Essential Communication Practices for Extreme Programming in a Global 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

We conducted an industrial case study of a distributed team in the USA and the Czech Republic that used Extreme Programming. Our goal was to understand how this globally-distributed team created a successful project in a new problem domain using a methodology that is dependent on informal, face-to-face communication. We collected quantitative and qualitative data and used grounded theory to identify four key factors for communication in globally-distributed XP teams working within a new problem domain. Our study suggests that, if these critical enabling factors are addressed, methodologies dependent on informal communication can be used on global software development projects.

Keywords: global software development, extreme programming, case study

1. Introduction

Distributed software development (DSD) is becoming common practice in today's industry. With DSD, software development teams are not physically co-located and, therefore, cannot see or speak in person on a regular basis. DSD ranges from team members being distributed over adjacent buildings to being distributed over different continents. Global software development (GSD) is the special case of DSD in which the dispersion of the team extends across national boundaries [47]. GSD includes both outsourcing and distributed teams within the same organization that are disbursed in different countries. GSD allows organizations to overcome geographical distances, to benefit from access to a larger resource pool, and to reduce development costs. Over the last decade, outsourcing of information technology (IT) has become a mainstream business phenomenon, and 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 [41], and 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 [2]. GSD has brought about its own unique set of challenges. 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 [13, 26]. These issues seem to preclude the use of processes reliant on informal communication, such as agile [10] methodologies.

Communication, particularly informal communication [24], plays a critical role in the success of a GSD team [9, 19, 23, 25, 40] and, therefore, has been chosen as the central theme of this paper. We describe a GSD team that used Extreme Programming (XP) [7], a process that *requires* an informal, communication-rich environment in order to succeed. Communication is one of the five central values of XP [6] and many of the XP practices explicitly support this value [53]. 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. The project 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 was located in the Czech Republic. On this project, the GSD team used XP for the first time and encountered a considerable amount of requirements volatility. Not surprisingly, XP's reliance on informal communication, as well as the use of an unfamiliar development methodology, presented challenges to the GSD team. The project was successfully completed and put into production after six months of development. The resulting software is currently being used by Tekelec's customer service personnel to train new customers.

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

Email 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 a globally-distributed team using Extreme Programming overcame the hurdles associated with global software development. We present a series of observations derived from both qualitative and quantitative data collected during the course of the project; this data collection was structured via the Extreme Programming Evaluation Framework (XP-EF) [52]. Analysis was conducted of the team's project artifacts, and interviews were conducted with the customer and project personnel. We examined the communication practices the team adopted to create a communication-rich environment in which informal information flowed actively between all project stakeholders. Specifically, the team was able to establish close customer-developer interaction, to address linguistic and technical issues, and to develop a project in an unfamiliar problem domain. Four conjectures concerning the success factors for globally-distributed XP teams emerge from our study. We present conjectures, rather than results, that also serve as challenges for researchers to enhance the community's understanding of the use of informal communication in GSD. Finally, we provide a corresponding list of recommended communication practices for global software development teams using XP.

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 as well as recommendations. Section 6 discusses the limitations of our study, 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 the results of 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 [13, 26]. 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 [12, 21, 32, 43]. Consequently, hurdles in communication will have dramatic effects in the success of GSD projects.

Besides differences in language and culture, GSD teams suffer from a lack of informal communication, resulting in low levels of trust and awareness of work and progress at remote sites. In GSD projects, managing cross-cultural issues is important. Strategies recommended in the literature include the use of cultural liaisons [9], bridgehead teams [33], straddlers (technical or managerial liaisons) [22], or rotation of management to cope with cultural diversity [16]. With bridgehead teams, 75% of the personnel are located offshore while 25% onshore. The onshore personnel are more experienced and culturally assimilated. These individuals work to understand the customer's requirements and translate them to the offshore programmers and to reduce miscommunication between customer and vendor [9]. The development manager role, discussed in Section 5.2, acted a bridgehead.

One class of software development activities directly affected by challenges in communication is 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 [45]. An in-depth field study of requirements engineering in a multinational organization [14] clearly identifies the multitude of factors impacting the effective management of requirements in GSD. 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 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 [14].

Offshore development is a particular case of GSD 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 [14], GSD projects benefit from requirements specifications well-defined at the onset of the project [49], 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 [45]. Frequent deliveries are also recommended as a successful GSD collaboration practice [42]. Incremental integration and frequent deliveries are also recognized as powerful techniques for managing requirements creep [29], and are a core practice of agile methodologies for co-located projects [34]. Though the study of agile practices in GSD has received little attention, there is some evidence that agile practices can be used in offshore development [18]. 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 was designed for use with 10-12 co-located, object-oriented programmers [27]. 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 continuous communication can be a challenge for GSD 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. The customer also provides a corresponding customer acceptance test (CAT) for each user story that, when passed, marks the completion of the user story. Since the initial user story often does not contain the specific details necessary for implementation, 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 case studies

Practitioners and researchers have reported numerous, predominantly anecdotal, studies of XP. There is a lack of industrial case studies of XP in general, and of studies of XP in GSD 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 [38] 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.

