Conjectures of Informal Communication-Centric Practices Observed in a Distributed Software Development Team

Lucas Layman[a,*], Laurie Williams[a], Daniela Damian[b], Hynek Bures[c]

[a] *Department of Computer Science, North Carolina State University, 900 Main Campus Drive, Raleigh, NC 27695, U.S.A.*

[b] *Department of Computer Science, University of Victoria, Victoria, BC V8W 3P6, Canada*

[c] *Radiante Corporation, 3108 Falconhurst Dr., Wake Forest, NC 27587*

Abstract

The challenges of distributed software development seem to preclude the use of informal communication-centric processes, such as agile methodologies. This paper describes an industrial case study of a distributed team in the US and Europe that successfully used Extreme Programming. The team overcame linguistic and technical issues, established fruitful developer-customer communication, and successfully developed a product in a new problem domain. The goal of our industrial case study is to better understand how the practices used by this distributed team enabled them to employ a communication-rich methodology and overcome the hurdles associated with distributed development. We collected quantitative and qualitative data and used grounded theory to develop conjectures about the practices used by the team that made them successful in a methodology focused on informal communication. Five conjectures emerge from our study. These conjectures include having a key member of one team physically locate with the other team; having a well-defined customer authority; maintaining prompt and dependable turnaround time on asynchronous queries; having short iterations; and having continuous access to product and process information.

1. Introduction

Global software development is becoming common practice in today's industry. The ability to develop software using remote teams allows software organizations to break down geographical distances, to benefit from access to a larger resource pool and to reduce development costs. Furthermore, over the last decade, outsourcing of information technology (IT) has become a mainstream business phenomenon, wherein offshore management is emerging as a critical business component for firms. The amount of software exports from India alone increased 16-fold from 1995-2002 [45]. The software production industry in India has seen annual growth rates of 30-40% in both employment and revenue in each of the last 10 years [3], and eight out of 10 large corporate IT departments are projected to outsource part of their technology services [36]. The popular press has reported that as much as 50% of IT jobs will be outsourced to offshore sites by 2015 [2]. Developing software in distributed teams has brought about its own unique set of problems. Issues regarding cultural and language differences, trust and commitment, extended feedback loops, asynchronous communication, and knowledge management add new difficulties to today's already-complicated software development [16, 30].

Researchers and practitioners need to better understand these issues of distributed software development (DSD) so that risk mitigation practices can be studied and incorporated. To this end, we conducted an industrial case study of a distributed team. Our case study involved a project contracted to an organization with a development group in the Czech Republic, Radiante Corporation, by a U.S. telecommunications software and hardware provider, Tekelec. The project involved the creation of a simulator for a telecommunications signal transfer point system and was the first time the two organizations had collaborated on this technology. Project management personnel, including the customer, project manager, project tracker, and development manager were located in the U.S. The contracted development team resided in the Czech Republic. All of the project management personnel were employees of Tekelec except for the development manager, who was the vice-president of Radiante Corporation. The distributed team used the Extreme Programming (XP) agile software development process [9] for the first time and encountered a considerable amount of requirements volatility. XP relies heavily on informal, face-to-face communication throughout the development lifecycle. This reliance on informal communication can be postulated as presenting challenges to a distributed team. The project was successfully completed and put into production after approximately six months of development. The resultant software is currently being used by Tekelec's customer service personnel to train new customers.

* Corresponding author: Tel: +1-919-513-5082

E-mail addresses: lmlayma2@ncsu.edu (Lucas Layman), williams@csc.ncsu.edu (Laurie Williams), DanielaD@cs.uvic.ca (Daniela Damian), hynek@radiante-corp.com (Hynek Bures)

The goal of our industrial case study was to better understand how the informal communication-centric practices used by a distributed team enabled the team to overcome the hurdles associated with distributed development. We present a series of observations derived from both qualitative and quantitative data collected during the course of the project; data collection was structured via the Extreme Programming Evaluation Framework (XP-EF) [56]. Analysis was conducted of the team's project artifacts, and interviews were conducted with the customer and project personnel. Specifically, we discuss how a project following the XP methodology was able to address linguistic and technical issues, to establish developer-customer communication, and to develop a project in an unfamiliar problem domain. As will be discussed, five conjectures concerning best, informal communication-centric practices for distributed software development teams emerge from our study. Based upon our observations, and in consultation with the project stakeholders, we provide a list of recommendations for software development teams that may find themselves in a similar situation.

The remainder of this paper is organized as follows: Section 2 describes related work, Section 3 outlines our research methodology, and Section 4 describes the software project in detail. Section 5 provides case study observations and conjectures, Section 6 discusses the limitations of our study, Section 7 lists recommendations for best practices, and we conclude in Section 7.

2. Background and related work

In this section, we provide a brief introduction to communications and requirements engineering practices in global software development. We also provide background information on XP and on the XP-EF, which was used to structure the data collection in our case study. Finally, we provide the results of other studies of co-located XP teams.

2.1. Global software development and requirements engineering practices

The processes of communication, coordination, and collaboration are at the heart of, and key enablers of, software development processes. Research continues to reveal the significant impact that distance has on project performance [30, 46]. In particular, communication seems to be an essential component of all software development collaboration practices and processes. Besides formal project communication, empirical studies suggest that developers rely heavily on informal, ad hoc communication [15, 26, 39, 48]. Consequently, hurdles in communication will have dramatic effects in the success of DSD projects.

Besides differences in language and culture, DSD teams suffer from a lack of informal communication, resulting in low levels of trust and awareness of work and progress at remote sites. In managing cross-cultural issues in global software development, strategies recommended in the literature include the use of cultural liaisons [12], bridgehead teams (sales and customer liaison groups) [40], straddlers (technical or managerial liaisons) [28], or rotation of management to cope with cultural diversity [19].

In particular, one class of software development activities directly affected by challenges in communication are requirements-related activities, such as requirements definition and negotiation, design, and project management. Activities of requirements engineering are one of the major difficulties in multinational organizations [50]. An in-depth field study of requirements engineering in a multinational organization [17] clearly identifies the multitude of factors impacting the effective management of requirements in global software development. The language barrier alone could be a significant problem in achieving shared understanding of the required functionality. Terms that convey different meaning in different organizational settings may be misinterpreted until late in the project with potentially damaging results. For example, "overlooking the development of a feature" may mean overseeing its development or simply forgetting its development. Further, multicultural interaction between developers and clients (internal or external to the developer organization) together with time delays and the inability to maintain an awareness of working context at remote sites contribute to major challenges affecting the entire software development process. Failure to fully understand the required system features, reduced trust and the inability to effectively resolve conflicts result in budget and schedule overruns and, ultimately, in damaged client-supplier relationships [17].

Offshore development is a particular case of global software development that is bound to suffer from these hurdles, given the reliance on geographically distributed client-supplier relationships. As requirements management activities are primarily impacted by distance [17], global projects benefit from a requirements specification well-defined at the onset of the project, thus avoiding the need to reiterate feature understanding. This specification is often used in planning processes, important in the organization and managing of distributed projects [50]. Frequent deliveries are also recommended as a successful DSD collaboration practice [47]. Incremental integration and

frequent deliveries are also recognized as powerful techniques for managing requirements creep [34], and are a core practice of agile methodologies for co-located projects [41]. Though the study of agile practices in global software development has received little attention, there is some evidence that agile practices can be used in offshore development [24]. In this paper we build upon these initial results.

2.2. Extreme Programming

The development team in our study followed the XP software development methodology. XP is an agile [13] software development process. XP was designed for use with 10-12 co-located, object-oriented programmers [32]. XP is an iterative development methodology with a short planning horizon (three month releases, 1-2 week iterations). In XP, a release corresponds to a stable, deliverable version of the product that can be used by customers. Iterations are shorter increments of development where individual tasks are assigned to developers and a working prototype of the system is evolved and periodically evaluated by project stakeholders. The XP practices do not include the preparation of extensive requirements or design documents prior to starting development. Consequently, XP is heavily reliant on continuous communication between stakeholders and tight feedback loops to clarify and specify feature implementation and to respond to change. This need for continuous communication can be a challenge for DSD teams and is the central theme in this paper.

