitioners
• Genre knowledge embraces both form and content. • Genre knowledge is a sense of what content is appropriate to a particular purpose, situation, and time.	• Form and Content conform to documented knowledge. • Document-based knowledge prompts decisions about relevant rhetorical content.

One interesting detail about the narrative documentation sample was the formatting. In the beginning, the narrative documentation was in story paragraphs with dialog quotations. However, the format of the documentation was changed in the process of development. The paragraph form was not conducive to a dialog-driven game and the paragraph transitions were too prosaic for computer code. Consequently, we changed the formatting of the narrative to match our development needs. There was no rule about appropriate formatting but we decided to map out the dialogs as close as we could to the tree structure of computer code. We identified what served us best for our particular purpose, situation, and time.

We even tried storyboarding the Grandfather's narrative at one point. Image 13 is just one of the many storyboard panels we used. The sketches are horrible by artistic standards but they conveyed the narrative for the particular purpose, situation, and time. We ultimately discontinued the use of storyboards but the storyboards illustrate that Form and Content are not static values outlined by industry guidebooks.

Image 13

A Storyboard Panel Depicting the Grandfather.



The girl gets back to the firepit and discovers the wood is gone. The grandpa emerges from the horse barn laughing. He plays innocent but the girl knows that he hid the woodpile. He tells her that he might be able to find it if she feeds the cows for him.

Duality of Structure	Duality of Structure for Practitioners
• Genre rules inform activities that constitute social structures • Genre rules inform activities that simultaneously reproduce these structures.	• Project planning, roles, and responsibilities match procedures established in the documentation. • The documented rules guide procedural and organizational decision-making.

Our team gradually specialized in various aspects of the development. Consequently, while the document ultimately determined what kinds of roles our little development project required, our roles inevitably changed the document. After all, Alpha was exclusively dedicated to development for the mobile platform and Alpha's software-specific needs would necessarily inform the information Alpha added to the document; consequently, the document contained information prepared for Alpha's roles and procedures.

Against what might seem logic or wisdom, we were always following and rewriting our own guidelines. Our situatedness was simply so comprehensive that we needed the document until we decided to change the document—at which point we needed the document again. For instance, even though Bravo was disconnected from the documentation in the beginning, he still needed it to remain cohesive with the rest of the team; yet, his demands might still alter the project's meaning, even if the cohesiveness and stability are not broken.

Community Ownership • Genre conventions signal a discourse community's norms, epistemology, ideology, and social ontology.	Community Ownership for Practitioners • Philosophy of development methodology expressed through documentation and practice.

The greatest strength of our project was our community ownership. Our team of four remote student developers was completely invested in a hub of communication that worked well for us. While encouraging community ownership was something I wanted to instill from the beginning of the project, I did not expect I would not need to do much for the group to own the documentation. With the exception of a few face-to-face meetings, all our communication was written and our project knowledge was maintained in the documentation.

American West and Replicating the Results

North American Genre Theory can describe the success of the American West Heritage Center development project's documentation. The documents had a dynamic, integrated role. The documents were the situated repository of the team's project knowledge. The documents had flexible form and content that evolved to meet the team's needs. The documents demonstrated a duality of structure; they straddled the fine line between imposing structure and reproducing structure. Finally, the community ownership of the team made the documentation a hub of communication.

Yet, the project documentation was not a comprehensive success. Not so apparent at the time was the project's reliance on written communication alone. In addition, I wanted to match my model of expectations in yet another project.

What is wrong with the american west project's results? I have emphasized the remote nature of the team's development situation. We had a classroom in which we met both together and with the other two student development teams. However, any development work was done remotely. Research, documentation, collaboration, coding, and graphic design all happened while four graduate students worked separately at home, at the library, at Dr. Shelton's IMRC lab or at campus computer labs. Because of the remote nature of development, any communication was necessarily written. Consequently, even while the hub of communication produced an impressive amount of documentation and matched my Model of Expectations, the hub of communication was also a function of remote teamwork.

When I look back, I realize the project easily matches the Model of Expectations because of the project's reliance on written communication. There was no verbal communication through the course of a development cycle. The only verbal communication occurred on Monday morning. At that time, I would collaborate with my co-developers and we would align our project with any new directions from Dr. Shelton. We would invariably change the role of the Grandfather after open discussion with the other student teams. We would split up responsibilities and identify our deliverables for the following Monday. We would troubleshoot any coding barriers and plan any necessary code fixes.

None of that Monday morning work was written communication; interestingly, none of those "off-stage" decisions were actually documented.

Can I replicate remote documentation with a centralized team? I was given the challenge to replicate the documentation in another project. I had seen duality of structure work in practice and was excited to go beyond the theory yet again. Dr. Hailey already had a project lined up. There would not be any software or games; however, the project involved web development and the generation of Flash videos for online learning modules.

CHAPTER V

POSTMORTEMS: ENGINEERING MODULE SUPPLEMENT PROJECT

CONTEXT FOR THE ENGINEERING PROJECT POSTMORTEM

The software design documentation is for a developer to write up sufficient elaborations so that a team of developers can understand design details. The problem is that contemporary software developers do not frequently abide by the genre rules. For example, in a 2006 email, a Salt Lake City, Utah software manager told me: “[Design documents] are not exhaustive or exacting blueprints that we slavishly execute through to completion...I say this because I suspect that some of the literature regarding the creation of design documentation...errs in describing them.” This software manager did not rely on industry conventions or guidebook recommendations; he adapted and employed his own rules and rhetorical forms.

This chapter is a postmortem for a Utah State University Department of Engineering course module development project. The project’s student developers would abide by genre rules and produce the documentation I expect to find among professionals. We would replicate the success of the American West Heritage Center documentation. However, just like industry professionals, the student developers did not abide by the genre rules. At the same time, I was able to observe other rhetorical forms that constituted “documentation activities” on the project and matched genre rules.

The Project's Origin

While I became increasingly interested in the specific oral communication practices of the student developers, I was also aware of other rhetorical forms. This chapter toes a fine line between identifying the documents that should have been delivered and how the research team identified oral communication in the first place.

Leadership of the project. In the summer of 2009, I was invited to coordinate supplementary course modules for the Department of Engineering. The project's funding came from the State of Utah Engineering and Computer Science Initiative. The principal investigator was Senior Associate Dean Dr. Wynn Walker and the project's development manager was Dr. David Hailey. Dr. Walker and Dr. Hailey selected professors to present supplementary content and manage live classes. In addition, they already had funding for interns who would film and edit the modules. However, until they brought me onboard they did not have someone to coordinate the various activities of the project. I coordinated both the film sessions and faculty schedules; in addition I intended to simulate a workplace situation and use Genre Theory to describe the documentation.

Exigency of the project. The Engineering Module Project set out to create four modular courses for the satellite campuses of Utah State University. Utah State University is located in the northernmost part of Utah. In fact, the university is only 30-miles from Idaho. Utah State University has 30 remote campuses dotting Utah. The students who attend those remote campuses would never set foot in an engineering class on the Utah State University campus in Logan, Utah. This means that while any student can major in Engineering, not every student can take a course from a professor of Engineering. For instance, students who attend the remote campus in Kanab, Utah will never likely drive 393 miles 2 or 3 times a week to attend classes on the Logan, Utah campus proper.

The Engineering department obtained funding to resolve this problem with the distribution of course modules that can supplement classes taught at remote campuses. There is a film studio in the engineering building and they planned to film the instruction there. The team filmed almost 150 hours of Engineering courses and used Camtasia software to capture instructional actions on the professor's computer screen; all told, they converted almost 300 hours of footage. The team threaded the instructor footage with the screen capture footage to create the supplementary modules. As a result of this project, remote students can watch master teachers present lectures.

Scope of Engineering Modules Project

I expected my project coordination with the Utah State University engineering department would help me generate documentation samples and therefore understand the socially dynamic actions of development teams, while developers break from the genre rules. I planned to play participant observer on a development team all summer. I had also wanted to imbed myself in a long-term development process. The greatest advantage of the project was any documentation we generated would not be protected by industry copyright and would not involve intellectual property. There is simply too much intellectual copyright to access proprietary documentation, never mind the software itself, for industry practitioners to trust a university researcher. Therefore, while I was pleased I did not have any problem accessing and sampling student project documentation for my research, the students did not actually write their documentation. I did not expect a failure to either generate documentation or draw useful research conclusions from that failure. However, I still observed how the student developers succeeded without proper documentation practices. I learned that contemporary development methodologies include rhetorical forms for working face to face—including speech acts.

Since I began researching software documentation in 2006, I have observed, interviewed and reviewed samples of professional developers but had not observed the social recursions of the software design documentation genre. The problem I had was finite document samples merely gave a peek at frozen moments isolated from the development cycle. Those peeks might showcase form and content but they did not showcase dynamism, situatedness, duality of structure, or community ownership. For instance, one professional developer gave me the table of contents alone when I asked for a general design document sample. Unless I had access to the social recursions and meaning-making then I could not understand the genre of software documentation.

I wanted to experience meaning-making decisions; I wanted to generate the documents; I wanted to have insider access to the developer community; I wanted to develop my own practitioner's perspective; I wanted to have documentation responsibilities and be accountable to a team of developers. The Engineering Modules Project with the Utah State University Department of Engineering gave me those opportunities.

Opportunities to imbed myself in the cycles of documentation helped me understand the impact of human agency in genre cycles. In their video game design manual (2007), Ernest Adams and Andrew Rollings claim "the key part of game design is transmitting the design to other members of the team" (p. 62). Adams and Rollings identify documentation as that key part. In other words, rather than isolated developers unaware of elements under development in another cubicle, the transmission of design permits communication, collaboration, and a general awareness. In fact, Cooper (1999) argues that software development documents are blueprints, rather than suggestions (p. 237); Cooper even goes so far as to argue that the developers should not have the authority to change the blueprint. Despite Cooper's suggestion, the best practice suggests rather than waste time verbally repeating elaborations at every inquiry throughout a week (Clements et al., 2002), teams should be able to access any of several central documents without necessitating the verbal repeat of elaborations, reflections, and solutions.

As the coordinator for the Engineering Modules project, I set out to research genre practices in software development. I expected to follow this plan:

1. Familiarize myself with the web broadcast course and the team of developers.
2. Review their documentation practices.
3. Train the developers in appropriate documentation practices.
4. Observe the paradigm shifts as some of the developers see the significance of ongoing, living documentation.
5. Interview my developers to understand their old and new perspectives
6. Manage the cycles of socially sustained written patterns that coincide with the rapid prototyping methodology of the software developers. In fact, based on previous observational experience, I know that documentation is a standard part of that cycle.
7. Interview my developers to discover how they conceive their agency as genre agents.

8. At the end of the summer term I can conduct one more round of interviews.
9. Fall semester will begin another cycle and I will have my field research data.

Research Context for Engineering Modules Project

The Engineering Modules project involved a mixed team of faculty and students, would feature modules grounded on strong research and would yield documentation.

Engineering modules development team. The development team consisted of myself as the coordinator, Dr. Hailey as the development manager, an Engineering Master's student intern, and three undergraduate English student interns. The student interns were split into two teams; each team had a camera operator and a video editor. The teams were not static so that any one intern could work with any other intern. However, the roles of camera operator and video editor were locked.

While the student interns worked with the filming and video editing, Dr. Hailey built the module interface using Adobe Flash. The original intent was the student interns would build the interface; however, the demands of filming and video editing were sufficient to warrant a change in plans.

As the coordinator, my job was to maintain contact with the Engineering faculty. While two of the course modules were filmed during a live class, the three other course modules were filmed in the studio. Therefore, while I did not need to schedule the live classes, I did need to schedule with the professors involved in the three other courses. In addition, there were three occasions where an error with the video processing necessitated an additional film session to record the lecture a second time; I had to schedule these extra sessions.

Modules as media or genre. Professors David E. Hailey, Jr. and Christine E. Hailey (1998; 2002; 2002) have researched teaching pedagogy in engineering classrooms. Particularly, they took interest in the efficiency of supplementary learning modules. They measured for the speed with which students use modules. They have researched both the considerations and genre choices, as opposed to media choices, in considering the inclusion of digital pedagogies. Their concern with digital media has to do with the genre and media choices teachers must make when incorporating digital media in the classroom. These modules are media meant to supplement the lecture genre; the face-to-face lecture is a medium that is not replaced by flash modules. In addition, the interface of these flash modules makes a quantitative difference in the comprehension of students.

Documentation. Genre theory researchers approach genre differently than traditional genre because they seek to understand who made the rules—the form, content, and purpose. Insofar as genres are not spontaneously formed from midair, researchers tend to look at genres as socially sustained writing. Thomas Kent (1986) writes of the significant difference: “In one sense, a genre is a system of codifiable conventions, and in another sense, it is a continually changing cultural artifact” (p. 15). By codifiable, Kent breaks down genre and suggests that as people interpret the very words and sentences they form the genre from their interpretation. In fact, he argues: “that our ability to recognize these formal, rule-bound conventions influences our response to different kinds of texts” (p. 39). In other words, even as genre is a continually changing cultural artifact, the most basic level of interpretation works to influence how the text is read.

Important qualifications. With all the emphasis on rhetorical forms, oral communication and the failure to produce documentation, I do not want to forget the importance of traditional software documentation and why it is still adds value to professional development projects.

The purpose of the internal software documentation of traditional development is to record the design, decisions, direction, and implementation plan for large teams. This purpose is still useful for contemporary developers. Oral communication cannot replace documentation because documentation is supposed to be a safeguard against untraceable speech in the first place. From this safeguard, I can identify three reasons why contemporary development teams cannot replace documentation with oral communication.

- Poor Memory. Three developers might have a great solution for a major problem found in the code; however, when they each go to their three respective teams one of the developers might not relay the solution in the same way as the other two do. However, if the three developers document their solution then everyone can rely on the documented solution.
- Repetitious Elaborations. One document can save the time of everyone. One programmer does not need to repeat a solution for every team, every day, every time someone wants to clarify the implementation of the solution. After all, a dozen different teams do not have to hound one team for various details if those details are recorded in clear documentation.
- The guidebooks indicate that documentation sets the project vision by which developers can direct their actions throughout the duration of the

> project. Guidebooks also indicate that documents are for planning; in this way, teams can direct their actions and coordinate their work, even if they are developing different aspects of the same project.

These qualifications are important to signal where the Engineering Modules development team failed to follow good practice and where good practice became too traditional to work for the team. Rhetorical forms seem to support contemporary practice but not at the expense of the clear advantages of documentation.

TRADITIONAL SAMPLE FROM THE ENGINEERING MODULES PROJECT

There are three samples of traditional documentation from among the few available. The Engineering Modules project simply did not yield a significant quantity of documentation. The objective of this section is to showcase the traditional document samples I did gather and juxtapose them against the samples of contemporary rhetorical forms I eventually identified. In addition, the purpose of traditional documents did not meet the situation or community of the Engineering Modules project.

Sample: Standard Operating Procedure

Images 14 and 15 are the two pages of a traditional Standard Operating Procedure document. The team used Adobe Premiere to overlay video and this Standard Operation Procedure document outlined the steps. The document uses screenshots (Image 15) to illustrate some key steps in an important development procedure. The team had two video feeds: a video camera recorded the lecture and the notepad laptop was loaded with Camtasia (a screen recording software) to record the notes the Engineering professor wrote for the overhead projection. There were also two audio feeds: one audio for the camera and another audio for the Camtasia recording. After the student interns uploaded all that digital data to the editing computer, the student interns would splice the elements together. The objective was to switch back and forth between the notes view and the lecture view; however, the primary view was a small box for the lecture video floating over the primary view of the notes. Consequently, the team needed a Standard Operation

Procedure (SOP) document to standardize the overlaying process. At the time the document was written, much of the video editing process was still undiscovered. While overlaying video was a pretty elementary action by the end of the summer, getting one video to layer on top of another video was a major achievement in the beginning.

Image 14

Page One of Overlaying Document.

Overlaying Video

This document explains the procedure for overlaying video footage in Premiere. Insofar as the project involves two video sources—camtasia footage and video camera footage—there needs to be a way to utilize both sources during the editing. For instance, the project can alternate between the camtasia footage only and the video camera footage only.

However, much of the editing includes the video camera footage on top of the camtasia footage in a small box.

Image 1 is a screenshot of the Adobe Premiere interface taken from an Adobe.com tutorial video.

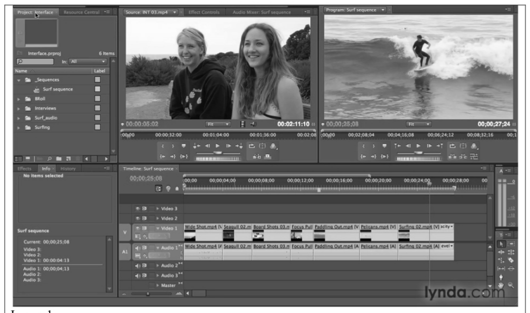


Image 1

Step One:
First, distinguishing between the two video boxes is important. The left box of image 1 is the source video panel; in other words, the original footage imported into premiere is preserved in the box to the left. In image 1, the source video panel features two women. The second box is the project video panel; this is the box to the right and displays the edited footage for the current project. The footage displayed corresponds to the play head bar in the project timeline. In the image above, the project video panel features a surfer.

Image 15

Page Two of Overlaying Document.

Step Two:
In image 2, the project involves a simple usage of the project timeline.

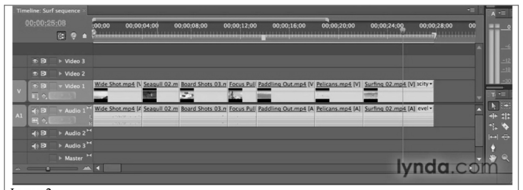


Image 2

However, in an active project, there will likely be several video and audio lines in the timeline. These lines of content can be spliced together or overlaid as in image 3. The project box can display several layers of footage at the same time. This is done by default; however, the default size of projects hides others. The various layers can be edited by making one or another layer invisible. In the case of the project panel for image 3, the top layer has been reduced so that the second layer is also visible. Image 3 showcases what overlaying footage means.