In XP, functional requirements are captured as user stories. User stories are informal, natural language descriptions of system features that are written on an index card. Each user story is written by the customer, who is also responsible for stating the priority of the user story and for providing a corresponding customer acceptance test that, when passed, marks the completion of the user story. Since the initial user story often does not contain enough information for the feature to be implemented, developers and customers have an on-going dialogue throughout the development process through which user stories are clarified and further refined. Through this on-going communication, the customer retains control over the product being developed.

2.3. Extreme Programming Evaluation Framework

We use a benchmark instrument called the Extreme Programming Evaluation Framework (XP-EF) to guide our longitudinal case studies of XP teams [42, 43, 56]. By using a measurement framework for all XP case studies, the case study results can be compared. The XP-EF has three parts:

- **Context factors (XP-cf).** Recording a case study's context factors (XP-cf) is essential for fully understanding the generality and utility of the conclusions as well as the similarities and differences between the case study and one's own environment. One cannot assume a priori that a study's results generalize beyond the specific environment in which it was conducted [7].
- **Adherence metrics (XP-am).** Most companies that use XP adopt the practices selectively and develop customized approaches to operate within their particular contexts [20]. These metrics provide insight into whether a team has adopted XP's practices. Since the focus of this paper is on communication, particularly relative to managing user stories and customer-developer interaction, we do not report on adherence to all of XP's practices in this paper.
- **Outcome measures (XP-om).** These metrics provide information to managers who want to know the business-related results of employing XP.

The XP-EF information for the project in our study can be found in Section 4.

2.4. Extreme Programming case studies

Practitioners and researchers have reported numerous, predominantly anecdotal and favorable, studies of the XP methodology. There is a lack of industrial case studies of XP in general, and of studies of XP in global software development in particular. However, some empirically-based XP case studies do exist. Abrahamsson [1] conducted a controlled case study of four software engineers using XP to implement data management software. Comparison between the first and second releases of the project yielded increases in planning estimation accuracy and productivity while the defect rate remained constant. Similarly, Maurer and Martel [44] reported a case study of a nine-programmer web application project. The team showed strong productivity gains after switching from a document-centric development process to XP.

Several case studies were structured using the XP-EF. A year-long case study was performed with a small, co-located team (7-11 team members) at IBM to assess the effects of adopting XP [56]. Through two software releases, this team transitioned to and stabilized its use of a subset of XP practices. The use of a subset of the XP practices was necessitated by corporate culture, project characteristics, and team makeup. The team improved productivity, reduced pre-release defect density by 50%, and achieved a 40% reduction in the post-release defect density when compared to the same metrics from an earlier release. A similar case study was performed with a small, co-located team at Sabre Airline Solutions™ [43]. The study compared the business-related results of a product developed by the Sabre-A team using traditional methods and a later release developed using XP. The study showed a 65% reduction in the product's pre-release defect rates, a 35% reduction in the post-release defect rates, and a 50% improvement in productivity (as measured by code output) in the XP release. A second case study was performed at Sabre Airline Solutions™ [42] with a larger, relatively inexperienced team. The study showed that the team yielded above-average post-release quality and average to above-average productivity relative to industry averages when the team utilized the XP methodology. These case study results suggest that the usage of XP can help improve the business-related results (such as quality and productivity) of teams operating within certain contexts. However, there has been virtually no research on the use of XP in a DSD environment where the informal communication required in XP may become strained.

3. Research method

Our research is an industrial case study of the entire lifecycle of one project. Case studies are used to collect data through observation of a project in an unmodified, contextual setting [58]. Case studies (or N=1 experiments [27]) can be viewed as "research in the typical" [23, 37] and are an evaluative technique carried out with realistic tasks and realistic subjects in a realistic environment [22, 54]. However, comparisons made via case studies can never be perfect due to the complex contexts of the projects. Yin [57] asserts that case studies should be utilized for "how or why" research questions, when the investigator has little control over the events, and when the focus is on a contemporary phenomenon with some real-life context. The limitations of our case study are described in Section 6.

Our case study covers an entire project, the period from initial user story gathering, through implementation, a release point, maintenance of the system, and finally the end of product development. The released product is still in use. However, personnel were reallocated to different projects. Currently the project has received resources and will be resumed in the near future.

To preserve the complexity of our case study data, a qualitative research approach called grounded theory [25] was used. With grounded theory, theories can emerge from gathered data, in contrast to research in which data is gathered to support or refute a theory. The intent is to generate or discover a theory that is "grounded" in data from the field. In grounded theory, new theory comes from a small sample size of cases, exploring a wide variety of variables. This approach is suitable for our study into the best practices of this distributed team since there we were not investigating any preconceived theories of what might make the distributed team successful or unsuccessful. In this research, we do not provide formal hypothesis testing or draw any general conclusions. However, we conjecture about some best practices for distributed teams based upon evidence from this case study.

As discussed in the following two subsections, we use a variety of qualitative and quantitative resources as a basis for our findings. Our study also consists of several pieces of quantitative information. By combining both quantitative and qualitative techniques, researchers can triangulate on their object of study. Through triangulation, two or more distinct methods are used to study the same phenomenon. Convergence and agreement between the results of multiple methods enhances the belief that the results are valid and not a methodological artifact [33]. Triangulation also captures a more holistic and contextual portrayal of the factors under study [33].

3.1. Qualitative information

Much of the remainder of this paper is based on qualitative information gathered from the project stakeholders that is supplemented by quantitative data where available. The qualitative information was supplied primarily from two sources. The first source was an informal email prompt sent to all members of the project management team in the U.S. as well as the lead developer in the Czech Republic. The prompt is as follows:

> *One thing that I have not heard much about are the problems with the project. So, what do you feel were the biggest hurdles, obstacles, shortcomings, or problems throughout the F15 project? Be as specific or as abstract as you like.*

The second main qualitative resource was interviews. Interviewing can be used to collect historical data from the memories of interviewees or to collect opinions or impressions [53]. An interview was conducted with the actual customer and the proxy customer after the system had been in production for three months. The interview questions may be found in Appendix A. A similar interview was conducted with the development manager. A final follow-up interview was conducted with the customer approximately six months after the system had been put into production.

3.2. Quantitative information

Our quantitative data came from a variety of sources. A simple script was used to count the number of source lines of code (thousand lines of executable code or KLOEC), classes, and methods via naming conventions in the code. Measurements of effort were gathered from the XPlanner[1] project tracking tool and analyzed using SPSS[2]. The classification and quantification of the email communications between developers and project management personnel described in Section 5.3 were gathered through a multi-hour session involving the lead author, the development manager, the project manager, and the project tracker. The collection of the team and process factors and the project outcome described in Section 4 was structured using XP-EF V1.4 [55].

4. Team and project description

This section describes the Tekelec-Radiante team and the project in our case study according to the categories of the XP-EF. From this point forward, we refer to the project as the F-15 project. The XP-EF details of the F-15 project that are not described in this section may be found in Appendix B. We also use the term "team" in this paper to refer to all of the project personnel in both the USA and the Czech Republic. We do, at times, refer to the teams individually, but a non-qualified use of the term implies the entire group. This is in accordance with the way in which the team viewed itself: not as two separate entities working at a great distance apart, but as one group striving to achieve a common goal. The customer noted that cooperation and teamwork were the biggest success factors in this project, and that the level of commitment was extraordinary.

4.1. Team factors

We begin by describing the development team, located in the Czech Republic, and project management team, located in the United States. The number of developers varied during the course of the project. The team started with four developers, then increased to seven as the delivery date approached. After delivery, the team size decreased to two developers who mainly served in a bug-fixing role with limited new feature development. The low domain expertise at the start of the project indicates the significant learning curve faced by the developers with regards to the signaling technology. While the developers had experience with telecommunications application development, they were required to learn many new and proprietary interfaces for the F-15 system. The customer for the F-15 project was a representative from the customer service department whose task it was to train new customers on the use of Tekelec systems. The project management team was comprised of seven individuals: the actual customer, two proxy customers (when the actual customer was unavailable), the project manager, the project tracker, and the development manager. The team members and roles that they play are summarized in Table 1 to provide a reference point for the rest of the paper.