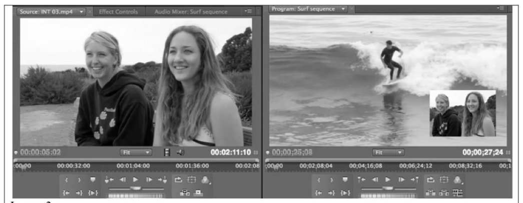


Image 3

Step Three:
A mouse is all that is necessary to overlay footage. Just as in the case of any file window, the size of the video image can be adjusted and moved around the project video panel. In image 3, footage of the two women has been reduced with the mouse and drug across the project video panel with the mouse.

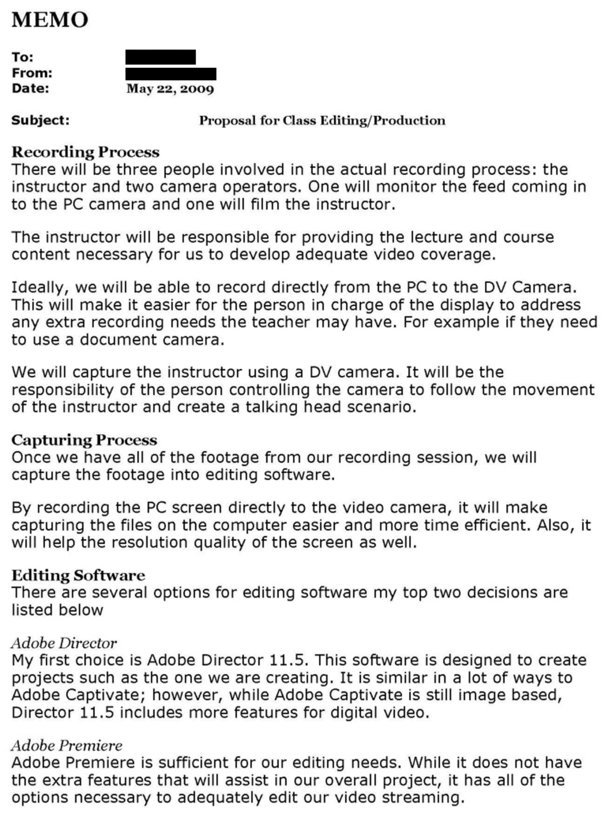
This document is incredibly useful for a new person coming onto the team and for a veteran to remember any formulaic steps applicable for a specific project. However, the document is only a resource—rather than a meaning-making tool. For instance, no design decisions rely on the overview (Image 14) or three-step procedure (Image 15); yet, the team made an undocumented decision to present the modules by overlaying video feeds.

Sample: Procedure Memo

Image 16 is a procedural memo document details the actual recording procedure and recommends the software that best fits the process. This is another kind of preplanning document. While a procedure document usually does not reside in a recommendation memo, the recording procedure was organized, tested, and ready for implementation.

Image 16

Procedural Memo Details Footage Management.



MEMO

To: [redacted]
From: [redacted]
Date: May 22, 2009

Subject: Proposal for Class Editing/Production

Recording Process
There will be three people involved in the actual recording process: the instructor and two camera operators. One will monitor the feed coming in to the PC camera and one will film the instructor.

The instructor will be responsible for providing the lecture and course content necessary for us to develop adequate video coverage.

Ideally, we will be able to record directly from the PC to the DV Camera. This will make it easier for the person in charge of the display to address any extra recording needs the teacher may have. For example if they need to use a document camera.

We will capture the instructor using a DV camera. It will be the responsibility of the person controlling the camera to follow the movement of the instructor and create a talking head scenario.

Capturing Process
Once we have all of the footage from our recording session, we will capture the footage into editing software.

By recording the PC screen directly to the video camera, it will make capturing the files on the computer easier and more time efficient. Also, it will help the resolution quality of the screen as well.

Editing Software
There are several options for editing software my top two decisions are listed below

Adobe Director
My first choice is Adobe Director 11.5. This software is designed to create projects such as the one we are creating. It is similar in a lot of ways to Adobe Captivate; however, while Adobe Captivate is still image based, Director 11.5 includes more features for digital video.

Adobe Premiere
Adobe Premiere is sufficient for our editing needs. While it does not have the extra features that will assist in our overall project, it has all of the options necessary to adequately edit our video streaming.

An intern wrote the memo at the very beginning of the Engineering Modules project. At the time, the team had not yet selected Adobe Premier and was researching the best digital video editing software tool. The memo recommended the software tools that fit the recording procedures. The memo outlines the recording procedure to identify the kind of tool needed to match development needs.

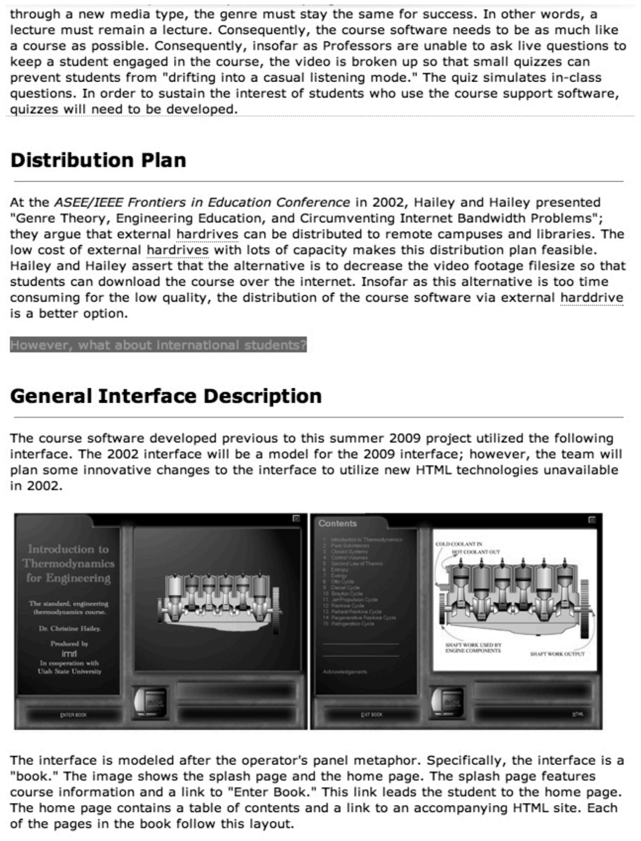
Unfortunately, the recommendation memo is the only document that details the recording procedures. The student interns should have documented something as important as the actual recording procedure itself. Instead, the student interns happened to document the recording procedure in a memo recommending video editing software. The importance of the document and the singular nature of the document's mismatched purpose only highlights the weak documentation practices of the Engineering Modules project.

Sample: Design Document

I created a Google Doc I could share with the interns. Images 17 and 18 are two pages taken from the document. I wanted the student developers to document as they went so that we could organize our efforts in a guided, traditional manner. Consequently, I created a list of potential features as placeholders until we knew more about what to document in those sections. At the time I started the document, I wanted to get going on understanding the interface development but I did not update the document. Part of the reason I did not update the document was because there was a conspicuous absence of documentation activity from the team. They were filming and editing; they were not developing the web interface. In fact, they showed no signs of taking up the interface through which students would use the modules.

Image 17

Page One of the Traditional Design Document.



through a new media type, the genre must stay the same for success. In other words, a lecture must remain a lecture. Consequently, the course software needs to be as much like a course as possible. Consequently, insofar as Professors are unable to ask live questions to keep a student engaged in the course, the video is broken up so that small quizzes can prevent students from "drifting into a casual listening mode." The quiz simulates in-class questions. In order to sustain the interest of students who use the course support software, quizzes will need to be developed.

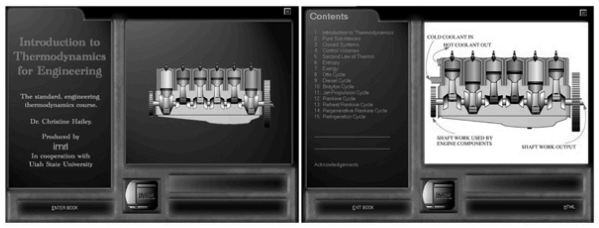
Distribution Plan

At the *ASEE/IEEE Frontiers in Education Conference* in 2002, Hailey and Hailey presented "Genre Theory, Engineering Education, and Circumventing Internet Bandwidth Problems"; they argue that external hardrives can be distributed to remote campuses and libraries. The low cost of external hardrives with lots of capacity makes this distribution plan feasible. Hailey and Hailey assert that the alternative is to decrease the video footage filesize so that students can download the course over the internet. Insofar as this alternative is too time consuming for the low quality, the distribution of the course software via external harddrive is a better option.

However, what about international students?

General Interface Description

The course software developed previous to this summer 2009 project utilized the following interface. The 2002 interface will be a model for the 2009 interface; however, the team will plan some innovative changes to the interface to utilize new HTML technologies unavailable in 2002.

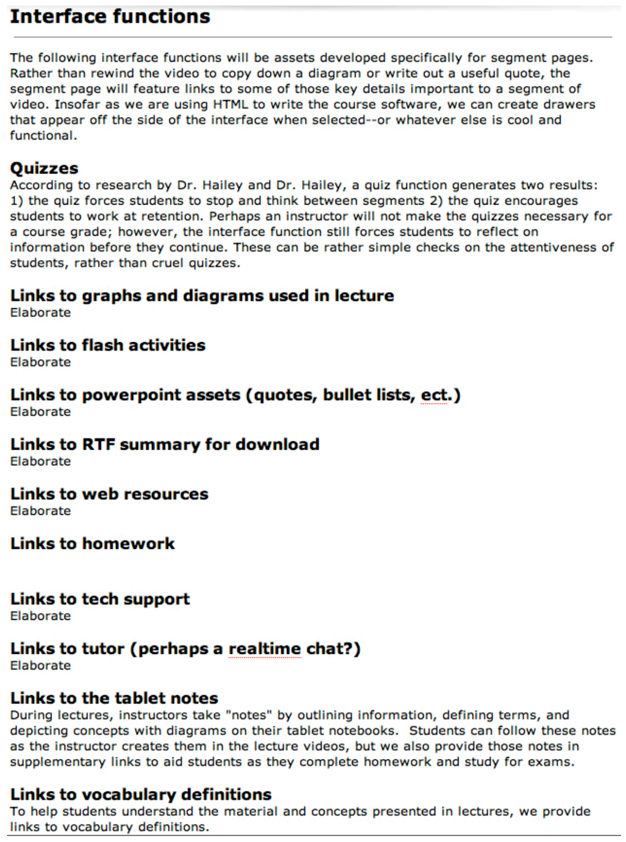


The interface is modeled after the operator's panel metaphor. Specifically, the interface is a "book." The image shows the splash page and the home page. The splash page features course information and a link to "Enter Book." This link leads the student to the home page. The home page contains a table of contents and a link to an accompanying HTML site. Each of the pages in the book follow this layout.

When I set up the document, I had listed features that were specific to an interface developed for the same purpose in 1998; the modulated design of that 1998 set of modules was supported by research (Hailey & Hailey, 1998). In addition, I listed functions that operationalize capabilities mentioned by Engineering Faculty as we introduced the project to a wider range of teachers. I felt that the team could be elaborating the various sections as we worked out a draft of the interface. These elaborations would be messy; many of the elaborations would be moved to an appendix (a location for archived development drafts).

Image 18

Page Two of the Traditional Design Document.



Interface functions

The following interface functions will be assets developed specifically for segment pages. Rather than rewind the video to copy down a diagram or write out a useful quote, the segment page will feature links to some of those key details important to a segment of video. Insofar as we are using HTML to write the course software, we can create drawers that appear off the side of the interface when selected--or whatever else is cool and functional.

Quizzes
According to research by Dr. Hailey and Dr. Hailey, a quiz function generates two results: 1) the quiz forces students to stop and think between segments 2) the quiz encourages students to work at retention. Perhaps an instructor will not make the quizzes necessary for a course grade; however, the interface function still forces students to reflect on information before they continue. These can be rather simple checks on the attentiveness of students, rather than cruel quizzes.

Links to graphs and diagrams used in lecture
Elaborate

Links to flash activities
Elaborate

Links to powerpoint assets (quotes, bullet lists, ect.)
Elaborate

Links to RTF summary for download
Elaborate

Links to web resources
Elaborate

Links to homework

Links to tech support
Elaborate

Links to tutor (perhaps a realtime chat?)
Elaborate

Links to the tablet notes
During lectures, instructors take "notes" by outlining information, defining terms, and depicting concepts with diagrams on their tablet notebooks. Students can follow these notes as the instructor creates them in the lecture videos, but we also provide those notes in supplementary links to aid students as they complete homework and study for exams.

Links to vocabulary definitions
To help students understand the material and concepts presented in lectures, we provide links to vocabulary definitions.

Unfortunately, the general design document had two problems: (1) the document was meant to be general but actually favored only one of the classes we were developing—we should have had a document for each of the four courses, as well as an overview document—and (2) the document sought to plan the project before we even knew what the project was. The general design document I started in Google Docs was clearly a pre-planning document that would serve the traditional method of software programming, rather than the contemporary methodology I was trying to study.

EVOLUTION OF SAMPLES FROM THE ENGINEERING MODULES PROJECT

The Engineering Modules Project was a troublesome project because from the beginning there was no documentation. Insofar as my objective was to obtain documentation, the absence of documents made me nervous. Based upon request, I was able to secure samples of one document or another but the student interns were clearly busy with 28 hours of editing each week. While the student interns did not seem any busier than my team during the American West Heritage Center tour game development, the situations, and community were both different nonetheless.

I expected to find the documentation solution all summer but I had to shift my expectations on numerous occasions. By the end of the summer, I was on the look out for any rhetorical forms, as opposed to traditional documentation.

High Stakes for No Documentation

Documentation plays a key role in development, even if I shifted my expectations in regards to the documentation samples I expected to find. The purpose of traditional documentation still serves contemporary developers, even if the form and content of the rhetorical form varies more than traditional development.

The following two stories showcase a great need for traditional documents on development projects like the Engineering Modules. Where the team produced a document in Story One, the team did not produce a document in the Story Two. The failure to document can generate problems on a project because professional teams would rely solely on verbal recollections and explanations. A written document would insure that there would be no need for verbal explanations. At the same time, the verbal explanations are the reason nothing catastrophic happened on the Engineering Modules project; the student interns team was small enough and worked close enough that oral communication stabilized the community.

- Story One. I asked one intern directly to produce a specific document for HTML tagging standards and the reflection on why the team chose the standardization. One set of interns innovated the process and standard; the other set needed the document for the transmission of a new procedure and for establishing consistent both direction and reliable performance.
- Story Two. One set of interns worked most of the morning finding the right way to export and import a string of files, while retaining the most performance and quality. Their decision was important for the project. The other set of interns was scheduled to work that afternoon and needed to read the decision, as well as the relevant procedures for the file conversion. If the file type and new procedure were not transmitted then one of the two teams would have to redo the editing.

The two stories illustrate near catastrophic moments during development. Close proximity and rapid cycles of development made all the difference for the student interns. While the student interns did not write documentation in both cases, the oral communication still managed to transmit design decisions to the team.

No community ownership. I had hoped to focus my research and observations on the written design documentation I could describe with the meta-language—the grammar of genre theory. In fact, I had hoped the Engineering Modules project would have given me another example I could match with the fantastic success of the American West Heritage Center tour game project.

Unfortunately, the project landscape changed. The interns were not writing documents about either filming or editing. More significant to my project expectations, the team would never write the web interface documentation. In fact, my advisor took the role of lead designer for the interface and completed the interface design over the weekend. He simply did not need the whole summer and four interns to create a Flash interface. While this was a welcome change, insofar as the four students filming and editing would not ever have had the time to design any interfaces, this also limited how much design documentation I could gather. My response to the absence of documentation characterized the way my document expectations continued to shift throughout the summer months: I expected traditional, comprehensive documentation; then I expected on-assignment documents—like a procedural memo; then I expected a wall full of user stories on index cards; then I expected to find handwritten artifacts that supported the project's stability and meaning-making.

Some professional teams simply do not document. One valid explanation of this perspective is the rapid change of design. After all, if the design changes so rapidly why have documentation in the first place? In this case, the rhetorical forms were pieces of communication. The scraps of paper, checklists, emails, meeting notes, and sticky notes all worked together simultaneously to sustain the meaning-making community. Consequently, the pieces of communication were not the rhetorical form responsible for the project's dynamism.

After the loss of traditional documents, the absence of contemporary index card stories, and the scarce pieces of communication, I was left wondering how the interns:

a. Sustained dynamically recursive activities
b. Cultivated situated knowledge
c. Adapted the form/content of their rhetorical forms
d. Sustained and reconstituted their project structure
e. Maintained community ownership of common rules and values

I needed to alter what samples I expected to gather, insofar as the interface was complete and the lead designer did not need a formalized document to do it. The project had so many parts and there was still time to expect documentation from some other component of the project plan. When there was no documentation, I looked for statements of work on index cards (Beck, 2000). When there were not any statements of work, I started scanning scraps of paper and sticky notes. As I demonstrate later, the rhetorical form I could describe with genre theory was oral communication.

IMMINENT CATASTROPHE FOR THE ENGINEERING MODULES PROJECT

From the perspective of traditional documentation values, the Engineering Modules project was a failure. I was coordinating a project meant to replicate the successful American West Heritage Center tour game documentation practices; I could not get the participants to write in the same collaborative fashion—no community ownership. In fact, I tried to train the interns with the Google Doc method I used for the tour game development project.

A Need for Project Documentation

After working with the editing software together all morning, I met with the interns about the development of the course interface documentation. I explained my plan to detail the general design document and showed them the initial skeleton. I wanted to start with identifying the things students would need to do with the interface—the features; the interns would elaborate as the team built a greater understanding of the project requirements. I expected many requirements would be deleted and others would be added. While I did not expect any work on an interface until the end of the summer, I saw no reason why the team could not record insight into the interface's projected requirements. Even as development of the interface began, I expected the interns would find the document as the best place to find the next development needs and the current development trajectory. Therefore, I wanted a messy space of communication and collaboration; reliance on the document would stabilize meaning-making, even while restructuring with individual contributions and innovation.

Different situations. While I was disappointed the corpus of documents paled in comparison to the American West Heritage Center tour game project, I realized they were two different situations (the genre triad of purpose, situation, and community). In one situation, the community of student developers was remote, without any common workspace; in the other situation, the community of student interns shared schedules and workspace—in fact, they shared the same computer. According to the purpose, situation, and community of genre theory, the rhetorical forms would necessarily be different; however, I was thinking too traditionally to look for anything but written documents.

Engineering Modules were Not a Catastrophe

It still remained to be seen if the difference was a new rhetorical form or simply a communication failure. At the time it seemed the latter, but the way the team avoided catastrophic failure suggested otherwise. The project could not be a communication failure if there was so much communication, even if none of it was written. There was something else that stabilized the situated knowledge and bound the community together. The Engineering Modules project kicked off the moorings of traditional documentation, had no known rhetorical forms to stabilize the project's meaning-making and there was still a complex situation at stake.

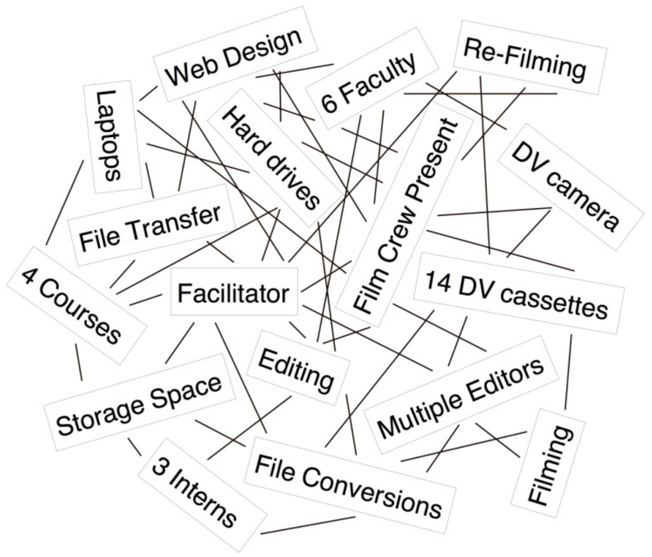
Industry guides might be clear about how to write documentation but developers do not follow the genre rules the guides outline. Consequently, I wondered what happened to contemporary development projects when developers abandoned the industry guidebook recommendations. So what did I do about it? Without traditional documentation practices, was the Engineering Modules project a catastrophic disaster waiting to happen?

Complex Situation

I facilitated a very complex situation that required enormous communication and coordination, without the necessary written documents to stabilize the Engineering Modules project. The avoidance of catastrophe was evidence alone that there was another rhetorical form that I was overlooking. Image 19 represents all the project components; much like Clay Spinuzi's genre ecologies, image 19 traces activities and their products and highlights the chaos that was stabilized by rhetorical forms I could not describe.

Image 19

The Chaotic Project Dependencies that Ended up Successful.



The project involved recordings for four Engineering courses. There were four-core faculty (an additional two faculty to share recording time), two live courses, two faculty to oversee the project, one graduate intern, two undergraduate interns, and myself as facilitator. As the facilitator on the project, my responsibility was to schedule recording sessions and rerecording sessions. The interns divided their team between filming the courses and editing the courses they filmed. Each filming session required the following equipment: notebook laptops, software (Camtasia) to record all activity on the laptop, a DV camera, up to two cassette tapes per lecture, and two wireless microphones to record audio for both the Camtasia software and the DV camera.

When the students returned to the office to edit they had to first import the DV footage to a computer in real time, use another computer to convert the Camtasia footage in real time and continue editing previously recorded lectures on yet another computer. External hard drives were used to transfer converted footage files to the editing computer, as well as to back up the courses. There was a 250 gig external drive for each of the four courses, as well as an internal drive for each of the courses. There was also the hard drive for each of the three computers, in addition to two other laptops rotated among the faculty for the live courses. Finally, there was an additional 1 terabyte external drive. In all, there were 15 drives to manage information on the project.

Example of a static development task. The interns needed protocols in order to transfer DV footage before reusing cassettes. Any one cassette could pass between both teams; when a team grabbed available tapes, a protocol would assure the student interns would not grab a cassette not yet uploaded to the computer. Without commonly understood protocols, the team could record over critical footage:

1. All footage needed to be converted and transferred immediately upon return to the office after a recording session;
2. Cassettes were stacked next to the importing computer;
3. Cassettes were stacked near the editing computer after importing;
4. The students needed to verify that the footage from both Camtasia and the DV camera were successfully recorded;
5. The interns needed to track the status of any footage: importing, converting, transferring, editing, imported, converted, transferred or edited.

Protocols and tracking were necessary because at the midpoint of the project the team was filming at least 14 hours a week. Consequently, 28 hours of footage (Camtasia and DV film combined) were being converted per week. Finally, interns had to be editing as fast and efficiently as possible to stay on top of the incoming recordings. The fact that we rarely refilmed was a miracle considering 28 hours of filming, 28 hours of processing and 28 hours of editing had to be managed every week by four intern students.

Despite all the activity and potential for human error, the student interns avoided catastrophe because they were in fact communicating with contemporary rhetorical forms.

RHETORICAL FORM SAMPLES IN THE ENGINEERING MODULES PROJECT

The project was not necessarily a catastrophic disaster waiting to happen because, rather than a failure of communication there was in fact a team of students who communicated really well together. The communication all happened at once because the tightly knit team worked in such close proximity—rapidly implementing changes, even while they made the respective decision. Simultaneous documentation is a significant change because all the documentation is done in nontraditional scraps, rather than an enormous governing document generated before development even begins. The pieces of communication were rhetorical forms that stabilized the project simply because the community was so small and worked in such small cycles of development.

The following are five samples of the rhetorical forms in the Engineering Modules project:

- Email and Calendar Updates
- Meta Data Documents
- Scrap Paper Notes
- Sticky Notes
- Oral Communication

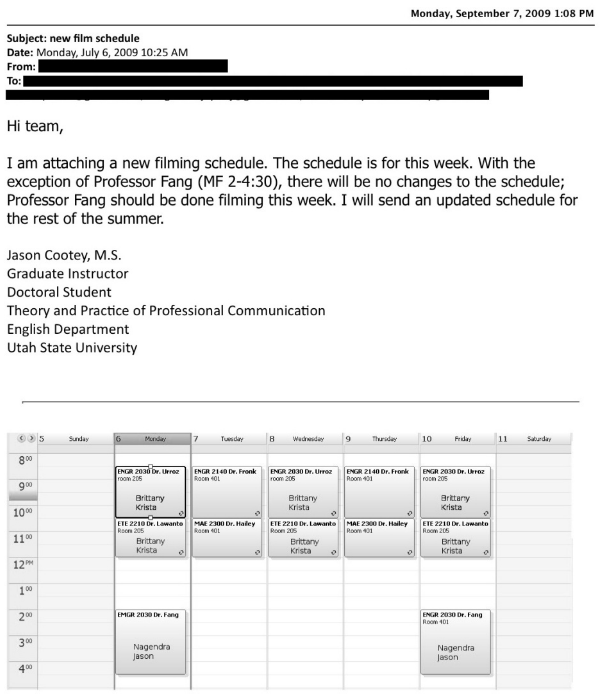
They are each samples of the simultaneous communication that sustained the entire project.

Email and Calendar Updates

I sent emails like image 20 every week to update all faculty and student interns of weekly schedule changes. I altered calendar images in Photoshop to organize the team's filming schedule. This was a necessary step because the schedule changed every week.

Image 20

An Email with a Weekly Calendar.



Monday, September 7, 2009 1:08 PM

Subject: **new film schedule**
Date: Monday, July 6, 2009 10:25 AM
From: [redacted]
To: [redacted]

Hi team,

I am attaching a new filming schedule. The schedule is for this week. With the exception of Professor Fang (MF 2-4:30), there will be no changes to the schedule; Professor Fang should be done filming this week. I will send an updated schedule for the rest of the summer.

Jason Cootey, M.S.
Graduate Instructor
Doctoral Student
Theory and Practice of Professional Communication
English Department
Utah State University

	5 Sunday	6 Monday	7 Tuesday	8 Wednesday	9 Thursday	10 Friday	11 Saturday
8:00 – 10:00		ENGR 2030 Dr. Urroz room 205 Brittany Krista	ENGR 2140 Dr. Fronk Room 401	ENGR 2030 Dr. Urroz room 205 Brittany Krista	ENGR 2140 Dr. Fronk Room 401	ENGR 2030 Dr. Urroz room 205 Brittany Krista	
10:00 – 11:00		ETE 2210 Dr. Lawanto Room 205 Brittany Krista	MAE 2300 Dr. Hailey Room 401	ETE 2210 Dr. Lawanto Room 205 Brittany Krista	MAE 2300 Dr. Hailey Room 401	ETE 2210 Dr. Lawanto Room 205 Brittany Krista	
12 PM							
1:00							
2:00 – 4:00		EMGR 2030 Dr. Fang Nagendra Jason				ENGR 2030 Dr. Fang Room 401 Nagendra Jason	

We started with a general calendar but that quickly became useless. Not only did schedules frequently change but I had to reschedule film sessions where there were conflicts. Consequently, one general schedule for the summer was not feasible. The team moved to a standard schedule oriented around the development team and the availability of equipment—a schedule for student intern shifts and to whom is assigned what equipment at any one time. However, we soon streamlined the process so that even the standard schedule was useless; the student interns had allotted specific time in their day for their internship so that their availability was reliable throughout the summer.

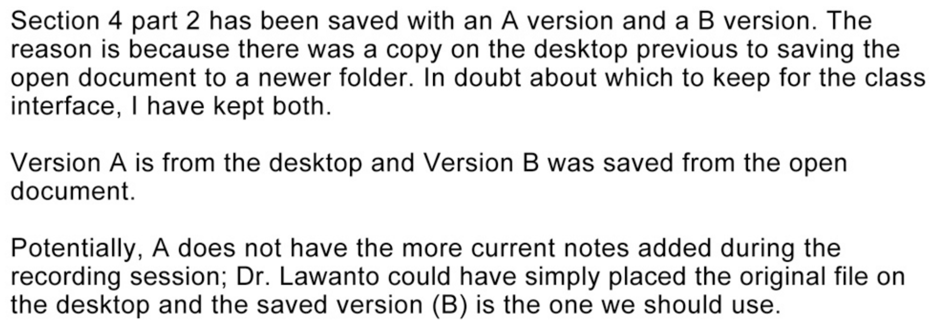
The only thing that was needed was a weekly schedule that connected filming sessions with student interns; the project needed a more flexible way to adapt the schedule for so many parties.

Meta Data Documents

Image 21 showcases a meta data sample is notation about another edit that had a version control issue. The student interns retained film files on cameras but did not always have the laptop used by the professors. This is because once the screen capture was obtained from a laptop, the screen capture file was stored on the appropriate drive—inside the appropriate course folder—inside the class time and date folder—inside the footage folder—inside the Camtasia folder.

Image 21

Meta Data File Tracks a Versioning Error.



Section 4 part 2 has been saved with an A version and a B version. The reason is because there was a copy on the desktop previous to saving the open document to a newer folder. In doubt about which to keep for the class interface, I have kept both.

Version A is from the desktop and Version B was saved from the open document.

Potentially, A does not have the more current notes added during the recording session; Dr. Lawanto could have simply placed the original file on the desktop and the saved version (B) is the one we should use.

In this case, the Camtasia folder for a particular lecture had two Camtasia files and this meta data document explained a versioning conflict and how it was resolved. The versioning was resolved so there was no enduring value to this document sample but it was still a rhetorical form, even if it had a very temporary purpose, situation, and community.

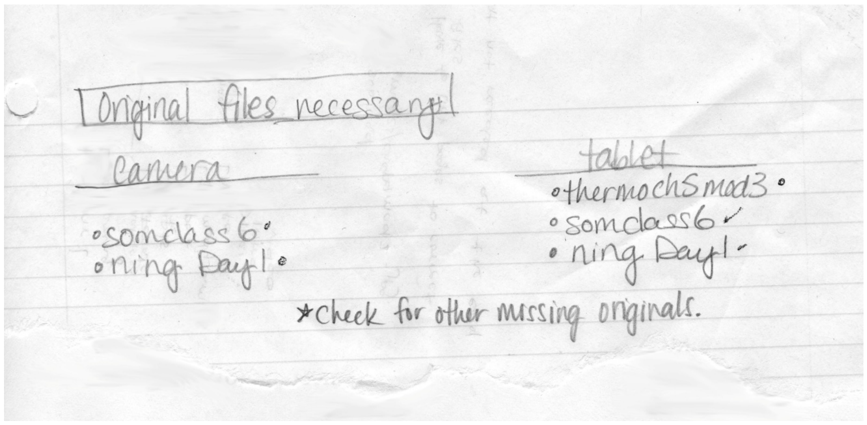
Pieces of communication like this meta data sample were located where they were needed when they were needed. However, there were not enough files like this to argue that there was an official protocol for writing meta data; rather, this was just one file that explained an atypical arrangement of files in that one single folder.

Scrap Paper Notes

There were several scraps of paper like image 22. Mostly, the scraps were piled behind the monitor of the editing computer. The scraps of paper were often "note to self" in nature. The interns needed to communicate asynchronously, while they shared the one editing computer. These pieces of communication facilitated the coordination of the editing process. This sample in particular dates back to when the team was still working out an organization plan for the same lecture footage from the different recording devices.

Image 22

Scrap of Paper for the Use of a Single Editor.



Original files necessary!

Camera	tablet
	thermochSmod3
somclass6	somclass6 ✓
ning Day1	ning Day1

*Check for other missing originals.

Both the camera cassette tapes and the laptop had digital footage. We figured out a protocol for managing the footage before the heavy filming began. In addition, we developed digital file organization and expanded our storage capacity by increasing external hard drives. This scrap of paper predated all that organizational decision-making and signals the documentation that should have been available. Those organizational solutions might have been something to document, if not for increasing demand for altering the organizational and storage solutions. Once heavy filming began we had more devices and more storage needs; consequently, the solution was reformulated to meet the resource needs of each new week.

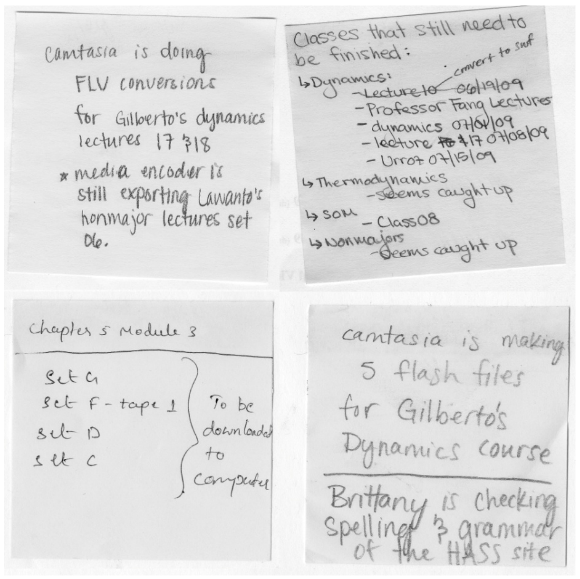
Scraps of paper like image 22 were the rhetorical form the student interns needed while they met project needs. That said, there were not many scraps of paper like image 22 because the student interns used another rhetorical form to stabilize their recursive situation—every time they came back to the editing computer they would have to manage another filming session of film cassettes, laptop footage, and more storage needs.

Sticky Notes

Sticky notes were affixed to the actual cassette tapes. We were recycling 14 digital videocassette tapes and it was absolutely important that footage was processed before the cassette tape was reused. Image 23 showcases sticky notes identified tapes that were not ready for reuse.

Image 23

Four Sticky Note Samples.



camtasia is doing
FLV conversions
for Gilberto's dynamics
lectures 17 & 18
* media encoder is
still exporting Lawanto's
nonmajor lectures set
06.

Classes that still need to
be finished:
↳ Dynamics:
~~-Lecture 10~~ 06/19/09 · convert to swf
- Professor Fang Lectures
- dynamics 07/01/09
- lecture ~~16~~ #17 07/08/09
- Urroz 07/15/09
↳ Thermodynamics
- seems caught up
↳ SOM - Class08
↳ Nonmajors
- seems caught up

Chapter 5 Module 3

Set G
Set F - tape 1
set D
set C
To be downloaded to computer

camtasia is making
5 flash files
for Gilberto's
Dynamics course

Brittany is checking
Spelling & grammar
of the HASS site

The "chapter 5 module 3" sticky note—the lower left sticky note—is a great example of how pieces of communication stabilized the situation, and community. Towards the end of the summer, the department replaced glass windows throughout the entire building. The office in which we worked was temporarily unavailable while the workers did their jobs. However, we still had to take the equipment to keep up with our ongoing filming project. After all, some of the engineering classes were live and could not stop for the replacement of windows in the Department of English building. Consequently, we had a stack of original footage that was not backed up, while windows were replaced in the room where the team's editing computer was located. We were able to return to processing and editing tapes once the windows were replaced.