[1] http://www.xplanner.org
[2] http://www.spss.com

Table 1: Personnel and Team Roles

Team Member	Role	Team	Affiliation	Location
Project manager	Oversee the F-15 team and ensure that development was completed on schedule.	Project Management	Tekelec	USA
Actual customer	The customer who would be using the system in production.		Tekelec	USA
Proxy customers (2)	Individuals familiar with the system that could make decisions on project scope and answer technical questions when the actual customer was unavailable.		Tekelec	USA
Project tracker	Responsible for maintaining and monitoring information in the project management tool.	Project Management	Tekelec	USA
Development manager	Technical and cultural liaison between the development group and the project management team. Served as the development coordinator.	Project Management	Radiante	USA
Development lead	Development head who also served as a development representative during planning meetings.	Development	Radiante	Czech Rep.
Developers	Implemented the product features and recorded progress daily.	Development	Radiante	Czech Rep.

A mailing list allowed individual developers to communicate directly with customers to answer questions regarding feature specifications. All members of the development and project management teams were included on the mailing list. The developers also held short status meetings (in the form of "stand up meetings" or Scrum meetings [52]) with the development manager using whiteboard software where voice communication was either through speakerphone or Voice Over IP teleconferencing software. The meetings lasted from 15-30 minutes and involved each developer briefly recounting of what he did on the previous day, any problems he encountered, and what he expected to do the current day. This meeting also served as a forum for the developers to discuss any important issues that needed to be resolved with the project management team. The development manager could also use this time to explain design details or project planning details that the developers needed to know.

4.2. Process factors

This section describes the technologies used during the software development process. A more complete description of the team's XP process usage can be found in Appendix B. The technological factors described in Table 2 give an overview of the software development approach taken during this project.

Table 2: Technological factors

Context factor	F-15 project
Software Development Methodology	XP
Project Management	Planning game/XPlanner; e-mail
Defect Prevention & Removal Practices	Automated unit testing. Code review on code that was not pair programmed. Some acceptance testing executed by the developers prior to delivery.
External/System Test	Actual customer performed regression tests on every deliverable.
Language	Python
Reusable Materials	Unified desktop. Same programs used by all developers. PyUnit test suite with custom GUI.

The XPlanner project management tool contained user stories and a mechanism for tracking the amount of effort developers spent on each user story. This information was used to analyze project velocity and determine the number of features that should be scheduled for each iteration. The XPlanner tool was accessible by the entire team, including customers. Developers used the tool on a daily basis to update the time spent working on user stories. During iteration planning meetings, the discussion centered on the information contained in the XPlanner tool and this information was updated in XPlanner by the project tracker during the call. He would then prompt the

development lead (via speakerphone) to reload the current page whenever a change had been made. While content change tracking was not available in XPlanner, user stories could be tracked if they were continued in subsequent iterations.

4.3. Project factors

This section describes the characteristics of the project itself to help generate a better understanding of the size and scope of the system. This information is summarized in Table 3.

Table 3: Project-specific factors

Context factor	**F-15 project**
Delivered User Stories	65
Domain	Industry-specific technology (product simulator for training)
Person Months	7.62
Elapsed Months	5.7
Nature of Project	New product development
Relative Feature Complexity	Moderate-High
Product Age	0 years (new project)
Constraints	Fixed-delivery date and fixed-budget (limited the number of people). Had to support remote delivery of the software.
New Classes	163
New Methods	817
Unit Test KLOEC	8.242
New Source KLOEC	9.261
Component KLOEC	9.261
System KLOEC	12.931

The project team delivered 65 user stories in the final product. There is no standard size for user stories, and the actual amount of work involved in each user story varies greatly. For example, one user story in this project involved creating a graphical representation of hardware components took 45 hours, while another story involved validating an inputted command took six hours. The amount of effort spent on the project totaled 7.62 person months, however this effort was not evenly distributed. Figure 1 shows the actual effort distribution through the project lifecycle, as obtained via the XPlanner data. As might be predicted, the amount of effort increased as the release deadline approached, hit its peak during the final iteration before release, and then dropped as the project entered the maintenance phase.

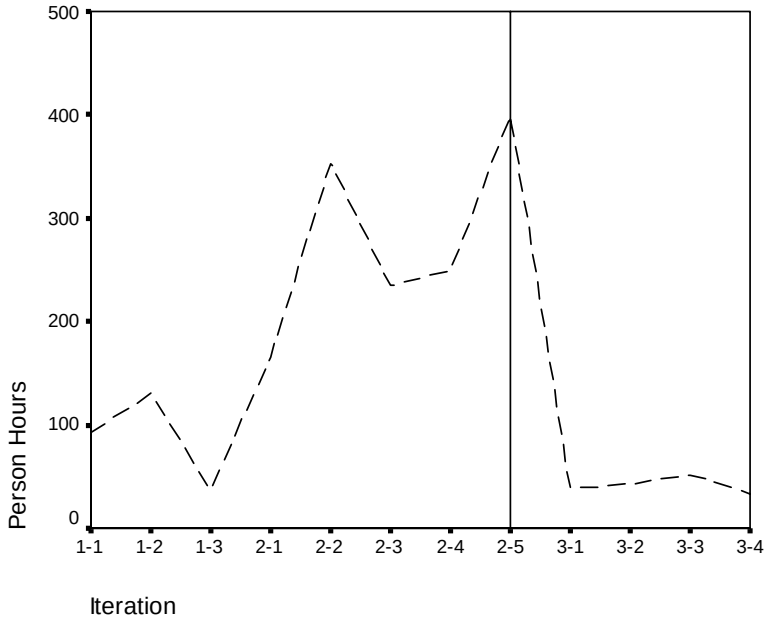

Figure 1. Effort distribution[3]

4.4. Project outcome

This section discusses the F-15 project's XP-om factors that concern the business-oriented results of the project. We also provide information from a customer interview conducted after the project had been in production. The XP-om results are summarized in Table 4.

Table 4: Project outcome (XP-om)

Outcome measure	**F-15 project**
Pre-release Quality (test defects/KLOEC)	N/A
Post-release Quality (post-release defects/KLOEC)	15 / 9.261 = 1.62 defects/KLOEC
Customer Satisfaction (interview)	Capability – Neutral Reliability – Satisfied Communication – Very Satisfied
KLOEC/person month	9.261 / 7.62 = 1.22 KLOEC/PM 17.685 / 7.62 = 2.32 KLOEC/PM (including test code)
User stories/person month	65 / 7.62 = 8.53 stories/PM
Putnam Productivity Parameter [51]	$PPP = (SLOC) / [(Effort/B)^{1/3} * (Time)^{4/3}]$ $(9261) / [(0.635/0.16)^{1/3} * (0.475)^{4/3}] = 15782.72$

We use two measures of quality as related to defects in the system. A pre-release quality measure (a surrogate measure of quality [38]) was not available since records of pre-release testing were not recorded. Just prior to release, the system was tested by the actual customer at a functional/system level, and discovered bugs were recorded as user stories. Post-release quality reflects the number of defects found by the customer once the system is in production. The customer entered bugs he encountered as user stories in XPlanner. The post-release defect

[3] The timeframe is broken up according to phases of development and roughly corresponding to releases. From iteration 1.1-1.3, the team had no defined customer. Iterations 2.1-2.5 represent the time period from when the customer was defined through to the major release point. Iterations 3.1 forward represent the maintenance phase of the project. The reference line at 2-5 represents the project's delivery deadline.

numbers for the F-15 project are lower than published industry averages for new software projects [4]. In an interview, the customer also stated that he was satisfied with the reliability of the product, noting only one fatal crash that was quickly resolved immediately after the product was shipped.

When asked how the customer felt about the capabilities of the product (in terms of features and functionality), the customer was neutral on the subject. The simulator did not cover all the desired aspects of the real system, including the entire command set of the signaling system and error messages that might be encountered while running the system. Due to the rigid deadline on the project, the functionality of the tool was intentionally reduced so that the project could be delivered on time. The productivity of the project was lower than published standards [4]. These numbers may be influenced by the use of a 4th generation programming language as well as extensive testing. When the test code created by the team is included in productivity calculations, the productivity is on par with published standards. However, the customer felt the project progressed very well in such a short amount of time. The customer also stated that the end users of the simulator enjoyed the product.

5. Conjectures

In this section, we present our conjectures about the informal, communication-centric practices of this distributed team. We have identified several factors which we believe were essential in enabling the F-15 development team to deliver a successful project despite the challenges it faced. We provide these conjectures based on the information presented in this paper as well as our own rationale. The type of qualitative inquiry reported in this paper does not produce results that can be generalized, as in quantitative analyses. However, qualitative research can draw implications, propose theories, or promote conjectures for further testing. When writing the conjectures, there are two parts that we address: the conjecture statement and an explanation of the statement with a discussion of related research results available in the literature. Larger quantitative studies can then be designed to evaluate the generalizability of the case study's conjectures. In this section, we propose five conjectures.