This sticky note kept a tally of the tapes that were not reusable so that no one inadvertently grabbed one for a film session. The other sticky notes are also examples where interns wanted to mark the status of their footage.

Oral Communication

The fifth sample is not something that we expected. We did not expect conversations to be so important to our development project. Oral communication is perhaps natural enough that it is overlooked in development practices. Our oral communication occurred on the telephone, while walking to a film session or while transferring footage to the appropriate hard drives. We made decisions at those times and implemented those decisions as soon as an hour later at the next film session. Unfortunately, there were no logs of this communication. Clay Spinuzzi experienced the same problem and did not include oral data in his research because oral data is not easy to capture or examine (2002). Consequently, I would have had to record, transcribe, and code the transcripts for both the Engineering Modules development team and interactions with the faculty with whom I coordinated film sessions. Even if we had anticipated oral communication, we would have had to record a full day of conversation, every day, for the entire summer—on top of the aggressive film schedule.

Before any Engineering Module filming began, Dr. Hailey, the four interns and I were testing equipment and identifying needs. For instance, we discovered the film booth camera was not designed to convert Adobe Flash file types, without degrading the quality. In addition to picture quality, we needed new equipment and modded recording devices. None of this could have been mapped out from the beginning. Rather, we figured out the solutions as we uncovered the problems; we spent so much time walking back and forth across the campus there was no time to write what we had already decided en route anyway.

I cannot help but consider the possibility that the student interns simply did not document like they ought to and that I failed to set the right community ownership from the very beginning. However, the simple fact is that the rhetorical situation necessarily involved a considerable amount of decision-making while walking across the campus. Once the students were back at the office the team had to break down the equipment and connect the cameras to various computers and the instructor-laptop to the appropriate external drive. Once the equipment was all set to process the film footage, the community would immediately divide up for off-campus jobs or summer courses; the student interns would exchange any necessary information with speech acts before leaving.

SUCCESS OF THE ENGINEERING MODULES PROJECT

When I started the Engineering Modules project, I wanted to simulate a development workplace and obtain samples of the documentation. While I did succeed in simulating a development workplace, the team did not write the documentation. In fact, other than sticky notes and standard operation procedure documents, the team did not produce much documentation at all. The explanation is that the team did not use a traditional process to develop the modules for the department of Engineering. Instead, the team used contemporary development methods and that means there was no major design documentation. All the writing was in simultaneous pieces of communication; in some cases, our documents were literally scraps of paper. However, the reason that our project was a monumental success, rather than a catastrophe, was because of our face-to-face communication. Our close proximity in our small workspace afforded a lot of oral communication. Speech acts were a compelling addition to the rhetorical forms developers use in their documentation activities; I was struck by the importance of oral communication in contemporary development practices. I had found the missing rhetorical form with which contemporary development teams stabilize meaning-making knowledge and activity in their communities.

Engineering Modules Repercussions

Yet, there is a limit to oral communication as the missing rhetorical form I can describe with genre theory. Those repercussions are worth acknowledging because they lend significance to the rhetorical forms I observed in the English Modules project. Even then, the meta-language bucks up against some of its own limitations in describing oral communication.

Scope of oral communication. For instance, the concept of oral communication is as vague as the concept of written communication. Oral communication constitutes multiple genres and is almost nearly too generalizeable to be considered a research conclusion. However, I specifically refer to a concept of oral communication distinguished by genre theory. I refer to a rhetorical form that has specific repercussions beyond the scope of this dissertation.

Verbal vs. nonverbal. The term "written" refers to pens, pencils, and paper. Once graphics became a reliable and accessible form of communication, scholars started distinguishing between verbal and nonverbal communication. Verbal refers to written communication that requires words; nonverbal refers to written communication that does not require words.

The researchers publishing in *Technical Communication* refer to verbal communication they refer to written documents, as opposed to graphics or art. They limit the meta-language to written communication because researchers of written communication research are really keen on written communication. While written communication researchers are no longer limited by the ink of a printing press, they should also use the meta-language to formulate more precise references to "non-verbal" rhetorical forms.

Contemporary communication. In the world of English research, the concept of text is very interesting. Text is not just the written word on paper or a computer screen. Rather, text is an array of patterns that authors and readers use to generate meaning together. This kind of definition might provoke many English researchers. For instance, the use of the word patterns is a problem because researchers might prefer to use the word “symbols.” The use of the word “meaning” itself would cause debate about the source of meaning in a text and who (author or reader) is actually responsible for meaning. In fact, some researchers might even be troubled that I suggest the author is involved at all. After all, meaning does not reside in a text as a clear transmission from an author because the text does not mean anything until a reader reads the text. That is why I use the phrase “generate together” instead of “transmission.” At the same time, I unfortunately already imply the word “read” because there are many kinds of texts that are not read.

The result is that just about anything is a text—is rhetorical. A movie is a text. The script is a text and the film reel is a text—full of disjointed frames for which the viewer is responsible to form together as an illusion of movement. A radio broadcast is a text. A news report, blog, Facebook page, chat log, course lecture, and an annotated schematic of an engine are all texts. A painting, a landscape photo, and a slideshow are texts. Even more, the actions players perform in a computer game, the patches on a family quilt, the pink flamingos decorating a lawn, and a child’s crayon scribbles are all texts.

Yet, even though the word choice is debatable, a text is an array of patterns that authors and readers use to generate meaning together. Why is all this so important? With all these kinds of texts, a meta-language is necessary to sift through them all. Finally, in a research world where everything is a text, the meaning-making oral communications in contemporary development are the new text.

MODEL OF EXPECTATIONS IN THE ENGINEERING MODULES PROJECT

While I expected written documentation, I ended up with several rhetorical forms and a strong component of oral communication. Consequently, the decision I have is whether to apply the model of expectations to the documentation I expected or the oral communication I received. Ultimately, the Model of Expectations is meant for analyzing the rhetorical form that carried the project. Much like the narrative documentation in the American West Heritage Center tour game, the oral communication carried development for the Engineering modules. The following Model of Expectations and discussion relates to the oral communication that supported the project's success. In each of the expectations, I try to identify where the documentation failed and where the oral communication filled the void left in the community.

Model of Expectations

Much of the communication happened walking between the Department of Engineering building and the Department of English building. We collaborated while setting up equipment and the interns worked in pairs on the video they filmed together. The next time we would cross the campus there would be new problems; in addition, decisions from the previous day had already evolved into new obstacles. Every activity was so intimate and every conversation was so relevant that oral communication was the rhetorical form that stabilized the Engineering Modules project.

Dynamism	Dynamism for Practitioners
• Genres are developed from actors' responses to recurrent situations. • Genres serve to stabilize experience and give it coherence and meaning. • Genres change over time in response to their users' sociocognitive needs.	• The document forms as the team uses it and responds to it. • The document is a common resource for teams and is a foundation for the innovative solutions teams require. • The document changes with team decisions so that it is a decision making tool.

Unlike the American West Heritage Center project, the Engineering Modules project did not have sufficient documentation. The Flash interface was developed without any legacy information. The HTML tags were documented for consistency at my request but that legacy document did not impact tagging procedures. Each week I released a new calendar because the film schedule would accommodate at least one faculty member's schedule change. Therefore, the documentation did not stabilize what was sometimes nothing more than chaos. In addition, while the team had plenty of sociocognitive needs, the documentation was not live and did not adapt to those needs.

There were written rhetorical forms the team found useful while they developed. For instance, the sticky notes served as documentation. However, rhetorical forms were limited to temporary information relevant to a specific edit on a specific day. After that context-sensitive situation was passed, there was no recursion because the rhetorical situation was vastly different the next day.

Rather than write about tagging standards, the team simply worked close enough that they knew the standards. Instead of preplanning recording procedures, the team discussed improvement each day on the walk back to the computer room, after filming. When the team selected media file types to preserve the best quality, they talked through their decision together and did not write up any spec documents. The common rhetorical form that brought stability and meaning-making to the project was the oral communication.

Situatedness	Situatedness for Practitioners
• Genre knowledge is derived from and embedded in our participation in the communicative activities. • Genre knowledge is a form of "situated cognition." • Genre knowledge continues to develop as we participate in the activities of the ambient culture.	• If it isn't documented then it didn't happen. The act of documentation is the formation of common knowledge. • Project productivity and long-term goals originate from documented knowledge. • Documents become more comprehensive and typified as the community collaborates and tests the document's relevance.

I tell all my developer friends that bus drivers have a bad rap. Professionals, without fail, reference the importance of documentation by accusing bus drivers: "I have to document because what if I got hit by a bus on the way to work." Insofar as a bus could have hit any one of the interns on the way to the campus for a day of filming, it seems to have been a mistake to go without documentation for an entire summer. However, the interns were embedded so closely together that their proximity was the location of the team's situated knowledge. They had acculturated each other into their knowledge system.

In other words, the interns were always talking and collaborating so that they all knew the project's procedures based upon their active participation in such a small community of six individuals. Even when one intern was absent, the other members of the team knew how to fill the role because knowledge of the role was stored in the closeness and agility of the team.

The sticky notes and other pieces of communication might have all stored project knowledge but the sticky notes typically served only one member of the team.

Consequently, the one singular activity represented by the sticky note might have misdirected the entire team—the collateral damage of a bus accident—but the community knew enough together to adapt.

Form and Content	Form and Content for Practitioners
• Genre knowledge embraces both form and content. • Genre knowledge is a sense of what content is appropriate to a particular purpose, situation, and time.	• Form and Content conform to documented knowledge. • Document-based knowledge prompts decisions about relevant rhetorical content.

Rather than point out there was no form and content because there was not any documentation, I want to emphasize that the importance of form and content to a genre is not because of paragraph content and the structure of headings. Rather, form and content is the rhetorical form in which the genre knowledge resides. In the case of the Engineering Modules project’s oral communication, the student interns morphed the “documentation” activity to the needs of their on-the-run community.

In contrast, the general design document I tried to create was initially built as a space for written communication and active, ongoing collaboration. I wanted situated knowledge in a written document because that was what developers should do. However, the team’s situated knowledge was already based on oral communication and my general design document was already inappropriate for the purpose, situation, and community. A written document was a lumbering monolith in an extremely contemporary community that shifted their community knowledge every week—sometimes every day.

Duality of Structure	Duality of Structure for Practitioners
• Genre rules inform activities that constitute social structures • Genre rules inform activities that simultaneously reproduce these structures.	• Project planning, roles, and responsibilities match procedures established in the documentation. • The documented rules guide procedural and organizational decision-making.

Every morning the team would work through the checklist of equipment required for a film session. In time, the team did not need the checklist document. That was not necessarily because any one member had memorized the list; rather, the team did not need the list because they had roles and those roles required specific equipment. The interns would know what equipment to take based on the activities they expected to do. They knew what activities they were expected to do because they negotiated those roles as they entered the computer room. They would reconstitute the "film crew" every time they had to configure the session's team, based upon a brief discussion about who was doing what with which equipment.

The team did not update a "roles and responsibilities" document every morning and they did not maintain a log to track all the possible roles for any one intern based upon varying configurations of "film crew." Such documentation would have been superfluous to the orally communicative activities that already sustained and reconstituted the team's structure.

Community Ownership	Community Ownership for Practitioners
• Genre conventions signal a discourse community's norms, epistemology, ideology, and social ontology.	• Philosophy of development methodology expressed through documentation and practice.

I began to despair when there was not traction for the general design document I created in Google docs. I knew there would not be community ownership and that I was on a team destined for documentation failure, as opposed to successfully replicating the documentation of my previous project.

Yet, I was the one with the wrong norms, epistemology, ideology, and social ontology. If genre = purpose + situation + community then I was in the wrong community until I could understand what rhetorical forms the Engineering Modules project actually required. I tried to impose my values and train the interns to my documentation mindset but the project had unique norms and epistemology. I simply needed to adapt myself and become an owner of a different set of rhetorical forms.

Discussion from Dated Blog Posts

I recorded project progress in a Wordpress blog (iseethecode.wordpress.com). I wanted to keep a research journal to which I could look back and find original reflections. My blog entries are dated reflections on development practices. More particularly, they are an evolving insight into development practices. Over the course of three months, I went from traditional documents to messy collaboration documents to oral communication. I find that these blog entries are the best way to discuss the results of the Engineering Modules project.

A discussion entry. On August 29, 2009, I blogged about the data I acquired from the Engineering Module project. I have edited the blog entry so that it is matches the tone of this chapter. I have incorporated the meta-language; for instance, I change “verbal communication” to “oral communication” so that the blog entry is more consistent with the meta-language of genre theory.

I have samples of weekly reports and massive amounts of email; I have scanned sticky notes, scraps of paper, and pages of notes scribbled on the back of random recycled paper. The thing that is really interesting about the absence of documentation was the oral communication that transpired in that void. The purpose of design documentation is supposed to facilitate meaning-making and leave a record with which teams can negotiate progress. The Engineering Modules project did not falter because the project cycles were so rapid and the team worked so close together (sharing the same equipment) that oral communication was all the team needed—seeing how the interns worked shoulder to shoulder. There was no need for a hub of communication similar to the American West Heritage Center tour game project.

I'm nervous about this argument because professional technical writers around the world will lament the advocacy of a technical writing vision that highlights the conspicuous omission of writing. I don't even entirely agree with the argument; I have too many questions. For instance, I wonder what kind of criteria or rules I can identify to describe a genre of oral software design conversations? I also wonder, if purpose is so important to a genre, how can oral communication that serves none of the purposes of design documentation replace design documentation? Perhaps I just need to back down from oral communication as an overlooked genre in contemporary software development. However, developers don't walk

> around mute and carry excerpts from the design document to hold up when they need to communicate. Rather, they do a lot of talking.
>
> More specifically, the sticky notes, memos, scraps of paper, weekly reports, and design documents all make up communication on a contemporary development team, along with the oral communication.

Reflection of original thinking. On June 3, 2009 I posted to my blog about progress on planning for the design of the web interface. I had set up the Google Doc general design document for the Engineering Modules project and anticipated an exciting collaboration. I reflected that my conception of software design documentation was very different than either guidebooks or practitioners. I envisioned a messy space of collaboration. I did not think of development documents as finite, presentable packages because the collaboration is ongoing until the software is deployed; development documents are sites of live communication. In fact, I wrote about how my vision of documentation is a paralogic hermeneutic (Kent, 1993) process that casts off codifiable trimmings and reduces documentation to an ongoing negotiation with changing stakeholders in a rhetorical situation. I commented that a messy space of collaboration actually limits the multiple strands of chaotic independent development because the document is the source from which individuals identify potential innovations.

On June 5, 2009 I blogged about how the American West Heritage Center tour game documentation was that paralogic hermeneutic process. On June 5, I listed the six negotiations that made the project paralogic:

1. We were developing and designing at the same time; our team was more productive and effective than other teams because we all could track a unified development effort
2. We worked remotely and uploaded our work so that everyone else could develop at the same speed
3. We were remotely developing with the most updated assets and elaborations so that we easily kept up with the one-week contemporary sprints
4. The team was full of motivated graduate students in a demanding course that assessed development participation so that my team felt pressured to deliver every Tuesday
5. We could count on updated elaborations as we divided up development responsibilities and relied on the progress of others in order to complete our own responsibilities
6. The absence of communication clearly impacted progress on the project and always increased pressure on one or another team member.

Five months later. On November 18, 2009 I wrote a blog entry about rereading the June entries five months later. I was struck by how much my beliefs had changed in merely five months. I did not realize that I had imposed traditional documentation rules on contemporary development methodology—whether or not it was a messy space of collaboration.

> My entire account of the first documentation blunder [referencing the general design document and Dr. Hailey's critique of it] assumes that documentation still has a prescribed role that we can fulfill if I had simply

been clear about the purpose of the document. After all, I have seen documents that clearly identify their purpose, audience, and scope in sections respectively named at the beginning of the document. In addition, if documenting before the project begins is a mistake then documenting after the project is complete is an even greater mistake. Consequently, the documentation needs to happen during the project. However, as I stated, I assumed that the documentation served a prescribed role.

We have since identified that communication on the project did not happen through the medium of a document; rather, communication happened at a much faster rate so that the medium of the document was inefficient. Instead, we [employed oral communication]. This is of course a no-brainer; [developers] talk to each other. However, a research field about writing is keener on the documents that a development team produces. In other words, I was looking to generate governing documents that articulated design plans, protocols, and detailed elaborations. However, we did not generate anything of the sort. Researchers in my field have built models of communication that are conspicuously deficient when it comes to oral communication. The benefit of oral communication was that we didn't use up development time to write elaborations we were already implementing.

I guess the moral of the story is that while I thought that clear writing was the solution on June 3, I had no idea that the solution ended up being oral communication.

Checking student practice against professional practice. As I have often suggested, it is possible that the student interns simply needed to document. It is possible that the "documentation" activities of professional developers require written documents and are best practice. Consequently, there is really no reason why the student interns should have jeopardized the Engineering Modules project. On the other hand, professional developers jeopardize their own development projects all the time and the meta-language of genre theory does not describe how they still manage to stabilize their meaning-making communities.

The rhetorical forms I identified during the Engineering Modules project demonstrate what contemporary professional developers do to manage their purpose, situation, and community. The following chapter seeks to verify whether the student interns were poor documenters or if they were showcasing a rhetorical form overlooked by the field of professional communication. The chapter features seven interviews that capture the insight of contemporary developers who seek to incorporate documentation standards into their practice. While they do not have the meta-language to reference rhetorical forms, as opposed to referencing documentation alone, they still relay the importance of oral communication.