5.1. Technical and cultural liaison

> *Conjecture 1: In a DSD team, having a key member of one team physically located with the other team can provide an essential two-way communication conduit.*

The development manager had been working with Tekelec for five years. During this time, he founded Radiante in the Czech Republic, which he also managed. He had hired all of the developers in the project in 2002, with the exception of the development lead that was hired in 1999. The time spent at Tekelec allowed the development manager to become familiar with the company, it technologies, and its needs. Consequently, he had a solid understanding of the system being built, far more so than the remote developers. In working closely with the developers, he was able to translate the functionality of the system and the problems it addressed in a language more familiar to the developers. According to the actual customer:

> *… [the development manager] had a really strong grasp of what [we] wanted. And he was able to effectively convey that to the developers.*

The development manager played an essential role as cultural and technical liaison between the development team and the project management personnel. The language barrier presented challenges early in project development. All of the developers and the development manager spoke English as a second language. None of the project management personnel (other than the development manager) spoke Czech. This language barrier resulted in some difficulties in understanding the technical aspects of the signaling system that the F-15 project would simulate. Since the F-15 developers had no prior knowledge of the signaling system, they had to learn many proprietary technical terms that were difficult to communicate by the project management team. This unfamiliarity with the technical language associated with the signaling system created a learning curve for the developers that made it difficult to understand feature requests early in the lifecycle.

For the developers, the development manager served as a technical liaison for the project management team who could answer technical questions and provide feature details quickly. For the project management team, he represented the developers and intimately understood the status and progress of the project. In many ways, the development manager was the most essential communication conduit, more so than the weekly phone calls and daily email messages. With a thorough understanding of both development and project management needs, he was able to guide the project efficiently. With an identifiable reference for both development and project management personnel to refer to, he significantly narrowed the communication gap caused by technical, distance, and language

barriers. These observations of the essential role of the development manager corroborates the existing evidence on the role of "expert designer" in software development [15] as well as of the cultural and technical liaison in global software projects [12, 40].

However, having such an essential team member is also as risk that he or she become unavailable during the project. This risks is not unique to DSD. A term often used to discuss the presence of such as key individual is "truck number" [14]. A truck number is the number of people who possess unique critical expertise. A project should not become overly dependent upon any small number of individuals [14].

5.2. A definitive customer role

> *Conjecture 2: In a DSD team, a well-defined customer authority is essential for effective decision making and a clear requirements statement.*

The customer role and its associated responsibilities proved to be the greatest source of difficulty with the new software process. The "on-site customer practice" of XP states than an active customer be continuously available, ideally physically present with the developers, to answer developer questions, to make planning decisions, and to provide customer acceptance test cases (CATs). The development manager, project manager, and development lead all pointed out that the customer role was ill-defined for the first month of the six month development effort. Project management personnel were providing the developers with high level goals, but there was no external customer with a defined purpose and usage for the system being build. The ill-defined requirements were also influenced by the team's first-time use of the XP process and unfamiliarity with how to write effective user stories. The F-15 project was to be a simulator for the entire signaling system, a "grandiose" goal according to the project tracker. The following example user story from early in the project that illustrates the high level and poorly-scoped nature of the user stories:

> *The simulator must recognize every command specified in the Eagle 31.0 command handbook, and be able to distinguish between mandatory and optional arguments. An appropriate rejection message shall be printed if the command syntax is incorrect.*

User stories with high level goals but ill-defined features pervaded the early stages of development. The development lead stated that there was "no clear goal during the first month" and one of the proxy customers pointed out that the lack of upfront defined project scope caused problems. User stories must contain enough information to provide some bounds of the feature's scope so that the user story can be broken down into several, readily-identifiable concrete tasks during the iteration planning meeting. Also, customers are required to provide customer acceptance tests (CATs) up-front to validate the user stories and guide development. Yet, during the first month of development, neither clear user stories nor CATs were provided. One of the proxy customers noted that the project being developed was not a usable application, so there was little interest in writing acceptance tests. With poorly scoped user stories and no CATs, the developers had no clear definition of what the system should do and the early releases became ignored by those playing the customer role.

These problems continued for one month until a new stakeholder joined the team who became the project's actual customer. The actual customer had a strong interest in creating the system for a particular purpose. The goal of the project became to create a simulator that would be used to teach a class on using the signaling system. Furthermore, an immovable delivery deadline was placed on the project. A clear goal now enabled the creation of well-scoped and concise user stories based upon the course description available in training materials and upon consultation with a highly motivated customer. The project manager noted that the team's project's lack of focus improved greatly with the new customer involved. The language of the user stories also became more specific:

> *The instructor shall be able to cause SLTC failure on link. The SLTC failure shall be reported every 30 seconds. REPT-STAT-CARD shall show loopback.*

The actual customer remarked that he began receiving prototypes two months into the project and ran tests against them to give feedback to the developers. As the project progressed, CATs became more commonplace. The customer and proxy customer wrote automated tests that were manually invoked to test system features. Furthermore, the developers now had an identifiable person who could answer their questions about user story details and implementation specifics. When the actual customer was unavailable for these questions (because of travel or other commitments), the proxy customers could refer to documentation about the content of the training class.

It is unclear if any of the initial lack of a defined customer are symptomatic or amplified by the distributed development. In general, these problems may be the result of adopting a new software process that was unfamiliar to all parties involved. Similar difficulties with the customer role have been observed in other XP studies of co-located teams [42, 43]. However, the unanimous agreement among all stakeholders that the project was drastically improved once the customer role became well-defined is noteworthy.

5.3. Short, asynchronous communication loops

> *Conjecture 3: In a DSD team, prompt responses to asynchronous queries positively impact development commitment and confidence and create a focused development environment.*

One problem area that is unquestionably influenced by DSD is intra-team communication [29]. Many of the communication difficulties experienced during the project can be attributed to an ill-defined customer role early in the project, as described in Section 5.2. Another significant portion of the difficulties resulted from the fact that the development team was six time zones ahead of the project management team, providing only two hours of overlap where both teams could communicate synchronously. This is particularly troublesome for the XP process, which is built around informal, face-to-face communication. The need for highly interactive communication between customer and developers is due in large part to the informal nature of XP user stories

During the first month of the project, communication between the development team and the project management team was limited. The project management personnel and the development lead took part in release and iteration planning meetings. These meetings were held in a conference room with the project management team with the development lead in the Czech Republic on a speakerphone. During these meetings, user stories were defined and added to the coming iteration to be worked on by the developers. One shortcoming of this approach is that it involved only one perspective from the development team, that of the development lead. Other developers were involved with the call only if they had a specific question for a member of the project management team; developers were typically not involved with the decision-making process. Another obstacle noted by the tracker was that some of the discussions were held around a whiteboard the development lead could not see. XP advocates these informal, whiteboard design meetings, but the information could not be shared with the development team. The other primary form of synchronous communication between the two remote teams were daily status meetings between the development manager and the development team, described in Section 4.1.

The daily status meetings and iteration/release planning meetings were not sufficient to field all of the developer's questions, however. In place of the informal face-to-face communication common in XP, the team used an informal e-mail approach. Approximately one month into the project, the F-15 team began to use a mailing list as their primary form of communication. All messages sent to the mailing list were distributed to all members of the project management and development teams. Typically, the developers would build a backlog of queries for the project management team during their workday in the Czech Republic. Whenever project management personnel arrived at work in the morning in the US, they would address the developers' questions. Therefore, the developers received answers to their questions when they returned for the next day's work. The e-mail exchanges took a very binary form of Question and Answer.

The prompt response to questions was essential in building trust that questions will be promptly answered. The customer resource must be accessible, responsive, and able to make conclusive decisions about project scope and functionality in order to assure a timely response to developer inquiries. A timely response will prevent development from slowing or progressing down an undesired path while awaiting a definitive answer. Furthermore, the customer must be committed to the development process and be actively committed to fulfilling his or her role in that process.

We identified messages concerning user story clarifications and specification, acceptance tests, bug identifications, planning and scope definitions, prototypes of layouts, and more. Figure 3 provides the breakdown of the various types of e-mails according to iteration. This chart provides many insights into the progression of the project.