CHAPTER VI

PREDICTING GENRES: PROFESSIONAL DEVELOPER INTERVIEWS

SPECIFICS ABOUT THE INTERVIEW

Contemporary developers employ many activities in an ongoing cycle of development; the activity of "documentation" is just one of those stages. They are the subject matter experts on their documentation activities but they necessarily speak of written communication whenever they speak of "documentation" activities. As Nietzsche suggests, their knowledge is set in a situation bound by language. I drew on the meta-language of genre theory to interview senior developers who could describe what was wrong with documentation and describe the rhetorical forms they used to document. I took seven professional developers to lunch in April of 2012. Over the course of seven scheduled lunch appointments, I produced nearly five hours of interview recordings. I talked with the developers about both their development methods and their documentation. In questioning them about documentation, I expected they would admit poor documentation practices; I expected they would advocate best practice, all the same; I expected they would relate what they do in the absence of documentation. However, where I expected to learn about oral communication practices, I also learned about other rhetorical forms they employ.

The American West Heritage Center project demonstrated what could have been a best practice example of documentation in a contemporary workplace. However, community ownership became really significant when the Engineer Modules project failed to replicate documentation in that way. There was not compliance to documentation requirements; yet, it was evident the project was not an imminent catastrophe. I observed pieces of communication, including oral communication, which successfully stabilized the community and their productivity. Much like the Engineering Modules project, the Lunch Interviews showcased a lot of pieces of communication too.

Forecasting interview results. I interviewed the professional developers to verify that they employed oral communication as their rhetorical form. More specifically, I expected that contemporary developers sustained their community knowledge by an oral communication practice that was unsupportable in traditional development methods; however, I did not expect there would be additional rhetorical forms on top of oral and written communication—pieces of communication scrawled on white boards and sticky notes. This was an important verification, insofar as I might have discovered that the American West Heritage Center project was the example of best practice after all. If that were the case, then the Engineering Modules project was a randomly successful aberration and traditional documentation really does work for contemporary developers.

The interviews in fact verify contemporary developers' "documentation" activities employ numerous rhetorical forms that are unaccepted by traditional methods that privilege written communication. In addition, those contemporary rhetorical forms are valuable when I use genre theory to predict the rhetorical forms I expect to find both stabilizing recursive situations and restructuring the rhetorical meaning-making of development communities.

Organization of the developers interviews. Rather than recount five hours of conversation, I draw the most relevant statements from the recordings. I take the lead from ethnographic researchers like Emerson et al. (1995) who empower the voice of their subjects; consequently, I rely on extensive block quoting to fill out each of the 11 points in the model of expectations. I also match the rhetorical form of oral communication to the EUPARS model; I want to measure and verify the appropriateness of oral communication in contemporary development practices.

Table 9 presents the Model of Expectations. In the case of the two postmortems, the Model of Expectations was a framework through which I could describe documentation with a meta-language. I matched each interview question to a bullet point from the Model of Expectations. Consequently, the subjects' answers to my questions are directly related to each expectation. In other words, whatever the subjects said about oral communication would directly inform each of my expectations.

Table 9

Model of Expectations Matched to Practitioner Wording

Model of Expectations	**Worded for Practitioners**
Dynamism • Genres are developed from actors' responses to recurrent situations. • Genres serve to stabilize experience and give it coherence and meaning. • Genres change over time in response to their users' sociocognitive needs.	Dynamism • The document forms as the team uses it and responds to it. • The document is a common resource for teams and is a foundation for the innovative solutions teams require. • The document changes with team decisions so that it is a decision making tool.
Situatedness • Genre knowledge is derived from and embedded in our participation in the communicative activities. • Genre knowledge is a form of "situated cognition." • Genre knowledge continues to develop as we participate in the activities of the ambient culture.	Situatedness • If it isn't documented then it didn't happen. The act of documentation is the formation of common knowledge. • Project productivity and long-term goals originate from documented knowledge. • Documents become more comprehensive and typified as the community collaborates and tests the document's relevance.
Form and Content • Genre knowledge embraces both form and content. • Genre knowledge is a sense of what content is appropriate to a particular purpose, situation, and time.	Form and Content • Form and Content conform to documented knowledge. • Document-based knowledge prompts decisions about relevant rhetorical content.
Duality of Structure • Genre rules inform activities that constitute social structures • Genre rules inform activities that simultaneously reproduce these structures.	Duality of Structure • Project planning, roles, and responsibilities match procedures established in the documentation. • The documented rules guide procedural and organizational decision-making.
Community Ownership • Genre conventions signal a discourse community's norms, epistemology, ideology, and social ontology.	Community Ownership • Philosophy of development methodology expressed through documentation and practice.

Interview Subject Profile

The seven professional developers have a cumulative 74 years of experience. They are each senior developers and many of them have worked together as a team for many years; two of them have worked together for nine years. I asked the developers to identify their own alias during the interview; the following list presents the profile of each developer—by alias. In addition, I asked the developers to share their key development philosophy; they work well together so it is no wonder that their philosophies follow a single theme.

Bob. Senior Developer. 12 years experience.

Interview Duration: 56 Minutes

Development Philosophy: Code must be easy to read and maintain. Must be simple enough that it doesn't need a document.

Joe. Senior Database Developer. 13 years experience.

Interview Duration: 43 Minutes

Development Philosophy: The reason for a report must be clearly understood.

Rebecca. Web Subject Matter Expert. 3 years experience.

Interview Duration: 51 Minutes

Development Philosophy: Build it right, even if it takes longer. Build what is needed—not what is desired.

Leaf. Senior Database Developer. 11 years experience.

Interview Duration: 32 Minutes

Development Philosophy: Get it out—quick and quality code—don't overcomplicate.

Judy. Senior Database Developer. 7 years experience.

Interview Duration: 36 Minutes

Development Philosophy: Flexibility. Optimization. Usability.

Hudson. Senior Software Engineer. 12 years experience.

Interview Duration: 44 Minutes

Development Philosophy: KIS method (Keep It Simple). There were too many times getting bit by overcomplicating the solution.

Brad. Senior Software Developer. 16 years experience.

Interview Duration: 34 Minutes

Development Philosophy: KIS method (Keep It Simple). Agile is only doing what we clearly know at the moment.

Interview Questions

The interview questions were each taken from a respective expectation in my Model of Expectations. With that 1:1 relationship, I was able to connect the reflections of the subject matter experts directly to my Model of Expectations. In addition to the 1:1 questions, I started with icebreaker questions. I wanted my subjects to have the opportunity to easily feel their strength as a subject matter expert. The icebreakers were philosophical by nature and set the mood for a contemplative interview experience, from the start. I expected that when I asked them about documentation that they would confess their poor documentation practices and explain what they do instead. However, the interviewees did not often explain what they would do instead; most often, they simply spoke about what the activity of "documentation" ought to be. Bob, Brent, and Hudson were the three that were the most contemporary with their documentation and therefore had the most useful things to say about rhetorical forms, outside of written communication. Table 10 showcases questions matched to each of the 11 expectations in the Model of Expectations.

Table 10

The Interview Questions Organized According to Genre Principles.

Interview Questions
Ice Breakers
1. How long have you been operating? Can you explain how you are agile and why you choose agile methods?
2. What is your development philosophy? What development standards do you value most?
3. What does a typical day look like for your own role and responsibilities?
4. If it is not documented it didn't happen. The absence of documentation is a failure to communicate. What kind of response do you have for these two statements?
Dynamism
5. I'm looking at these two documents. Describe how they fit into your workflow--both writing and using them.
6. Describe a situation in which you needed the documentation to resolve the team's confusion about the design.
7. Some professionals refer to live documents or organic documents. Describe a situation in which an organic document a) evolved with the development cycle and b) informed the development cycle?
Situatedness
8. What changes when design concepts are written down, rather than merely "known" by the team.
9. Industry writers suggest documentation is a bible or blueprint or governing document. Vision documents. Guiding documents. What kind of governance does documentation have in your shop?
10. Aggregation. Compounding. Synthesis. These are all words that suggest a whole is formed by the sum of its parts. Can you explain how a document is the sum of development activities?
Form and Content
11. Books have recommended outlines and there are templates available online. What kind of adaptations do you make when you measure your work against industry samples? Describe an experience when the recommendations didn't fit right.
12. In addition to standards, there are other things that don't fit right. You make decisions about direction, design, procedure, and operational details that don't always fit right. I'm looking at this document sample; can you tell me some tough decisions that involved this document?
Duality of Structure
13. I've observed that your workplace is organized to meet specific needs. In what way did you use documentation to identify those needs and record your business solutions? Can you tell me how well the documented business solution works for everyone else in the company?
14. How much do you draw on documented business solutions when you have a meeting? Can you describe what that would/should look like in a perfect world?
Community Ownership
15. You told me about your development philosophy. Now that we have discussed documentation so much I wonder how you connect your documentation to your philosophy.

Most Surprising Interview Question

My biggest surprise was the fourth Ice Breaker question. I used two popular idioms from professionals that advocate traditional documentation standards. I expected my subjects would be quick to admit problems with documentation but quickly recover as the subject matter experts who know how things really work. I did not expect the subjects to disagree with the first idiom, while also agreeing with the second. They felt that "If it is not documented it didn't happen" degraded their work, insofar as they do not document. Brad's response to the question was simply: "It did [happen] though." On the other hand, Bob disagreed because the computer code is the best documentation.

Even while they disagreed with the traditional implications of no documentation, they still agreed they were a failure of communication. However, what they meant by agreeing was not what I originally meant by the questions. When they explained their agreement it was always insofar as the documentation left for those that come after. They failed to communicate to other developers or they failed to leave the appropriate details so they could pick up where they left off. What they did not say was that they failed to communicate while they were working together. That underscores the importance of the degradation of the first idiom; they were contemporary communicators who met their common software philosophies; they were agile, responsive, accurate, and productive.

Brad puts the seeming contradiction in perspective:

> "Failure to document is failure to communicate" is only true in a world where no one can communicate outside of written forms. There are all kinds of ways to communicate. Meetings where I show [the customer] how it works. Walk over and tell me—whatever you just did it sucks make

> it better. No [developer or customer] knows up front what they want. There is never a situation where someone has crystal clear vision of building something; even if there were [such a situation], a developer would not know how exactly to build something. Sometimes the effort to document is a failure of communication.

Brad argues his team communicates really well and that documentation is not as important as the efficient communication that gets the job done right.

RESULTS: MODEL OF EXPECTATIONS

The Model of Expectations is well adapted for describing postmortems with a meta-language and is the foundation for the interview questions. I translated each expectation into a more industry-based expectation of traditional documentation. Those practical descriptions are worded for practitioners and faithful to the documentation rules recorded in industry guides. Each interview question directs the developers to those "Worded for Practitioners" expectations; however, the developers' answers do not match the expectations because they do not use traditional documentation. Consequently, I use the meta-language of genre theory to describe the rhetorical forms the developers use.

I have 4 hours and 55 minutes of interview recordings. The subjects I selected carefully articulated 11 exciting answers, based on their senior-level industry experience. I wish I could simply use a transcription of the interviews for the chapter and let the developers speak for themselves and their profession. Unfortunately, I can only sum up their positions and highlight some of the most significant responses. When I highlight those significant responses, I will provide extensive block quoting to let the subject matter expert explain the rhetorical form for me.

Unfortunately, answers to these questions were not definitive about either written or oral communication. There were all kinds of rhetorical forms—use cases, white boarding, index cards, meeting notes, etc. The mistake I made was to let the developers answer my questions with how documentation ought to work in such situations; I think I should have challenged them with a single, game stopping exception: "But you don't document." They would have responded more along the lines of my expectations; they would have talked specifically about the other rhetorical forms they use. However, the answers I did receive do in fact point to the value of oral communication; yet, they also point to the ongoing importance of written communication and the large array of other rhetorical forms they use during "documentation" activities.

Results of Dynamism

Actors' responses to recurrent situations. Interview Question: "Describe how [documents] fit into your workflow--both writing and using them."

For the most part, the seven developers tried hard to write traditional documents. Often, there documentation involved groups outside their team. In those cases, an outside group's documentation submission would be requirements that may not have met the needs of the recursive situation. The developers would seek out and engage the outside group to build the document that meets the needs. However, Bob and Hudson described more contemporary experiences with documentation that identify different rhetorical forms—oral communication, whiteboards, and diagrams.

Joe suggests a good document requires less follow up; Joe builds business intelligence reports for colleagues and must follow up on poorly documented requirements. Judy depends on the requirement documents submitted by colleagues; she expects a specific format and specific information; she always reads the documentation before she begins development so that she knows she has all the information she needs. Rebecca documents release notes every time she rolls out web tools or tool enhancements; however, there are some tools in which the audience is so close to the actual development that she does not document the release. Much like Rebecca, Leaf also documents release notes. However, Leaf works on a system that impacts much more of the organization so Leaf must review communication documents from other teams and release his own notes as well. Brad writes guides, charts, and diagrams for other developers; he does not typically write rationale, unless the "normal way" did not work.

Bob said contemporary methods seem to require less documentation because the team works so closely with so many iterations—suggesting strong oral communication.

> I think we tend to have…whether agile has pushed this way or not I'm not very sure because I didn't do a lot of waterfall type design or documentation in the past. I see that we tend to require less documentation to convey ideas between developers because we are working so close and such fast iterations and on the same code at the same time that there almost seems to be less of a need for documentation.

Hudson remarked that he worked both traditional and contemporary development methods; he suggested written documentation is weak in both methodologies.

> When I was doing waterfall companies would say they were waterfall but they wouldn't give you the time or resources to do waterfall. So they would force you to pretend waterfall…The thing about agile is it is an honest contract. You and the business owner agree to build things iteratively and to rapidly prototype and to have versions and revisions and iterations. Push stuff out roughly and revisit and change it over time. In the same amount of time it took to flush that out in the waterfall model you are further along, better product, more quality, using the agile

> methodologies…documenting the process documenting the software documenting the roll out the access the security none of that is important enough to stop development on the next thing.

Stabilize experience and give it coherence and meaning Interview Question: "Describe a situation in which you needed the documentation to resolve the team's confusion about the design."

Bob described the problem with both stabilization and meaning-making in his own practices—the usefulness of written communication and how it should be done:

> Here is how we should architect the app. There is a piece over here that does blah and a piece over her that does blah and a piece over here that needs to talk to these two. And you go through this discussion and everyone in the room is in [agreement]…where you start having problems is when someone is building their piece and you start building your piece and you can't remember how they are supposed to interact. And what you end up with is two pieces that don't work together.

Leaf and Rebecca both related different situations in which the absence of a document seriously impacted the stabilization of the community. There were numerous high management stakeholders, competing interests, and no documentation to unify everyone. Leaf remarked on the levels of confusion, as well as the frequency in which components were built incorrectly. Rebecca expressed frustration that the project was actually six projects merged together in one single nebulous intersection. The overlap was not defined and destabilized expectations. Further still, Rebecca described how departments did not work well because in the absence of a defined, documented intersection, managers would not agree on prioritization; they all felt their needs were priority one in a project where each group was left to conduct meaning-making on their own. In contrast to Bob's example where oral communication was enough, Leaf and Rebecca both describe situations in which the community of agents is too diverse to warrant anything but an official, written document.

Brad also described a situation that highlighted written communication; in this case he did write documentation. The code he developed at the time deviated from the expected deliverable and clearly brought coherence and meaning to the situation: "Documentation actually helped on our current project. The workflow was itself a hack to get around the implementations so documenting if, when and where brought a lot of clarity to everyone in the process."

Response to users' sociocognitive needs. Interview Question: "Some professionals refer to live documents or organic documents. Describe a situation in which an organic document a) evolved with the development cycle and b) informed the development cycle?"

The developers clearly outlined the need for documentation but did not identify written rhetorical forms as their particular sociocognitive need. In fact, Bob frequently abandons written documents.

> Good intentions—when you make changes log it. But when you run against timeline or you forget [it is] difficult to be accurate and make it worthwhile. Where we have failed with live documentation is we did not have someone with us who had the primary goal was to keep the documentation up. Usually left to developers who are too interested in the

> code to make it worth while…It needs to be iteratively at the same time…you have to be this all encompassing team that all have their fingers in the stew and working on it a little bit at a time.

Bob's sociocognitive need is either an deeply immersed writer or a team that relies on an organic document that morphs to community knowledge.

Hudson declared that organic documentation is the only way but admits there is only one organic document the team maintains:

> The organic document we have is the one that talks about our coding policies and coding procedures. That document has been a living document that we keep that helps us all develop in a way that we can read and understand. But I think that living documents is the only way to go. A living document is kind of a necessity in a fast paced environment…there is no procedure to update it. In the past we periodically come back together as a team and say we should get our standards together.

Results of Situatedness

Participation in the communicative activities. Interview Question: "What changes when design concepts are written down, rather than merely "known" by the team."

The developers did not all take a clearly traditional position. Bob, Joe, Leaf, and Judy each agreed that a written document was the best way to transmit information to the team and retain the team's situated knowledge. Rebecca complained about all the phone calls when things are just "known"; she also pointed out that agreement in meetings was difficult because no one could remember previous decisions in the same way. Hudson suggested that reliance on undocumented "knowledge" encouraged knowledge experts to silo themselves.

Brad strongly opposed that documentation was the source of situated knowledge. Brad felt a healthy, traditional document interfered with good communication.

> Pro side: greater consistency, people are building the same vision. On the con-side, if anything causes that vision the need to change. You end up getting more confusion. Because some people will be working off known data and some people will be working off current documentation. And some people will be working off old documentation. So the challenge is to recognize when to either [abandon] a document or how to [operate] changes to everybody. In case of the giant waterfall project so many people made the changes in so many places so often across so much of the project. That keeping the document version was impossible. And even if every update had been made to the document. You basically would have re-read a novel multiple times to keep up with the project. The document was so large it was unusable. It got to the point where it got so cumbersome that we wouldn't even work off of the document. We would just go back and tell them so it would get added to the document. Which they would diligently do. But then no one would read what we'd added to it, because no one was working off of it. So in the end the document was schizophrenic use of words. It was pretty interesting. If you wrote five

> chapters of a book being written by five different authors, and those authors weren't allowed to talk to each other. It was literally 1000 pages.