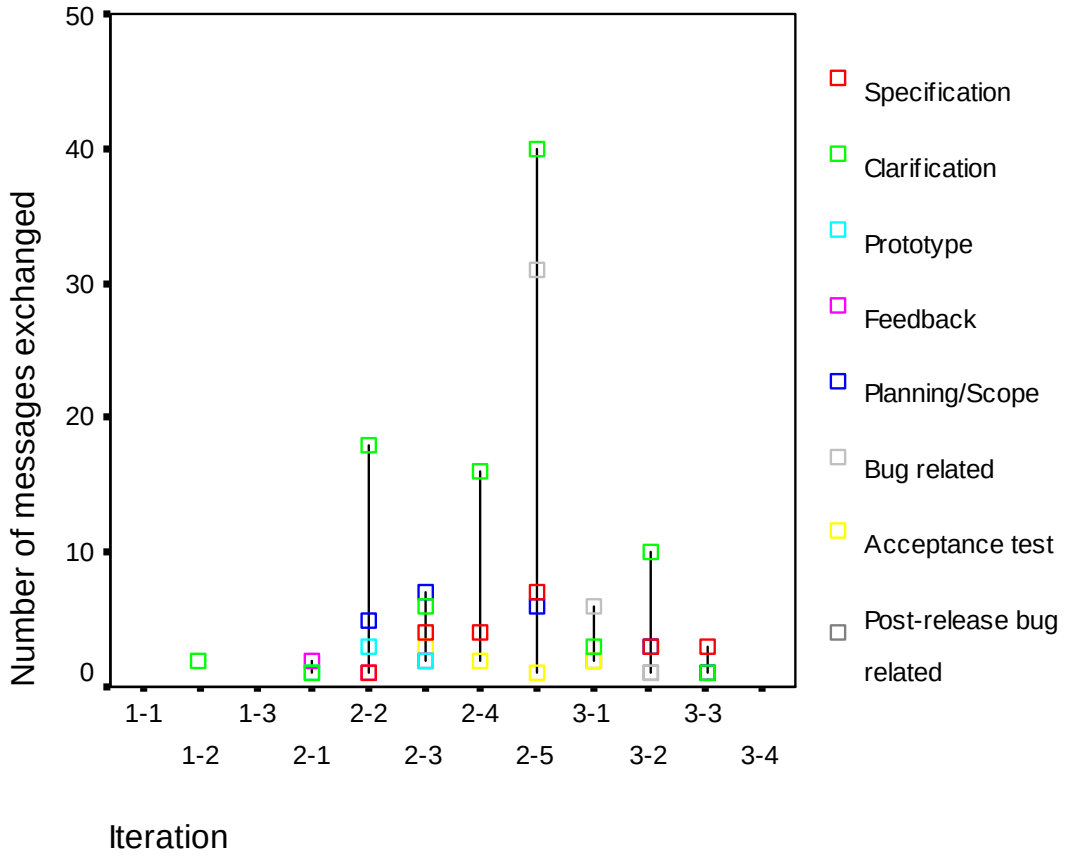

Figure 3. E-mail Message Type Frequencies

Iteration 2-1 marks the beginning of widespread use of the mailing list, and also marks the point in which the customer role became well-defined in the project. In subsequent iterations, the number of messages increases significantly. We note that this chart only encompasses messages sent to the mailing list, and not between individuals. The number of clarification-related messages dominates the list as developers uncovered undefined aspects of the features they were implementing.

We also note that there are a number of specification-related messages, which correspond to the customer defining new or changing existing user stories during the iterations. Planning and scope related messages deal with adding or removing functionality from user stories. These messages suggest that the customer was actively involved throughout the process, and influenced the day-to-day activities of the developers.

At iteration 2-5 (the release point), the communication flow increases significantly. Here, also, we see bug-related messages begin to appear as the actual customer was more rigorously testing the product before delivery. The bug-related messages also appear at approximately the same time as acceptance testing messages. After the release point, we see some post-release bug messages, as well as specification and clarification messages, but on a much smaller scale.

This e-mail analysis provides us with two important observations. First, the specification and scoping messages indicate that the customer was influencing the process on a daily basis. Second, that communication between developers and customers was occurring on a regular basis and not just at critical points during the lifecycle. This seems to suggest that it is possible for e-mail communication to sufficiently exhibit the characteristics implicit in face-to-face communication in XP. While face-to-face communication may be the richest form of communication [21], email communication may serve as an adequate replacement when face-to-face (or telephone) interaction is impossible or cost-prohibitive. Other research [31] has suggested that the software industry could benefit from a communication network where e-mail is recognized as a communication-rich and productive medium.

5.4. Short iterations

Conjecture 4: In a DSD team, having short iterations is an effective means of understanding the problem domain, obtaining customer feedback, understanding requirements, validating priorities, and directing near-term, future work.

The team used two-week iterations wherein development was planned in detail. At the beginning of each iteration, an iteration planning meeting was held that involved the entire project management team and the development lead. The project management team gave feedback to the developers on the previous iteration's completed work. The customers prioritized the remaining incomplete user stories and selected which ones would be implemented in the coming iteration. The development lead and the development manager then created an estimate of the amount of time it would take to finish a desired user story. User stories were added or removed from the release plan as the system evolved.

In the F-15 project, the developers were unfamiliar with the system their product was meant to simulate. Consequently, there was a significant learning curve at the beginning of the project on the application domain. However as the project proceeded, the developers became more familiar with the problem domain based on customer feedback during iteration meetings, conversations with the development manager, and available documentation. Short iterations and frequent deliveries thus emerge as powerful strategies for requirements management in distributed customer-developer relationships, in catching misunderstandings and validating the developers' understanding of the required functionality early in the project.

Iterative development has been used in software development for decades [7] and is most appropriate for any project in which the scope is ill-defined [10, 11], as was the case early on in the F-15 project. Further, we believe that iterative development applies to distributed software development and to newly-formed teams for an additional reason: possible inexperience in the domain. Often outsourced contractors will be working in a new problem domain, at least initially [18]. Practices of iterative and incremental development, prototyping and frequent interaction with the customers are cited in the requirements engineering literature [9, 11, 32] as main risk management techniques for bridging the communication gap with the customer and effective management of client requirements and expectations. Frequent deliveries are also recommended as a successful collaboration practice and for making the process more visible in inter-organizational software development in which clients and developers are apart [44].

5.5. Process visibility and control

Conjecture 5: In a DSD team, providing the team with continuous access to process and product information can help to improve process control and plan effectiveness.

In XP, the customer is not only in control of the features, but also of scheduling and prioritization. This requires that the customer has high visibility into the process to accurately gauge the project's progress and the evolution of the system's features. In a distributed project, process visibility may become an issue when the project management team is not present to observe first-hand the progress made on the project [44]. Furthermore, without active, face-to-face communication, it may be difficult to get a failing project back on track since project management personnel are not on hand to actively guide the project, encourage team members, and oversee status meetings [44].

In a globally distributed project, a universally accessible knowledge based that documents project status can help in achieving project control through high project. In XPlanner, both customers and developers accessed digital representations of user story cards. These user stories, in turn, contained more concrete tasks. Each task had a corresponding progress bar that indicated how much time has been spent on the task by developers, how much time is estimated to be spent in total, and whether or not the task has been completed. Project management personnel, including the customer, could see how individual features in the system were progressing. The developers updated the information in XPlanner three times per week during the early and middle phases of the project, and updated it on a daily basis late in the project. The XPlanner tool was cited by both the development manager and the customers as being essential to the successful management of the project. A highly visible process stands in contrast to a scenario where the project is black-boxed to an outsourced development group who delivers a product several months later that may or may not be in accordance with the customer's specifications.

One way to interpret the planning effectiveness of a project is through its schedule adherence. The information in XPlanner provided project velocity measurements. These velocity measurements were then used to determine the

amount of work that was planned in coming iteration and placed bounds on the number of user stories the customer could integrate into the next iteration. The team's estimation accuracy (how well their estimates of user stories matched the actual time spent) is shown in Figure 4. In Figure 4, the bars are individual user stories and are grouped according to iteration (a user story that spanned multiple iterations appears in each iteration discretely). Bars above the line represent overestimates, while bars below the line represent underestimates. The scale is somewhat skewed because of the sizes of estimates involved, e.g. an estimate of 6 hours with an actual of 1 hour results in 500% overestimate. There were 52 overestimates, 33 underestimates, and 17 correct estimates out of 102 samples. The cardinality and magnitude of the overestimate errors is much greater than that of the underestimates, indicating a tendency for the group to error on the side of caution when creating their estimates. Underestimates are of greater concern, since they reflect additional work and possibly omitting functionality due to time constraints. The largest magnitude of relative error for a single underestimate was 90%.

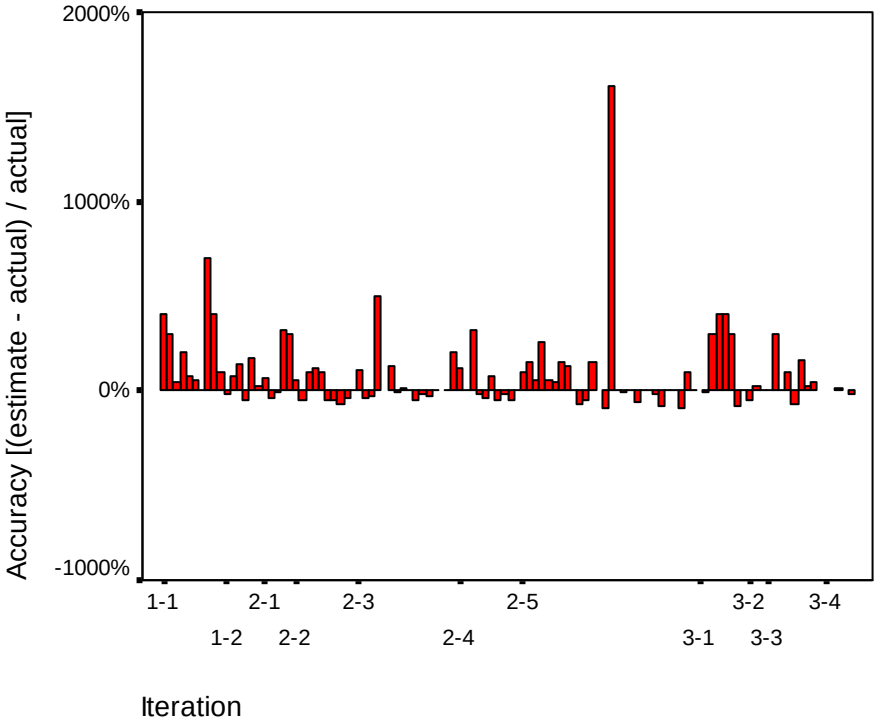

Figure 4. Estimation accuracy of user stories

6. Case study limitations

Yin [57] outlines four tests of judging the quality of research design, including case studies, in producing meaningful results. We enumerate the limitations of our case study using these four test categories:

- **Construct validity**, or establishing measures that reflect the theory under test, can be problematic in case study research. To combat this issue, operational measures for the concept being studied should pre-established, such as with a benchmark like the XP-EF. The XP-EF has been used in multiple case studies [42, 43, 56], and its measurements were constructed using Basili's Goal-Question-Metric approach [5]. The majority of our analysis is reliant upon interviews and questionnaire responses. Published examples for customer satisfaction interviews were used as guides for these artifacts [6, 35]. Additionally, we did not begin with any theories, but let conjectures emerge. Our goal was not to perform rigorous scientific analysis of data, but to build theories from observed phenomena to motivate further study.
- **Internal validity** or the occurrence that all the potential factors that might influence the data are controlled except the one under study. The uncontrollable nature of case studies and the inevitable confounding factors cause internal validity concerns. There is also a concern that the industry participants may behave differently when under observation. The research interest in this project did not begin until the project neared its major

release point. At that time, only a few project management personnel were aware that a study was possible. Data collection commenced several months after the project had ended.

- **External validity** or how well the results of the study can be generalized to the world outside the research situation. External validity is the strength of industrial case studies. However, the results can only be generalized to the context in which the case study was conducted. Potts [49] contends that the real-world complications of a large industrial project are more likely to produce representative problems and phenomena than a laboratory example. He also points out that searching for completely generic findings via case studies is illusory. As such, we provide conjectures only and make recommendations for further research, but draw no definite conclusions. We do believe that this study is beneficial to other projects in which the customer-developer relationship is distributed and where there is a need for frequent interaction.
- **Experimental reliability**, which assesses whether another investigator would get similar results by following the experimental procedures with the same case study. As with construct validity, the use of a benchmark can aid in experimental reliability. The quantitative data in this study can be reconstructed following the procedures in XP-EF v1.4. The action of counting and classifying e-mail listserv messages described in Section 5.3 is partially subjective. No standards were used to define the classification categories, and the classification of individual messages was a decision made by the development manager, project manager, and project tracker based upon high level interpretations of the individual categories. Thus, it is possible that repeating this experiment with the same personnel would yield some different classifications. However, we do not draw upon this data to test a hypothesis or dispute fact, but only to support a conjecture and to help better understand the project under study. Also, the interviews conducted with project stakeholders are likely to yield different results based on the individual style of the interviewer. Any concerns drawn by non-consistency among interview responses can be ameliorated by only discussing themes that were stated by multiple interviewees, such as was done in this paper.

7. Conclusion

We present a case study of a globally distributed software development team with members in the United States and the Czech Republic. We have discussed common issues that arise in distributed development, such as language, communication, and commitment, and how those issues took shape in this project. The team used the XP development methodology, which is oriented around frequent, informal communication between customer and developer. Despite geographical and time distances, the project management and development groups were able to communicate effectively across several time zones. Through the use of an e-mail listserv, globally-available project management tool, and an intermediary development manager who played a strong role in both groups, the team was able to create a successful project that is in production today. Our observations suggest that the challenges of DSD teams do not preclude the use of informal communication-centric practices. In fact, the development manager speculates the team was not successful in spite of the use of these practices, but because of the use of these practices.

We have formulated conjectures of the critical factors that contributed to the team's success. These include the invaluable role played by the development manager, who acted as a communication pipeline between the two groups and played the advocate of both groups on a daily basis. We also note the importance of short, asynchronous communication loops that can serve as a sufficient surrogate for synchronous communication. High process visibility and short iterations were also important for the customers to guide the project effectively while the developers were working on a new and unfamiliar problem.

These conjectures are a first step toward a more formal investigation into using communication-centric methodologies in global software development. Our observations are encouraging, and we believe that, despite barriers of time, language, and distance, the use of informal communication-centric practices can be leveraged to produce successful projects.

Acknowledgements

The authors would like to thank Chet Fennell, Jack Gibson, Mike Kelly, Richard Laurent, and James Young of Tekelec and Ludek Smid of Radiante Corporation for their invaluable comments and for supporting this paper. The North Carolina Center for Computing and Communication (CACC Core Grant 04-06) partially funded this research.