A form of situated cognition. Interview Question: "Industry writers suggest documentation is a bible or blueprint or governing document. Vision documents. Guiding documents. What kind of governance does documentation have in your shop?"

Bob relies on diagrams, mockups, and screenshots. Joe obtains documents but if the colleague deviates he insists on elaboration so he only needs to code one time based on precise requirements. Rebecca does not think docs are governing docs; they are clarifying docs. She does not want to be stagnant just because some 16-page document governs the only way to do something; a document should clarify requirements and it is up to her how she does it. Leaf prefers to think of documentation as more of a blueprint than a guiding document; Leaf wants overall rules but no dictation.

Judy describes documentation as sort of sharing but it is not a contract either.

> It's a knowledge sharing, and it's a key to enter to something that you need to involve but you are not aware of. It can save you a lot of time. Documentation is important and I will always request documentation if possible. But I also understand it's not a contract. Meaning that you have to there are some errors in the documentation—not 100 percent accurate.

Hudson admits he has not experienced a governing document that creates a situated knowledge. To illustrate the document with which he has experience, he described a long two-hour meeting in which he developed the user stories. He speculated that a collection of user stories might be acceptable for the "documentation" activity.

> I think that would be something more like a user story…the user story conflicts with something [or] the user story is the blueprint and the list of user stories then could be an overall vision document but again that is something that happens so rarely in our development. Often, we have to assume the user story or know it; most of the time we don't record our user stories in documents. There may be a rough lists in a project request. I had a two-hour meeting where they basically spit out all their requirements verbally and maybe someone takes notes.

Brad does not want the documentation to rule his life because documentation does not rule his colleagues that identify their requirements. He stated that such colleagues are not governed by the document when they change requirements halfway along the project timeline.

> Only if the requirements folks are willing to obey too. That's the problem. It's always one sided. The people on the deciding end of things think they can make whatever changes whenever they want and they just gotta change the document. And we're expected to live with that. But we can't make any changes to it. Doesn't work that way. Go to an actual architect contract them to build a skyscraper. And after they've got to the 26th floor of 52 floors. Just go ahead to tell them you want to move where the elevator would be located. And see what answer you will get. The answer is no. Absolutely not! If it comes to me quitting—fine. But I'm not moving the elevators. But software is all virtual so nobody holds to that. So Documents can't rule my life. Because they don't' rule the other side.

> Fortunately because we're virtual we can do it. Often. But there has to be a corresponding conversation. Timelines, effort, why are we moving it. Is it worth it? And that will ultimately change whatever you document. So the documentation would have to adapt if were trying to keep pace….Documentation is so large and cumbersome, it started being a deposit of information without anyone ever trying to withdraw information from it, and document not being read is not a lot of value. There are reasons in there, it was just too big and too irrelevant to any one individual's work task. It's got to stay relevant.

Participate in the activities of the ambient culture. Interview Question: "Aggregation. Compounding. Synthesis. These are all words that suggest a whole is formed by the sum of its parts. Can you explain how a document is the sum of development activities?"

Bob suggests that the documents show the whole picture (what systems talk to what systems) so when a developer sees how data is exposed in another part of the system that necessarily changes the way the developer acts.

> Documentation in the form of mock-ups [aren't the sum of development activities]. But documentation in the form of UML tag around and that kind of thing. I think it helps solidify that because while you may be working on this one piece over here that's just the interaction with the data base. When you see the whole picture. When you see the entire system diagramed out. You really get to see what systems were talking about. Systems and a lot of those things will influence how you develop your

> piece. If you know that your data is eventually going to be exposed over here then you may develop the interactions with that data differently. So I guess that's backwards from what I'm trying to explain. Every little piece in that system by itself should be …if it's well factored code and that kind of stuff…that data library should be able to stand on it's own and be used by multiple things. How it fits into that big picture is really important to how you develop. So you take all these little pieces. Then that big picture shows you what you built.

Brad expressed concern about a document that is an accumulation of a team's project knowledge. He cautions against the Frankenstein monster; he states that some parts are not needed for some roles—do not need to see those parts. I edit out the proprietary information with the following convention: "[…]".

> Yah, the whole is formed from the sum of it's parts. That [question] has to issue a warning that so too was Frankenstein. So in order for it to be the good sum of it's parts it should document as little as possible only what's relevant and [eventually] detailed decision making that doesn't affect the processes as a whole to the decision makers. And if they require a separate document they would just reference it. So for instance on a web site: I may develop a […] for my own use for the database. But the web designer doesn't care doesn't needs to see it. They can ask for it if they want. But they shouldn't be involved in that document. And so if you have the single responsibility philosophy around the documents that would help. And then on the same note, as a […] developer I could care less

> what a CSS file says. Literally matters to me not one bit. (CSS document the cascading style sheet)

Results of Form and Content

Embraces both form and content. Interview Question: "Books have recommended outlines and there are templates available online. What kind of adaptations do you make when you measure your work against industry samples? Describe an experience when the recommendations didn't fit right."

Bob has searched the web forums for solutions but (assuming the online forum answer is a kind of documentation) often the solution is the solution for a specific context and does not fit into another context. Joe told a story of when he built an online help file; he planned to base it off of Microsoft's help file system but decided to adopt a "purely screenshot vision" of the help file, where the screen is presented and people can click on the parts that interest them to get context-sensitive help. In addition, Joe described an unimplemented user demand with which he could adapt the form and content yet again.

> The type of documentation that I've done in the past is like programming standards. Also I did mention I did build a help file for that one system. And I used, and looked at Microsoft's help file, and I structured my help file according to that standard…[but] instead of having them read through a bunch of text I put a picture of my screen on the image and let them hyper click. Click it on the screen Click the button and it pops out a message, Saying what it's doing. So it's more of a visual…The problem is the help file worked so well, that sometimes people would go to the help

> file, and be like "I can't enter it in." If I would have had time I would have made it a little bit more noticeable that this is documentation as opposed to the actual application itself. So you'd know which they were in because it was so much alike.

Rebecca was once given a document that detailed every feature of three different sites so that one single new site could do everything the other three sites did; this distressed her at the time because the consequent documentation did not highlight any distinguishing details.

Hudson does not want circus—lots of content that should be separated or deleted; he will "grow down or grow up" the document, based on what he needs. Brad advocates very specialized documents; Brad fills out general documents if he is required but general documentation does not add lot of value to his process. The documents that he finds helpful are the ones that are specific to some very separate piece of detailed information.

Appropriate content for a particular purpose, situation, and time. Interview Question: "In addition to standards, there are other things that do not fit right. You make decisions about direction, design, procedure, and operational details that do not always fit right. I'm looking at this document sample; can you tell me some tough decisions that involved this document?"

Bob describes being empowered to change documentation a little; if he did not understand it or it did not feel right they did it the best way they could. Leaf was pretty clear about his position—just change the document.

Brad shared a story demonstrating that the document could not have kept pace with decisions on a project.

> The actual project had to be re-engineered 6 times and it was all because we had made assumptions about minimum functionality on an underlying system and it all proved false. The system was far less functional than we ever expected. So the hard part is that if we had documented that project—which we didn't—but if we had we likely would have said, "well we're going to build this so that we can send this to that." So what would happen is because of the fact of the end destination of that couldn't handle the traffic we had to add multiple interceding pieces to manage the slow flow or feed that data into the underlying systems. You know that initial design document would not have captured what it looks like at the end and it would have been updated to keep pace with it. Really the document would not have driven most decisions.

Results of Duality of Structure

Inform activities that constitute social structures. Interview Question: "I've observed that your workplace is organized to meet specific needs. In what way did you use documentation to identify those needs and record your business solutions? Can you tell me how well the documented business solution works for everyone else in the company?"

Bob states they are lacking on this and are working on it; a document would head off questions to the developers; Bob highlights they are happy to help but they do not remember all the time.

Joe uses email and the knowledgebase system as his documentation; these methods let the users write documentation.

Rebecca does not feel like she impacts the structure because while she consistently issues release notes, she knows that no one reads her release notes. Brad relates a similar problem; very few of his documents broadcast outside.

Hudson spent several hours documenting recently and they had a great document up front; however, smaller projects involve just telling or email.

> We spent several hours in several meetings on what we were going to build and how we were going to build it, with several people in the room. [One colleague] recorded what we wrote and said and drew on the boards and we have now a real nice document. Which is a really good guide to what we should develop and how it should work. We spent that time up front on that project to do it because it was such a big complicated important project for the company. That would never happen on smaller projects that may be just as impactive. That worked out pretty well. But it was kind of a throw-back towards waterfall development. (To record business decisions on smaller projects is different.) Mainly it's by telling through user testing and acceptance and just telling them how to use the new system the new process. And email those out to them, that sort of thing. There's no documentation. There's no formal way to describe a process there.

Simultaneously reproduce these structures. Interview Question: "How much do you draw on documented business solutions when you have a meeting? Can you describe what that would/should look like in a perfect world?"

Bob identifies that documentation is very important for meetings because the group has already discussed and the documentation is the findings of that discussion—the Kamban board is their record of previous meetings. Bob argues that if anything were remembered incorrectly then anything developed from that is flawed. Leaf agrees with Bob because he states the document is the baseline for each meeting; without that baseline it is like starting from scratch.

Joe wants to walk through a screenshot during a meeting and Rebecca likes how one of the company's business analysts takes minutes that are project up on the wall. Judy looks for email because she works with a lot of legacy, potentially obsolete, documentation.

Hudson says they do not have the utopian documentation; they have little "blurbs" and lists of requirements.

> This sounds like that would be a utopia right. That would be great if we could do that. I've never had a document like that. That describes it too much. We do have blurbs which could be sort of. So in our IT projects request system. We for this element system recorded a list of requirements so we used that list of requirements as clocking points in several meetings. Where we'd look at them and say this is what the requirements are and that helped us to solve the problem, build the system, build pieces of the system to support the requirements there. So it was a

> glimpse of what it could be. Because even in agile it would be good to have documented user stories and a list of requirements at the beginning of a project. Even the smallest project. So I think the big thing that I notice that when we have those we get to the solution quicker than if we do not.

Brad states that living documents have no governance. He confirms that simply by rewriting or updating a document, the document has no governance. On the other hand, Brad suggests that even if the document did govern, the stakeholders would just scratch out X and write in Y, even if the documentation said X.

> Back then [when I used traditional Waterfall methods] sure they'd always bring the document. And we'd point out they were four versions behind and they'd point out that the newest version even if they were worried about it didn't include what they were talking about. And the few times we were able to say the documentation clearly states this, they were able to say, "well let me change the documentation". So the fact that it was written down was meaningless in the face of new requirements. [At my current employer] we'd all write it down but I think that it's still true. If the business came to us and said well we needed to do X now and we said "but you wrote down Y" then they would say "well, here hand me that" scratch scratch scratch, do X. There! Happy? So we don't even bother to write it down. See. That's the thing about living documents; it's great if you ever would want to refer to them later. Which isn't usually done. Because if their living, then there not. They have no governance.

Community Ownership and Results

***Norms, epistemology, ideology, and social ontology.*.**Interview Question: "You told me about your development philosophy. Now that we have discussed documentation so much I wonder how you connect your documentation to your philosophy."

Unfortunately, the developers interpreted this question as an opportunity to draw a conclusion so that I did not always feel they were helping me understand community ownership.

Bob's documentation is in his code; however, code is so granular that it should not need a code comment and the less granular the more documentation is required. Brad is very similar because he talks about documents that only fulfill one highly isolated, detailed, finite purpose; he likes his documentation to be specific and as granular as necessary.

Rebecca says documentation is important for disconnected departments because the documentation enables communication for people not in the same room. For instance, both Joe and Leaf are remote employees; both Joe and Leaf take requirements from colleagues so they want fully detailed documents and do what they can to obtain full documents

> With disconnected departments or people, either by land or by time, documentation is extremely beneficial in the process because customer involvement and feedback is crucial in agile development. Documentation enables communication between people that are in the room at the same time or people that are dislocated. For example, I don't want to have to call people every 10 minutes. So I have to take notes when I'm on the

> phone with [with a colleague]. To note how I'm going to do something. Versus Derek I don't usually take notes because he's just right there. If I have a question I just turn around and say "Hey [colleague], did we think about this." So that's where Documentation is particularly beneficial.

Hudson's KIS applies to code and should apply to his documentation. He wants simple docs that tie into his code 100%. He does not want "circus"; he does not want to jump through hoops and follow all kinds of rules.

> Ya if I was to be able to make all the decisions on what amount of documentation and how we documented in our process. Those philosophies would tie into it 100%. So the "keep it simple". I would apply that to code. I would also apply that to my documentation. I would make the documentation useful, to the point, but simple. Right.
> So…some of these rules of something like that as far as you have to do these 12 steps and all the circus in my documentation. I'd keep it simple in my documentation. Something I coined I hate circus I don't like jumping through hoops. Or pretending just for somebody's ego. Our boss says you need to do this. It doesn't help the process. Right? So you have to go move the card on a board, you have to tell me an estimate even though we know that estimates don't work. Cause were not up front documenting, were not upfront estimating. An architect our stuff, we're very agile. We jump in, we spit something out, we reiterate. We spit something out again—we reiterate. That comes very contrary to "tell me when it's going

to be ready" philosophy, but we have people above us who are pretty old school that don't understand that or don't want to accept that. So they still ask for deadlines.

INTERVIEWS WITH THE EUPARS FRAMEWORK

David E. Hailey created the EUPARS model to identify the appropriateness of text for various sites. While Hailey (2013) uses the word "text," in his book, I substitute "rhetorical forms" so I can stay consistent with the rest of this dissertation. Yet, Hailey goes to great lengths to define "text" very loosely in his book. Hailey believes everything is a text because, "any pattern that can be separated from other patterns is by definition text" (unpublished, p. 70)—much like Derrida and Roland Barthes. For instance, Barthes (1975) pushes text further than "words on a page" by distinguishing work from text: "a 'work' is a closed, finite product of traditional canonical literature; a 'text' is an open process with which one can interact creatively" (p. 517). In this way, a traditional text is actually a work, whereas Barthe's text is an interactive object that draws on the creative attention of an audience. In other words, text is not a single dormant unit; text can be a host of rhetorical forms.

Hailey argues that not all rhetorical forms belong in all genres. For instance, he demonstrates his point with a website that places the company's "About the Company" statement on the front page. According to the EUPARS assessment model, the front page is an inappropriate place for that specific text. As reason, Hailey suggests the rhetorical stance and the structure are inappropriate for a front page. After all, the visitor does not want to read through an “About the Company” statement; rather, the visitor wants to see content. In addition, the structure is poor because an “About the Company” text is written with a narrative style. The visitor does not likely wish to examine dense text; rather, front page visitors want to see bullets, numbered lists and chunked information.

Table 11

Hailey’s Demonstration of His EUPARs model.

	Evaluation	**Appropriate? If not, why not?**
Exigency	Someone in a decision-making position feared that potential customers coming to the page would not know who the company is and how inexpensive their products are.	**YES**
Urgency	It is probably important that *some* information be here.	**YES**
Purpose	Immediately inform the user.	**YES**
Audience	They accurately describe their audiences in the paragraph: various portions of the public and construction community interested in solar power. The audience will range between people completely uninformed on the subject (wanting to learn about it) to people with certification on the subject and simply wanting a less expensive resource.	**YES**
Rhetoric	The rhetorical breakdown is subtle. The copy is clearly designed to persuade the audience that Wholesale Solar is a good place to buy solar products, and it assumes that audience is open to persuasion. The audience comes to the page and will find it useful to immediately know they are in a right place, as it is useful for the company to be able to pitch itself to the audience.	**NO**: The audience will need to *immediately* know where they are. The paragraphs do not meet that need. Instead, they force the audience to read unnecessarily dense blocks of text.
Appropriate	Narrative	**NO:** The appropriate structure for this kind of content is bulleted or numbered lists.

The above table presents the assessment data for the “About my Company” rhetorical form. The rhetorical form meets the requirements of several EUPARS terms; however, Hailey describes his evaluation of what kind of structure the rhetorical form has and then describes why the rhetorical form is appropriate or not.

The Interviews Presented in the UEPARS Table

The following is my own evaluation of the rhetorical form I have overlooked with genre theory—oral communication. According to the developers I interviewed, while oral communication is a new rhetorical form, it is not the only new rhetorical form. As Hailey (2013) suggests, a web page is the location of several genres. Similarly, Clay Spinuzzi (Spinuzzi, 2003) suggests genres make up a genre ecology in which one single rhetorical form connects to all the others. In similar way, software documentation is several rhetorical forms--pieces of simultaneous communication. However, while oral communication is not appropriate for contemporary developers, neither is traditional documentation. In the end, the developers used multiple rhetorical forms.

Table 12

Application of EUPARs to the Lunchtime Interviews.

	Evaluation	Appropriate?
Exigency	They don't have resources or business buy-in to deliver as many documented deliverables as undocumented deliverables so they need faster ways to communicate.	Yes
Urgency	Business wants solutions now. The developers need to communicate now. Takes longer to document than it does to develop sometimes.	Yes
Audience	Talk to each other. Whoever needs to be involved can be involved; if you are not involved then do not engage. But they admit failing transmitting outside group, whether written or not	No. Inappropriate if it replaces written rhetorical forms. But is appropriate if it is one among an ecology of rhetorical forms.
Purpose	Can't replace documents but oral communication is fast.	Yes
Rhetorical Stance	Team is persuaded and it works for developers great. outside don't actually read it so how can they be disappointed at the lack of transmission.	No. Written is inappropriate and so is oral.
Structure	Written is inappropriate. They talk and collaborate in tight cycles and close proximity. However, their remote colleagues still require written documents.	Yes.

The Interviews and Appropriate EUPARS

Oral communication seems appropriate for the exigency, urgency, audience, and structure of contemporary developers. In other words, oral communication seems to do a better job than traditional documentation for these senior developers because it meets the need of contemporary developers who need quick and agile responsiveness to business needs. Much like the student interns of the Engineering Modules project, the lunchtime interview participants did not write documentation, as recommended by the software industry. They talk a great deal at their cubicles, in five-minute "stand up" meetings and at the white board. They had artifacts for many of those oral communications (temporary content on a white board, diagrams, user stories, etc.) but their reliance on their six-foot proximity was clear. They move between cubicles, talk over cubicle walls, share screens, and stand together talking between all their cubicle desks. During the interviews, they described how documentation is supposed to look but they were quite clear that they did not work that way.

Joe and Leaf expend lots of energy making sure they have all their traditional documentation; yet, they are the remote colleagues on the team and may necessarily need written communication. Judy is in the middle; she wants traditional documentation, does not like working with outdated legacy documentation and recognizes that the business will likely change before she can develop the alteration—never mind document the alteration too. Brad is on the other side of the spectrum embracing oral communication and thinking beyond the constraints of traditional documentation. In fact, Brad relayed the story about 1000-page document; he said the company frequently distributed printed volumes but the subsequent software "sucked."

The Interviews and Inappropriate EUPARS

On the other hand, oral communication is not appropriate for either the audience or the rhetorical stance. Simply put, traditional documentation might suit the audience and rhetorical stance the best. Yet, that is not a game changer because the developers were always quick to (1) identify they did not actually document in that way and (2) outline the flaws in that kind of documentation. Most importantly, they mentioned the array of rhetorical forms they used to substitute the clunky documents they did not write.

Inappropriate for the audience. In most cases, oral communication worked well but the audience was not limited to only the development team. In other words, if the team was the only audience then oral communication worked well for them. However, the documentation they did write was used by different departments in the company so that the audience was not simply the developers. Consequently, oral communication was not the best way to transmit information outside the small, tightly unified, contemporary development team; they did not share cubicle walls with every department.

Inappropriate for the rhetorical stance. Traditional documentation assumes that large groups need a center of meaning from which they can find unity and direction. By their own admission, the developers each relayed reservations that oral communication creates either unity or clear direction. Yet, the developers were also clear that traditional documentation does not actually unify or direct either. For instance, Brad adamantly stated that the effort to document was often a failure of communication in and of itself. Hudson was frank about the weaknesses of traditional documentation. He remarked that even his experience with traditional development workplaces did not have the resources to document according to best practice.

The conclusion here is a different kind of rhetorical stance evidenced in the interviews. The developers extended their agile, contemporary values to "documentation" so that they employed many rhetorical forms to transmit the center of meaning to relevant stakeholders. They did not simply write or talk; they scribbled on white boards, tacked index cards on the wall, and traced out diagrams. Their transmission strategy was pieces of communication like the sticky notes and meta data files I gathered during the Engineering Modules project.

DISCUSSION OF RESULTS

There is more to contemporary "documentation" activities than simply written documentation. Documentation also includes an array of rhetorical forms. However, as long as developers and researchers speak the language of written documentation, those additional rhetorical forms remain outside detection. By breaking the recursive activities into purpose, situation, and community, I have highlighted both the oral communication and pieces of communication that constitute the rhetorical forms of contemporary developers.

Shifts to contemporary situation, community, purpose. The Situation of development shifted from traditional to contemporary practices to accommodate smaller, short-term projects. The contemporary practices also acknowledged that users had a stronger voice that could improve the quality of software production.

The Community of developers adjusted their mindsets to value the speed and flexibility they needed to meet development schedules and client needs. The community altered decision-making processes to support projects that changed so quickly; decision-making needed on-the-spot spontaneity, rather than ponderous decision-making policies.

The Purpose of documentation activities had to change with the development practices, even if contemporary developers still tried to make traditional documents work. The simple fact was that traditional, preplanned documents served a different situation and community. Contemporary developers need documentation that both supports and matches the speed of development. They need a record of the most recent decisions, without spending half the day updating the written documentation every time.

Contemporary developers need rhetorical forms like oral communication. Even a snapshot of action items listed on the whiteboard is a rhetorical form that supports development and records key decisions.

Answers to the research problem. I wanted to know why developers were not documenting like I thought they should be. I knew industry recommendations for documentation and I knew how genre theory described the documentation I expected. However, contemporary developers simply did not document as I expected or did not document at all. I set out to answer three research questions:

1. What do they use instead?
2. Are current approaches appropriate?
3. If they are not appropriate, what should the developers be doing?

Instead of written documentation, developers use a variety of rhetorical forms. According to the EUPARS model, those pieces of communication are the appropriate rhetorical forms for the purpose, situation, and community.

The Impact Rhetorical Forms on Genre Theory Research

Rhetorical forms. Both professional communication researchers and genre theory researchers seem to focus on written communication. Researchers of written communication naturally look for written communication when they observe written communication practices. This fixation limits the predictive power of genre theory in cases like contemporary software development when written communication is not the rhetorical form on which they rely. With the addition of a rhetorical form like oral communication, genre theory researchers have a complete meta-language to go beyond the language of written communication and describe rhetorical forms with a meta-language like genre theory.

Verbal precision. Researchers need more precise terminology than verbal and non-verbal text—particularly because neither really references speech acts. Even in cases when researchers refer to verbal communication, they typically suggest written communication that has words—as opposed to the wordless non-verbal communication (i.e. illustrations). The primary implication is that verbal communication should be reserved for speech acts alone and more descriptive terminology should account for other rhetorical forms.

Genre ecology of oral communication. Oral communication is a blanket term. There are many genres of oral communication employed by public speakers, actors, and singers. In fact, when my family members having kitchen table arguments about politics they are deploying a genre of oral communication. Developers utilize a large range of oral communication genres. Much like Clay Spinuzzi's genre ecologies, researchers need to understand the web of genres that sustain contemporary development communities.

The Impact of Rhetorical Forms on Software Development

Oral communication as a value. While contemporary development methods are not brand new, they are not mature practices that have entirely replaced traditional methods—there are plenty of traditional developers still. For instance, government developers often use traditional methods for the federal software systems that services thousands of users. Consequently, this dissertation supports oral communication as a supportable value in contemporary development methodologies, even if traditional developers do not count oral communication as a "documentation" activity. After all, when traditional developers even have cycles, those cycles are measured in months, and formalized documentation is a necessary solution for something everyone on the team likely forgot months ago. However, contemporary developers measure their cycles in days and their oral communication carries their tightly coupled iterations.

Developer guilt. When developers meet me, they often confess their documentation weaknesses. After all, I am the software documentation researcher and I would know just how bad they really are. To a certain extent that is true; at the same time, as a software documentation researcher, I have wondered at the distinction between the documentation I expect and the documentation I actually find. Developers should stop feeling so guilty about weak documentation practices because they judge their contemporary practice by traditional values. The addition of new rhetorical forms means the development of more community ownership in areas they had not yet seen; after all, they were very busy trying to write the traditional documents they knew they should write.

Pieces of communication. Developers surround themselves with communication that they do not recognize—rhetorical forms not acknowledged by traditional, industry authorities. Consequently, they have a lost rhetorical opportunity to build a strong communication practice because they do not acknowledge the pieces of communication in their development ecosystem. Once they see their pieces of communication, they can formulate a plan to transmit meaning through those rhetorical forms that clearly work for the team. They can develop community ownership for those pieces of communication and develop dynamic ways to formalize them as they adapt their form and content again and again.

Natural fit of documentation. Finally, developers should stop trying to fit traditional documentation into contemporary development practices. Without a meta-language, I was unable to separate documentation from the "documentation activities" too. However, the meta-language outlines the pieces of communication that stabilize purpose, situation, and community. Contemporary developers have oral communication, along with other rhetorical forms, with which they already have a natural fit.

Opportunities for Ongoing Research

Three future adaptations of a similar study.

- Use one specific document to which all interviewees can refer. While I planned questions that would refer to a specific document, the lunchtime interviews were not conducive to the acquisition of samples. However, future work would need to secure common documentation samples on which various questions are based—that way the interviews control for that variable.
- Contemporary development is an enormous umbrella for a very diverse range of methods. There are many specific titles with very specific differences. Those differences impact the community ownership of developers because their choice of methodology is a choice that reflects customized development principles. In short, Scrum, Agile, Kanban, Object-Oriented, etc are a few names among many possibilities. In addition, the responsibilities of different developers in a team makes a difference in how their chosen methodology functions. My

interviews included software engineers, database engineers, and web engineers; future study should narrow the interview pool to one specific kind of software developer.

- What are the rules for an oral communication genre in development? When there is a purpose, rhetorical situation and meaning-making community there is also a set of parameters to isolate community knowledge, determine the standards for enculturation and clear criteria for a structure that must be both stabilized and reconstituted. The meta-language needs to break up the genre ecology and outline how the various rhetorical forms are used and what oral communication does or does not count as a "documentation" activity.

Three important issues to resolve with future research. I do not think the interview questions prompted enough discussion of oral documentation practices and I do not think I was aggressive enough countering noble industry sentiments with "That is a well stated documentation principle but you do not document. So what do you do instead if you don't document?" I did not want to tell them to tell me that oral communication was a new rhetorical form; however, I should have done more to get the interviewees talking about oral communication.

With the documentation success of the American West Heritage Center Tour project and the documentation failure of the Engineering Modules project, I wonder how much more remote developers might rely on documentation than onsite developers. Remoteness can be a barrier that traditional documentation resolves. However, if the interviewees are under such pressure to complete development projects for the departments they serve, then how come the remote employees still managed to produce documentation?

I need to verify oral communication is not simply a poor substitute for what should be a best practice. Even if I'm verifying that professional developers do in fact employ oral communication as a valid rhetorical form, is it simply the company will not commit resources to support healthy documentation practices? I might be simply justifying a poor solution. Perhaps developers should just document and stop making excuses.

REFERENCES

Adams, E., & Rollings, A. (2003). *Andrew Rollings and Ernest Adams on game design*. Berkeley, CA: New Riders Games.

Adams, E., & Rollings, A. (2007). *Fundamentals of game design*. Upper Saddle River, NJ: Pearson Prentice Hall.

Antoine, V. (1985). The software documenter: A new specialist. *Technical communication, 32*(3), 16-18.

Askehave, I., & Nielsen, A.E. (2005). *What are the characteristics of digital genres? - Genre theory from a multi-modal perspective.* Paper presented at the System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference.

Bakhtin, M.M. (Ed.). (2007). *Speech genres and other late essays* (11th ed.). Austin, TX: University of Texas Press.

Barker, T.T. (1998). *Writing software documentation: A task-oriented approach*. Boston, MA: Allyn and Bacon.

Barker, T.T. (2003). *Writing software documentation: A task-oriented approach*. Boston, MA: Allyn and Bacon.

Barrett, K.H., John; Hilmer, R.; Posner, D.; Snyder, G.; Wu, D. (2003). Pseudo interactive's cel damage. In A. Grossman (Ed.), *Postmortems from game developer*. San Francisco, CA: CMP Books.

Barthes, R. (1975). *The pleasures of the text*. New York, NY: Hill and Wang.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.

Berkenkotter, C., & Huckin, T.N. (1995). *Genre knowledge in disciplinary communication: Cognition, culture, power*. Hillsdale, NJ: L. Earlbaum Associates.

Berkun, S. (2005). *The art of project management*. Sebastopol, CA: O'Reilly.

Blyler, N.R., & Thralls, C. (1992). The social perspective and professional communication: Diversity and directions in research. In N. R. Blyler & C. Thralls (Ed.), *Professional communication: The social perspective* (pp. 3-34). Newbury Park, CA: Sage.

Brown, D.M. (2007). *Communicating design: Developing web site documentation for design and planning*. Berkeley, CA: Peachpit Press : New Riders.

Bruffee, K. (1986). Social construction, language, and the authority of knowledge: A bibliographical essay. *College english, 48*, 773-790.

Brumberger, E.R. (2003). The rhetoric of typography: The awareness and impact of typeface appropriateness. *Technical communication, 50*(2), 224-231.

Brumberger, E.R. (2004). The rhetoric of typography: Effects on reading time, reading comprehension, and perceptions of ethos. *Technical communication, 51*(1), 13-24.

Burke, K. (1945). *A grammar of motives*. New York, NY: Prentice-Hall, inc.

Burke, K. (1950). *A rhetoric of motives* ([1st ed.). New York, NY: Prentice-Hall.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R. (2002). *Documenting software architectures: Views and beyond*. Upper Saddle River, NJ: Addison-Wesley.

Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis, IN: Sams.

Denton, L., & Kelly, J. (1993). *Designing, writing, and producing computer documentation*. New York: McGraw-Hill.

Doumont, J.-l. (2002). Verbal versus visual: A word Is worth a thousand pictures, too. *Technical communication, 49*(2), 219-224.

Dragga, S., & Voss, D. (2003). Hiding humanity: Verbal and visual ethics in accident reports. *Technical communication, 50*(1), 61-82.

Emerson, R.M., Fretz, R.I., & Shaw, L.L. (1995). *Writing ethnographic fieldnotes*. Chicago, IL: University of Chicago Press.

Foss, K.A., Foss, S.K., & Trapp, R. (Eds.). (2003). *Contemporary perspectives on rhetoric* (3rd ed.). Prospect Heights, IL: Waveland Press.

Freedman, A., & Medway, P. (1994). Locating genre studies: Antecedents and prospects. In A. Freedman & P. Medway (Eds.), *Genre and the new rhetoric* (pp. 1-20). Bristol, PA: Taylor & Francis.

Grant-Davie, K. (1997). Rhetorical situations and their constituents. *Rhetorical review, 15*(2), 264-279.

Grossman, A. (2003). *Postmortems from game developer*. San Francisco, CA: CMP Books.

Hailey, C.E., & Hailey, D.E. (1998). Hypermedia, multimedia, and reader cognition: An empirical study. *Technical communication, 45*(3), 330-342.

Hailey, D. (2014). *Reader centric writing for digital media—Theory and practice*. Amityville, NY: Baywood Press (in press).

Hailey, D.E., & Hailey, C.E. (2002a). *Genre theory, technology, and knowledge distribution.* Paper presented at the Professional Communication Conference, 2002. IPCC 2002. Proceedings. IEEE International.

Hailey, D.E., Jr., & Hailey, C.E. (2002b). *Genre theory, engineering education, and circumventing internet bandwidth problems.* Paper presented at the Frontiers in Education, 2002. FIE 2002. 32nd Annual.

Harmon, W., & Holman, C.H. (1996). *A handbook to literature* (7th ed.). Upper Saddle River, NJ: Prentice Hall.

Johnson-Sheehan, R., & Baehr, C. (2001). Visual-spatial thinking in hypertexts. *Technical communication, 48*(1), 22-30.

Kent, T. (1986). *Interpretation and genre: The role of generic perception in the study of narrative texts*. Lewisburg [PA.] London: Bucknell University Press; Associated University Presses.

Kent, T. (1993). *Parologic rhetoric*. Lewisburg, PA: Bucknell University Press.

Larman, C. (2003). *Agile and iterative development: A manager's guide*. Boston, MA: Addison-Wesley Professional.

Markel, M. (1998). Testing visual-based modules for teaching writing. *Technical communication, 45*(1), 47-76.

McAllister, K.S. (2004). *Game work: Language, power, and computer game culture*. Tuscaloosa, AL.: University of Alabama Press.

Miller, C.R. (1979). A humanistic rationale for technical writing. *College english, 40*(6), 610-617.

Miller, C.R. (1994). Genre as social action. In A. Freedman & P. Medway (Eds.), *Genre and the new rhetoric* (pp. 23-42). Bristol, PA: Taylor & Francis.

Nietzsche, F. (1914). *The will to power: An attempted transvaluation of all values* (A. M. Ludovici, Trans.). London, UK: T. N. Foulis.

Qiuye, W. (2000). A cross-cultural comparison of the use of graphics in scientific and technical communication. *Technical communication, 47*(4), 553-560.

Ragaini, T. (2003). Turbine's asheron's call. In A. Grossman (Ed.), *Postmortems from game developer*. San Francisco, CA: CMP Books.

Robinson, P.A., & Etter, R. (2000). *Writing and designing manuals*. Boca Raton, FL: CRC Press.

Rüping, A. (2003). *Agile documentation: A pattern guide to producing lightweight documents for software projects*. Chichester, England; Hoboken, NJ: Wiley.

Rutter, R. (1991). History, rhetoric, and humanism. *Journal of technical writing and communication, 21*(2), 133-153.

Schultz, C., Bryant, R., & Langdell, T. (2005). *Game testing all in one*. Boston, MA: Thomson Course Technology and Premier Press.

Spector, W. (2003). Ion storm's deus ex. In A. Grossman (Ed.), *Postmortems from game developer*. San Francisco, CA: CMP Books.

Spinuzzi, C. (2001). *Software development as mediated activity: Applying three analytical frameworks for studying compound mediation*. Paper presented at the 19th annual international conference on computer documentation, Sante Fe, NM, USA.

Spinuzzi, C. (2002). *Modeling genre ecologies*. Paper presented at the 20th annual international conference on computer documentation, Toronto, Ontario, Canada.

Spinuzzi, C. (2003). *Tracing genres through organizations: A sociocultural approach to information design*. Cambridge, MA: MIT Press.