References

[1] P. Abrahamsson, "Extreme Programming: First Results from a Controlled Case Study," 29th IEEE EUROMICRO Conference, Belek, Turkey, 2003, pp. 259-266.
[2] S. Armour and M. Kessler, "USA's new money-saving export: White-collar jobs" in *USA Today*, August 5, 2003.
[3] A. Arora and A. Gambardella, "The Globalization of the Software Industry: Perspectives and Opportunities for Developed and Developing Countries," National Bureau of Economic Research, Washington, DC, 2004.
[4] Bangalore Benchmarking Special Interest Group, "Benchmarking of Software Engineering Practices at High Maturity Organizations," Bangalore Software Process Improvement Network, 2001.
[5] V. Basili, G. Caldiera, and D. H. Rombach, "The Goal Question Metric Paradigm," in *Encyclopedia of Software Engineering*, vol. 2: John Wiley and Sons, Inc., 1994, pp. 528-532.
[6] V. Basili and S. Green, "Software Process Evolution at the SEL," *IEEE Software*, vol. 11(4), pp. 58-66, 1994.
[7] V. Basili, F. Shull, and F. Lanubile, "Building Knowledge Through Families of Experiments," *IEEE Transactions on Software Engineering*, vol. 25(4), pp. 456 - 473, 1999.
[8] V. R. Basili and A. J. Turner, "Iterative Enhancement: A Practical Technique for Software Development," *IEEE Transactions on Software Engineering*, vol. 1(4), pp. 266-270, 1975.
[9] K. Beck, *Extreme Programming Explained: Embrace Change*, Reading, MA: Addison-Wesley, 2000.
[10] B. Boehm and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods," *IEEE Computer*, vol. 36(6), pp. 57-66, 2003.
[11] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, vol. 21(5), pp. 61-72, 1988.
[12] E. Carmel and R. Agarwal, "Tactical Approaches for Alleviating Distance in Global Software Development," *IEEE Software*, vol. 18(2), pp. 22-29, 2001.
[13] A. Cockburn, *Agile Software Development*, Reading, Mass: Addison-Wesley Longman, 2001.
[14] J. O. Coplien and N. B. Harrison, *Organizational Patterns of Agile Software Development*, Upper Saddle River, NJ: Pearson Prentice Hall, 2005.
[15] B. Curtis, "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, vol. 31(11), pp. 1268-1287, 1988.
[16] D. Damian, "Global Software Development: Growing Opportunities, Ongoing Challenges," *Software Process: Improvement and Practice*, vol. 8(4), pp. 179-182, 2003.
[17] D. Damian and D. Zowghi, "Requirements Engineering Challenges in Multi-site Software Development Organizations," *Requirements Engineering*, vol. 8(3), pp. 149-160, 2003.
[18] J. M. Earl, "The Risks of Outsourcing IT," *Sloan Management Review*, vol. 37(3), pp. 26-32, 1996.
[19] C. Ebert and P. D. Neve, "Surviving Global Software Development," *IEEE Software*, vol. 18(2), pp. 62-69, 2001.
[20] K. El-Emam, "Finding Success in Small Software Projects," *Agile Project Management*, vol. 4(11), 2003.
[21] M. El-Shinnawy and L. Markus, "Acceptance of Communication Media in Organizations: Richness or Features?" *IEEE Transactions on Professional Communication*, vol. 41(4), pp. 242-253, 1998.
[22] N. Fenton, S. L. Pfleeger, and R. Glass, "Science and Substance: A Challenge to Software Engineers," *IEEE Software*, vol. 11(4), pp. 86-95, July/August 1994.
[23] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, Boston, Mass: Brooks/Cole Pub Co., 1998.
[24] M. Fowler, "Using an Agile Process with Offshore Development", http://martinfowler.com/articles/agileOffshore.html, last updated: April, 2004, accessed: February 22, 2005.
[25] G. Glaser and L. Anselm, *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Chicagom, IL: Aldine de Gruyter, 1967.
[26] R. E. Grinter, "Recomposition: Putting It All Back Together Again," Conference on Computer Supported Cooperative Work, Seattle, Washington, USA, 1998, pp. 393-403.
[27] W. Harrison, "N=1: an alternative for empirical software engineering research?" *Empirical Software Engineering*, vol. 2(1), pp. 7-10, 1997.
[28] S. Heeks, S. Krishna, B. Nicholsen, and S. Sahay, "Synching or Skinking: Global Software Oursourcing Relationships," *IEEE Software*, vol. 18(2), pp. 54-60, 2001.
[29] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An Empirical Study of Global Software Development: Distance and Speed," International Conference on Software Engineering, Toronto, Ontario, Canada, 2001, pp. 81-90.
[30] J. D. Herbsleb and D. Moitra, "Global Software Development," *IEEE Software*, vol. 18(2), pp. 16-20, 2001.

[31] K. Higa, O. R. L. Sheng, B. Shin, and A. J. Figueredo, "Understanding Relationships Among Teleworkers' E-Mail Usage, E-Mail Richness Perceptions, and E-Mail Productivity Perceptions Under a Software Engineering Environment," *IEEE Transactions on Engineering Management*, vol. 47(2), pp. 163-173, 2000.

[32] R. Jeffries, A. Anderson, and C. Hendrickson, *Extreme Programming Installed*, Upper Saddle River, NJ: Addison Wesley, 2001.

[33] T. D. Jick, "Mixing Qualitative and Quantitative Methods: Triangulation in Action," *Administrative Science Quarterly*, vol. 24(4), pp. 602-611, December 1979.

[34] C. Jones, "Strategies for Managing Requirements Creep," *Computer*, vol. 29(6), pp. 92-94, 1996.

[35] S. Kan, *Metrics and Models in Software Quality Engineering*, Second ed, Boston, MA: Addison Wesley, 2003.

[36] J. King, "IT's Global Itinerary: Offshore Outsourcing is Inevitable," *Computerworld*, vol. 37(37), pp. 26, 2003.

[37] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case Studies for Method and Tool Evaluation," *IEEE Software*, vol. 12(4), pp. 52-62, 1995.

[38] B. Kitchenham, *Software Metrics: Measurement for Software Process Improvement*, Cambridge, MA: Blackwell, 1996.

[39] R. E. Kraut and L. A. Streeter, "Coordination in Software Development," *Communications of the ACM*, vol. 38(3), pp. 69-81, 1995.

[40] S. Krishna, S. Sahay, and G. Walsham, "Managing Cross-cultural Issues in Global Software Outsourcing," *Communications of the ACM*, vol. 47(4), pp. 62-66, 2004.

[41] C. Larman and V. Basili, "Iterative and Incremental Developments: A Brief History," *IEEE Computer*, vol. 36(6), pp. 47-56, 2003.

[42] L. Layman, L. Williams, and L. Cunningham, "Motivations and Measurements in an Agile Case Study," Proceedings of the Workshop on Quantitative Techniques for Agile Processes (QUTE-SWAP '04), Newport Beach, CA, 2004, pp. to appear.

[43] L. Layman, L. Williams, and L. Cunningham, "Exploring Extreme Programming in Context: An Industrial Case Study," 2nd IEEE Agile Development Conference, Salt Lake City, UT, 2004, pp. 32-41.

[44] F. Maurer and S. Martel, "Extreme Programming: Rapid Development for Web-Based Applications," *IEEE Internet Computing*, vol. 6(1), pp. 86-90, 2002.

[45] NASSCOM Report, "The IT Software and Services Industry in India Strategic Review 2001," NASSCOM, 2001.

[46] G. M. Olson and J. S. Olson, "Distance Matters," *Human-Computer Interaction*, vol. 15(2/3), pp. 139-179, 2001.

[47] M. Paasivaara and C. Lassenius, "Collaboration Practices in Global Inter-organizational Software Development Projects," *Software Process: Improvement and Practice*, vol. 8(4), pp. 183-200, 2003.

[48] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "People, Organizations and Process Improvement," *IEEE Software*, vol. 11(4), pp. 69-81, 1994.

[49] C. Potts, "Software Engineering Research Revisited," *IEEE Software*, vol. 10(5), pp. 19-28, 1993.

[50] R. Prikladnicki, J. L. N. Audy, and R. Evaristo, "Global Software Development in Practice: Lessons Learned," *Software Process: Improvement and Practice*, vol. 8(4), pp. 267-281, 2003.

[51] L. H. Putnam and W. Myers, *Measures for Excellence: Reliable Software on Time, Within Budget*, Englewood Cliffs, NJ: Yourdon Press, 1992.

[52] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, Englewood Cliffs, NJ: Prentice Hall, 2001.

[53] C. B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering*, vol. 25(24), pp. 557-572, 1999.

[54] D. Sjøberg, B. Anda, E. Arisholm, T. Dybå, and M. Jørgensen, "Conducting Realistic Experiments in Software Engineering," International Symposium on Empirical Software Engineering, Nara, Japan, 2002, pp.

[55] L. Williams, L. Layman, and W. Krebs, "Extreme Programming Evaluation Framework for Object-Oriented Languages -- Version 1.4," North Carolina State University Department of Computer Science, Raleigh, NC, TR-2004-18, June 17, 2004, 2004.

[56] L. Williams, W. Krebs, L. Layman, and A. Antón, "Toward a Framework for Evaluating Extreme Programming," 8th International Conference on Empirical Assessment in Software Engineering (EASE 04), 2004, pp. 11-20.

[57] R. K. Yin, *Case Study Research: Design and Method*, vol. 5, Third ed, Thousand Oaks, CA: Sage Publications, 2003.

[58] M. V. Zelkowitz and D. R. Wallace, "Experimental Models for Validating Technology," *IEEE Computer*, vol. 31(5), pp. 23-31, 1998.

Appendix A

0) What is your role within the organization?

Project-specific questions

When prompted, the scale for the questions below is:
1. Very Dissatisfied 2. Dissatisfied 3. Neutral 4. Satisfied 5. Very Satisfied

1) What product do you use?