Spinuzzi, C. (2004). *Four ways to investigate assemblages of texts: Genre sets, systems, repertoires, and ecologies*. Paper presented at the 22nd annual international conference on design of communication: The engineering of quality documentation, Memphis, TN, USA.

Spinuzzi, C., Hart-Davidson, W., & Zachry, M. (2006). *Chains and ecologies: Methodological notes toward a communicative-mediational model of technologically mediated writing*. Paper presented at the 24th annual ACM international conference on design of communication, Myrtle Beach, SC, USA.

Thomas, J.B. (1987). *Honoring the farm: identifty and meaning in personal narratives.* (M.S.), Utah State University, Logan, UT.

Upton, B. (2003). Red storm entertainment's rainbow six. In A. Grossman (Ed.), *Postmortems from game developer*. San Francisco, CA: CMP Books.

Williams, A. (2003a). *Assessing genre as rhetorical performance in software design.* Paper presented at the IEEE international professional communication conference, Orlando, FL, USA.

Williams, A. (2003b). *Examining the use case as genre in software development and documentation*. Paper presented at the 21st annual international conference on Documentation, San Francisco, CA, USA.

Winsor, D.A. (1990). Engineering writing/writing engineering. *College composition and communication, 41*(1), 58-70.

Winsor, D.A. (1999). Genre and activity systems: The role of documentation in maintaining and changing engineering activity systems. *Written communication, 16*(2), 200-224. doi: 10.1177/0741088399016002003

Yates, J., & Orlikowski, W. (2002). Genre systems: Structuring interaction through communicative norms. *Journal of business communication, 39*(1), 13-35.

Zachry, M. (2005). Paralogy and online pedagogy. In K. Cargile Cook & K. Grant-Davie (Eds.), *Online education* (pp. 177-192). Amityville, NY: Baywood Publishing.

APPENDIX

APPENDIX 1: IRB APPROVAL

UtahState University OFFICE OF RESEARCH AND GRADUATE STUDIES	**Institutional Review Board** USU Assurance: FWA#00003308 **Exemption #2** **Certificate of Exemption**	Full Accreditation

FROM: Richard D. Gordin, Acting IRB Chair True M. Rubal, IRB Administrator	[illegible] [illegible]

To: David Hailey, Jason Cootey
Date: March 29, 2012
Protocol #: 4331
Title: Agile Documentation Practices

The Institutional Review Board has determined that the above-referenced study is exempt from review under federal guidelines 45 CFR Part 46.101(b) category #2:

> Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless: (a) information obtained is recorded in such a manner that human subjects can be identified, directly or through the identifiers linked to the subjects: and (b) any disclosure of human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

This exemption is valid for three years from the date of this correspondence, after which the study will be closed. If the research will extend beyond three years, it is your responsibility as the Principal Investigator to notify the IRB before the study's expiration date and submit a new application to continue the research. Research activities that continue beyond the expiration date without new certification of exempt status will be in violation of those federal guidelines which permit the exempt status.

As part of the IRB's quality assurance procedures, this research may be randomly selected for continuing review during the three year period of exemption. If so, you will receive a request for completion of a Protocol Status Report during the month of the anniversary date of this certification.

In all cases, it is your responsibility to notify the IRB prior to making any changes to the study by submitting an Amendment/Modification request. This will document whether or not the study still meets the requirements for exempt status under federal regulations.

Upon receipt of this memo, you may begin your research. If you have questions, please call the IRB office at (435) 797-1821 or email to irb@usu.edu.

The IRB wishes you success with your research.

4460 Old Main Hill	Logan, UT 84322- 4460	PH: (435) 797-1821	Fax: (435) 797-3769	WEB: irb.usu.edu	EMAIL: irb@usu.edu

APPENDIX 2: LETTER OF INFORMATION

Department of English
3200 Old Main Hill
Logan UT 84321
Telephone: (435) 797-2733

LETTER OF INFORMATION
Agile Documentation Practices

Introduction/ Purpose Dr. David Hailey in the Department of English: Theory and Practice of Professional Communication at Utah State University is faculty advisor for a research study to find out more about internal Agile software documentation practices. Jason Cootey is a PhD student conducting this research study for his dissertation project. You have been asked to take part because your organization has been identified as an Agile developer who maintains internal software documentation. There will be approximately five total participants in this research.

Procedures If you agree to be in this research study, you will need sit for an interview. You are not required to answer all the questions in the interview but answers to all questions will return the most value for the time taken. The interview will be a verbal conversation, with close to a dozen talking points, which will last between 30-50 minutes.

One consideration that will improve the quality of the interview is the inclusion of proprietary documentation. While documentation is not required, a reference will facilitate discussion and make answering questions easier. You can provide documentation for the duration of the interview in print form or on a computer. After the interview, you may withdraw the proprietary documentation. However, Jason Cootey would like to negotiate the usage of sample documentation, if at all possible.

Risks Participation in this research study may involve some added risks or discomforts. These are limited to the management of intellectual property and proprietary operations documentation. It is not the intention of either Jason Cootey or Dr. David Hailey to disclose proprietary data to either your competitors or other professional communication researchers. Due to the elicited nature of the interview procedure, conditions for the use of this information will be negotiated at the conclusion of the interview. Strict adherence to those conditions will be maintained in the publication of research data. Any ongoing usage of the research study data will be included in the conditions.

Benefits No direct or possible benefits, major or minor, to the research participants or to others that may be reasonably involved in the proposed research, are expected, either now or in the future.

Voluntary nature of participation and right to withdraw without consequence
Participation in research is entirely voluntary. You may refuse to participate or withdraw at any time without consequence or loss of benefits. You may use the following two methods to register your intent to withdraw: send an email message to j.cootey@aggiemail.usu.edu.

Confidentiality Research records will be kept confidential, consistent with federal and state regulations. Only Jason Cootey and Dr. David Hailey will have access to the data that will be kept in a password protected computer in a locked room. If you require access to the research data then you can request access during the interview visit; the interview data is recorded with proprietary technology that is accessible through a proprietary, password protected website for 60 days after the access request. To protect your privacy, no identifying information will be gathered for the purposes of this research study. The research data will remain in Jason Cootey's possession for the duration of the research study for future reference.

IRB Approval Statement The Institutional Review Board for the protection of human participants at Utah State University has approved this research study. If you have any questions or concerns about your rights or a research-related injury and would like to contact someone other than the research team, you may contact the IRB Administrator at (435) 797-0567 or email irb@usu.edu to obtain information or to offer input.

Investigator Statement "I certify that the research study has been explained to the individual, by me or my research staff, and that the individual understands the nature and purpose, the possible risks and benefits associated with taking part in this research study. Any questions that have been raised have been answered."

Signature of Researcher(s)

______________________________ ______________________________

David Hailey, PhD *Jason Cootey, M.S.*
Principal Investigator Student Researcher

VITA

Jason L. Cootey

Education

2014 PhD, Theory and Practice of Professional Communication, Utah State University Logan, Utah

2006 Master of Science, Literature and Writing, Utah State University, Logan, Utah

2001 Honors Bachelors of Arts, English Literature, University of Utah, Salt Lake City, Utah

2001 Bachelors of Arts, Psychology, University of Utah, Salt Lake City, Utah

2000 Shakespeare Summer Program, Cambridge University, Cambridge, England

Academic Achievement

Salt Lake Community College Technical Writing Program 2013

Utah Valley University Adjunct Faculty appointment 2011

Project Director Technical Communication student development projects 2009 and 1010

Facilitator on Engineering Video Course development grant 2009

Graduate Student Stipend Enhancement Award 2009

Research Assistant on the Interdisciplinary Media Research Consortium grant 2007-2008

Student Athlete Instructor Award Spring 2006

Research Assistant on the Creative Learning Environment grant 2006-2007

The Marion D. and Maxine C. Hanks Foundation Grant 2004

Utah State University Graduate Student Stipend for teaching

Courses Taught

ONLINE COURSES
Utah State University Online Courses
Graduate Instructor, English 1010, Introduction to Writing, 1 section Fall 2009
Graduate Instructor, English 2010, Intermediate Writing, 1 section Spring 2008
Graduate Instructor, English 1010, Introduction to Writing, 1 section Spring 2008

Stevens-Henager College Graphic Design Software Online Courses
Adjunct Faculty, 2 sections Summer 2010

CAMPUS COURSES
Salt Lake Community College, English 2100, Technical Writing
Adjunct Faculty, 3 sections Spring 2014
Adjunct Faculty, 2 sections Fall 2013

Utah Valley University, English 1010, Introduction to Writing
Adjunct Faculty, 2 sections Fall 2011

Utah State University English 3080 Technical Writing for Non-English Majors
Graduate Instructor, 2 sections Spring 2010
Graduate Instructor, 1 section Fall 2009

Utah State University English 2010 Intermediate Writing
Graduate Instructor, 2 sections Spring 2009
Graduate Instructor, 2 sections Fall 2008
Graduate Instructor, 1 section Fall 2007
Graduate Instructor, 2 sections Spring 2007
Graduate Instructor, 1 section Spring 2006
Graduate Instructor, 2 sections Fall 2005

Utah State University English 1010 Introduction to Writing
Graduate Instructor, 2 sections Fall 2006
Graduate Instructor, 2 sections Spring 2005
Graduate Instructor, 2 sections Fall 2004

Workshops

Spring 2006. Utah State University Learning Games Initiative. Neverwinter Nights Design Tool Orientation. Objective: lead students in research discussion, while also collaborating about design ideas.

Fall 2005. Utah State University Composition Program. Panel of second year student graduate instructors for incoming graduate instructors. Objective: familiarize new graduate instructors to teaching at the university level through interaction with second year peers.

Publications

Revised Submission Requested "Creating Community Narratives: Patterns that form Narratives in Community MMORGs" as a web article for *Kairos*.

Revised Submission Requested "From the Hive Mind: demonstrating the loss of the writer's personal space." Invited to revise by *Computers and Composition*.

"Usability Testing, User Goals, Engagement, and *Aristotle's Assassins*." *Usability of Complex Information Systems: Evaluation of User Interaction*. Chapter 15. Eds. Michael J. Albers and Brian Still. Boca Raton, FL: CRC press. 2011

"Playing in Genre Fields: A Play Theory Perspective on Genre." SIGDOC proceedings. Co-authored with Ryan M. Moeller and David M. Christensen. 2007.

"'The Peripatos could not have looked like that,' and other educational outcomes from student game development," *Games and Simulations*. Book chapter. Co-authored with Ryan M. Moeller and Ken S. MCallister. Eds. Brett E. Shelton and David A. Wiley. Rotterdam, The Netherlands: Sense Publishers. 2007.

"I've Looked Deep Into the Darkness*." Nebula: Generalist* 3.4. November 2006.

"Culpability and Transgression in the Monomania of Ahab." Abstract pulished in *Leviathon*.

"The Suppressed (or lifted) Version of Joseph Conrad's Heart of Darkness." *Myths of Self*, Special Edition, Utah State University, 2005.

"Walking off the Dover Cliff." Journal of the Wooden "O" Symposium 2004. Editor in Chief Diana Major Spencer.

"Analysis of Interchange in *A Midsummer Night's Dream*." Proceedings National Conference on Undergraduate Research NCUR 2003. (Abbreviated version) Editor in Chief Robert D. Yearout.

"Analysis of Interchange in *A Midsummer Night's Dream*," Honors Senior Thesis 2001 University of Utah Marriott Library

Advisor: Professor Morriss Partee

Conferences

Submission: "Innovative Software Documents and New Rhetorical Forms" Rocky Mountain Modern Language Association. Salt Lake City, UT. October 2014.

"User-generated Computer Game Manuals as a Force for Change on Professional Practice." at the Rocky Mountain Modern Language Association. Salt Lake City, UT. October 2009.

"If This Isn't Real, Then What Is It? New Lexicon for Virtual Worlds and MMORPGs" at the Virtual World Best Practices in Education VWBPE Conference. Hosted in the *Second Life* MMORG world. March 2009

"Creating Community Narratives: Patterns that form Narratives in Community MMORGs" at the Southwest Popular Culture Association. Albuquerque, NM. February 2009.

"Classroom Interfaces, Access, and *Second Life*" at the Intermountain Graduate Conference. Utah State University. April 2008.

"I Know What You Didn't Do Last Summer: Using Educational Game Development to Motivate Students" at the Southwest Popular Culture Association. Albuquerque, NM. February 2008.

"Turning Operators into Machines. Teaching the Relationship between Humans and Technology" at the Popular Culture Association. Boston, MA. April 2007

"What Textbooks and Templates Don't Teach about Design Documentation" at the Southwest Popular Culture Association. Albuquerque, NM. February 2007.

"Multimodal Outcomes: Using Game Design to Meet WPA Goals for First-Year Composition" at the Two Year Colleage Association West Conference. Park City: October 2006

"Communication, Modality, and Interface in Online Video Games" at the Intermountain Graduate Conference. Utah State University: April 2006.

"Derrida Purloins Poe's Reader" at the Rocky Mountain Modern Language Association. Coeur d'Alene, ID: October 2005.

"Reminiscence: the Psychological Value of Natural Spaces After Wordsworth Leaves the Woods" at the Association for Studies in Literature and Environment.

University of Oregon in July 2005.

"Culpability and Transgression in the Monomania of Ahab" at the American Literature Association in Boston, MA: May 2005.

Research Experience

Primary Investigator—Interviewer—Software Developer Interviews

- Spring 2013 Seven Interview Sessions
- Planned 14 interview questions based on theoretical model
- Used the North American Genre Theory model of document assessment

Research Assistant—facilitator—Engineering Video Course development

- Summer 2009 project grant
- Cooperation with English and Engineering
- Manage undergraduate filming and editing RAs
- Facilitate weekly progress with film and editing
- Develop production protocols
- Coordinate faculty schedules, course schedules, and film crew schedules

Research Assistant—project manager—Interdisciplinary Media Research Consortium

- Spring 2007, Summer 2007, and Fall 2007 project grant
- Research assistant cooperation with English, Instructional Technology, and Graphic Design
- Manage undergraduate Graphic Design RAs
- Report weekly progress to the local IMRC
- Assign and follow up on tasks assigned to undergraduate RAs

Research Assistant—project manager—Creative Learning Environment

- Spring 2006 semester project grant
- Research assistant cooperation with English and Instructional Technology
- Manage undergraduate RAs
- Report weekly progress to the national Learning Games Initiative
- Update the "Design Document" for the project software
- Organize design tasks for undergraduate RAs
- Research publication venues for research

Research Assistant for Librarians at the University of Utah Marriott Library

- Train patrons on the usage of Library databases
- Create research solutions with patrons

Research for Honors Senior Thesis

- Research work completed in both Marriott Library and Cambridge

University Library
• Extensive class work in both Utah and England

Research Assistant in Psychology Sense and Perception Lab
• Connect probes to skull for ERP experiments that test correlational relationship between cell phone usage behind the wheel and drunk driving
• Carefully observe ERP screens to insure experiment succeeds
• Research trials to test the efficiency of various interface formats for anesthesiology computer screens

Academic Committee Work

Mentorship Committee for PhD English Students
• Chair and founder
• Community of advice and support
• Incoming student welcome get-togethers

Reviewer for ITSE special issue.
• International Journal of Interactive Technology and Smart Education
• Recommended submissions for publication

English Department Library Committee
• Represent English Department during library policy changes

English Department Travel Committee
• Review English Department travel policy

English Department University Studies/Breadth and Depth Humanities Committee
• Review General Education requirements for the Composition Program

Student Association of Graduates of English (SAGE) Web Presence Committee
• Distribute assignments for informational updates
• Webmaster
• Design and update SAGE website

Special Activities Committee
• Generating the Intermountain Graduate Conference in cooperation with Idaho State University
• Promotion of the 2005 and 2007 conferences
• Preparation for USU to host the Philological Conference next year

Computer Action Committee
• Work with colleagues to clarify computer problems before reporting to

the computer technicians
- Negotiate with technician staff for timely service
- Liaison of technician staff to office colleagues

Pilot Assessment Program
- The assessment is an instrument for the Writing Program's accredidation
- Administer assessment prompt to English 1010 and 2010 students
- University reader for assessment papers turned in by students

Service to/in the Community

Volunteer Employment Councelor
- American Fork, UT Employment Center Resume Assistance (2013)
- Sandy, UT Employment Center Resume Assistance (2012)
- Logan, UT Employment Center Resume Assistance (2011)

Student Community Writing Projects Coordination and Guidance
- Disability Resource Center equipment/software technical descriptions (2010)
- Collaboration software instructions for campus computer services (2009)
- Healthcare Reform Brief for Utah Senators (2009)
- Instruction materials for PTA red-ribbon week (2009)
- Pamphlet for local Animal Shelter (2009)
- Simulation manual for local High School debate team (2009)
- PTA red-ribbon week service presentations with school children (2008)
- Graphical software file conversion instructions for grant project IMRC (2008)

American West Heritage Center 2009
- Programming an Educational Simulation
- Design Documentation
- Promotional Assessment and Materials

Utah State University Cycling Team (2006-2009)
- Homecoming Parade organization
- Fit 200 elementary school students with helmets at local school (2009)
- Team fundraising
- Colleagiate racing in Colorado/Wyoming Circuit
- Recruitment

Epilepsy Awareness for Utah State University Undergraduates (2004-2005)
- Surveys

- Awareness Lectures
- Preparing to generate informational pamphlets for University faculty

Poetry Workshop at residential facility for at-risk youth (2005)

- Poetry presentation
- Lead exercises for poetry groups
- Judge Poetry talent show
- Poetry reading

Board member of Epilepsy Association of Utah (2003-2004)

- Organize fund raisers
- Work out a budget and spending
- Public Education
- Run statewide support groups
- Patient education for families with new diagnosis of epilepsy