2) What version of this product do you use?

3) On a scale from 1-5, how satisfied are you with the *reliability* of this product?

4) What are the effects of any *reliability* problems in this product? Please comment.

5) On a scale from 1-5, how satisfied are you with the product's *capabilities?* That is, does it meet your needs in terms of features and functionality?

6) In what ways do the product's *capabilities* fail to meet your expectations? Please comment.

7) On a scale from 1-5, how satisfied are you with *communication* with the development organization? Rate this on the basis of communicating with and receiving feedback from the development team or marketing representative. Do not base this comparison on communications with customer support or other avenues.

8) Please describe *who* you interact with at the development organizations, e.g. a marketing representative, project manager, developers themselves. If you interact with more than one contact, please indicate which is the primary contact.

9) On a scale from 1-5, what is your overall satisfaction with the product?

Appendix B

The XP-EF information not detailed in Section 4 may be found here. We begin with the sociological factors of the team. This information is summarized in Table 5.

Table 5: Sociological factors

Context factor	**F-15 project**
Team Size (developers)	2-7
Highest Degree Obtained	Masters: All
Experience Level of Team	<5 years: 7
Domain Expertise	Low-Moderate
Lang. Expertise	Moderate-High

Exp. of Project. Manager	High
Specialist Available	None
Personnel Turnover	300% (4 people joined, 2 left)
Morale Factors	Many public holidays in both the Czech Republic and the US. New hires to be trained. Development team changed offices.

The ergonomic factors discussed in Table 6 include information about the flow of communication between the developers and the project management team. XPlanner was used as the principal project management tool, and contained information for project tracking and work-hour estimations. Iteration and release planning meetings involved the entire project management team and typically one representative from the development team.

Table 6: Ergonomic factors

Context factor	**F-15 project**
Physical Layout	Open office with personal desks
Distraction of Office Space	Changed from moderate to low
Customer Communication	Communicated through XPlanner, mailing list, iteration and release meetings. Access to customer proxy on a daily basis via email.

Table 7 briefly describes geographic factors involved. The developers were not distributed over multiple sites. The development lead and the rest of the development team were located in the Czech Republic. The project manager, project tracker, development manager, and customer were all located in the U.S.

Table 7: Geographical factors

Context factor	**F-15 project**
Team location	Distributed over two continents Raleigh, USA, North America and Brno, Czech Republic, Europe
No. customers	1
Customer location	Remote, six time-zones away
Supplier / Subcontractor	None

Further insight into the F-15 team's development process can be provided by the XP adherence metrics provided in Table 8. The focus of this paper is on the requirements practices; the other information is provided for a broader view of the team's development process.

Table 8: XP adherence metrics (XP-am)

Process metric	**F-15 project**
Planning metrics	
Release Length	4 months
Iteration Length	10 days
Coding metrics	
Pairing Frequency	77.5%

Inspection Frequency	Approx. three or four instances in the whole project
Testing metrics	
Test Coverage	N/A
Test Run Frequency	Regression set: every 8 hours Quickset: once per hour or more
Test LOC / Source LOC	0.91
Classes w/ Corresponding Test Class	100%

The XP adherence metrics in Table 8 help quantify the process that was used during the F-15 project. A practice of XP is short release periods of not more than three months. The F-15 project was under development for four months before a release was made. There was a fixed deadline for this project, as the simulator was to be used for a training session in Egypt on a fixed date. Development continued until the day of delivery. After delivery, product maintenance primarily consisted of bug support and only minor feature development.

Pairing frequency was assessed using the XPlanner project management tool. As developers worked on the user stories, they recorded in XPlanner the amount of time they spent on a particular task and if they paired with another developer during that time. By looking at the total number of hours spent pair programming during the project and the number of hours spent working alone, we were able to estimate the pairing frequency. This number is considered an estimate due to possible inaccuracies in reporting on the part of the developers. Also, these numbers do not reflect time spent collaborating on design. It is interesting to note that the amount of solo hours increased as the delivery deadline approached (see Figure 5), as did the ratio of solo hours to paired hours (see Figure 6). This behavior is consistent with prior studies [42] that indicate that developers will work alone when under deadline pressure, perhaps believing that they work more efficiently by themselves.

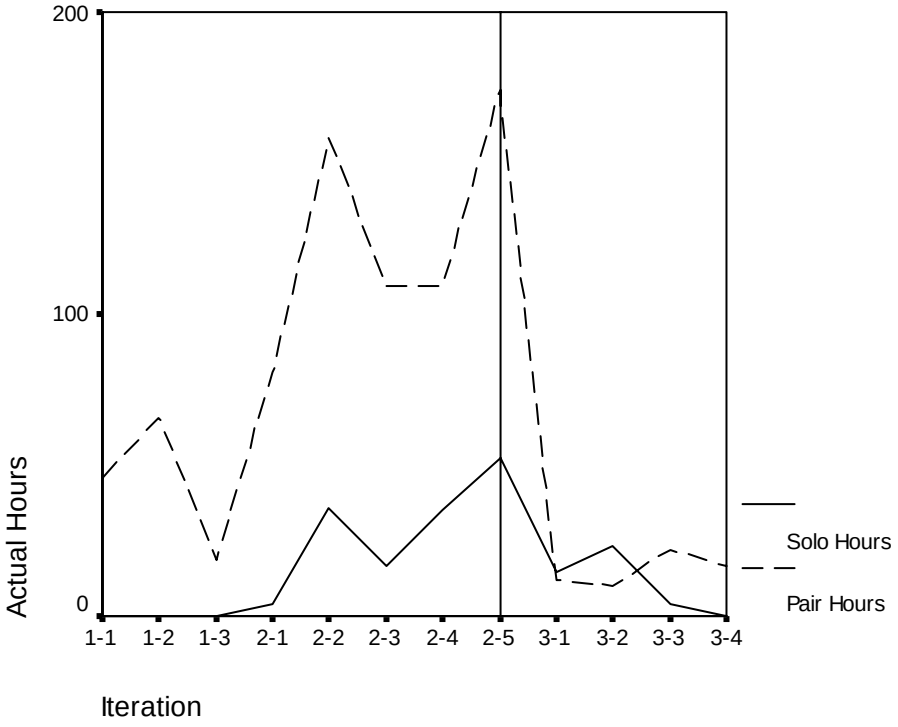

Figure 5. Paired hours vs. solo hours

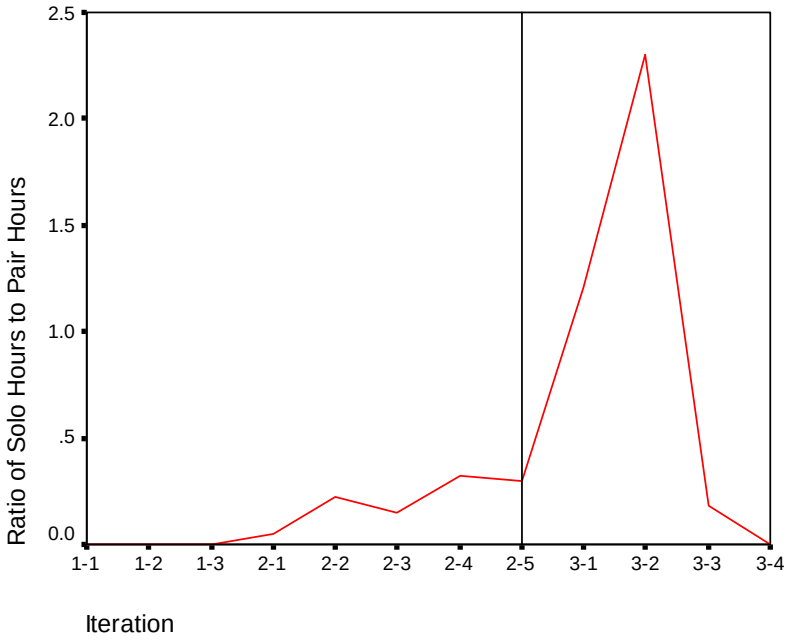

Figure 6. Ratio of solo hours to paired hours.

For testing purposes, the team used the PyUnit unit testing framework for use with the Python language. Every class written in the new system had a corresponding unit test class, and the team generated almost as much test code as source code, as shown in Table 3. Individual developers ran unit tests at least once per hour, while the entire unit test regression suite was run by a build machine every eight hours. The project customers performed some system and functional testing of the system.