The XP-EF [52] is a benchmark for expressing the context of XP case studies, the XP practices an organization has selected to adopt and/or modify, and the business-related results of this adoption. In the XP-EF, researchers and practitioners record essential context information of their project via the XP Context Factors (XP-cf). Recording context factors such as team size, project size, criticality, and staff experience can help explain differences in the results of applying the methodology. The second part of the XP-EF is the XP Adherence Metrics (XP-am). The XP-am use objective and subjective metrics to express concretely and comparatively the extent to which a team utilizes the XP practices. When researchers examine multiple XP-EF case studies, the XP-am also allow researchers to investigate the interactions and dependencies between the XP practices and the extent to which the practices can be separated or eliminated. Part three of the XP-EF is the XP Outcome Measures (XP-om), which enable one to assess the business-related results (productivity, quality, etc.) of using a full or partial set of XP practices.

The XP-EF has been used to structure several XP case studies. In studies conducted with two small, co-located teams (one at IBM and one at Sabre Airline Solutions™), improvements in post-release quality, pre-release quality, and productivity were observed after the teams changed from a more traditional process to XP [35, 52]. Another study with a larger team at Sabre Airlines Solutions™ 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 [36]. 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 little research on the use of XP in a GSD environment where the informal communication required in XP may become strained. Xiaohu, et al. discuss how XP practices can reduce communications issues in a GSD project and provide a limited discussion of an GSD enhancement project that used XP [53].

3. Research method

Our research is an industrial case study of an entire project from initial requirements definition through the end of product development. Case studies are used to collect data through observation of a project in an unmodified, contextual setting [55]. Case studies are powerful techniques for pursuing "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 [54].

The research questions we sought to investigate were to determine what factors allowed GSD XP teams to create successful products. This includes understanding both the everyday mechanics of team process, as well as an evaluation of the final product in terms of quality, productivity, and customer satisfaction. This data was gathered using the XP-EF. Furthermore, we use a qualitative research approach called grounded theory [20] to preserve the complexity of our case study data. The intent is to generate or discover a theory that is "grounded" in data from the field. This approach is suitable for our study since we were not investigating any preconceived theories of what might make a GSD XP team successful or unsuccessful. In this research, we do not provide formal hypothesis testing or draw any general conclusions. However, we conjecture about some communication practices for GSD XP teams based upon evidence gathered from this case study.

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 remaining developers were not contacted due to their limited availability during the case study analysis. Five out of seven recipients responded; responses were received from the project manager, tracker, development manager, development lead, and one of two proxy customers. The responses ranged from one page of prose to three pages of detailed lists. The prompt is:

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

The second main qualitative resource was interviews. A face-to-face interview was conducted with the customer and the proxy customer after the system had been in production for three months. The interview questions may be found in the Appendix. A similar face-to-face interview was conducted with the development manager. Both interviews took approximately 45 minutes. A final, 30 minute follow-up interview was conducted over the phone with the customer approximately six months after the system had been put into production.

We also collected a variety of quantitative data. 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. The data on the team, process factors, and the project outcome described in Section 4 was structured using XP-EF V1.4 [51]. We used both qualitative and quantitative data collection methods in trying to capture a more holistic and contextual portrayal of the factors under study [28]. Convergence and agreement between the results of multiple methods enhances the belief that the results are valid and not a methodological artifact [28].

4. Team and project description

This section describes the Tekelec-Radiante project. 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 [37]. 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 the project management team, located in the USA. The number of developers varied during the course of the project, from four at the beginning, to seven near the delivery deadline, to two during the maintenance phase. The developers had experience in

[1] http://www.xplanner.org

telecommunications development but were required to learn many new and proprietary interfaces for the F-15 system, which created a non-trivial learning curve. 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 team members and their roles are summarized in Table 1 to provide a reference point for the rest of the paper.

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 Mgmt	Tekelec	USA
Customer	The customer who would be using the system in production.	Project Mgmt	Tekelec	USA
Proxy customers (2)	Made decisions on project scope and answered technical questions when the customer was unavailable.	Project Mgmt	Tekelec	USA
Project tracker	Responsible for maintaining and monitoring information in the project management tool.	Project Mgmt	Tekelec	USA
Development manager	Technical and cultural liaison between the development group and the project management team. Served as the development coordinator.	Project Mgmt	Radiante	USA
Development lead	Development head who 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.

4.2. Process factors

This section describes the technologies used during the software development process. A more complete description of the team's use of the XP process can be found in [37]. 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	Iteration and release planning sessions/XPlanner; email
Defect Prevention & Removal Practices	Test-driven development. 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	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 team implemented many tenets of the XP process, while customizing others to fit their needs. The team practiced test-driven development (TDD) and unit tested all code. Over 75% of the work was spent in pairs (as recorded by the XPlanner tool) and followed rigorous coding standards. The developers integrated their code into a build machine at least once per day on which a regression test suite was run every eight hours. Also, the developers practiced collective code ownership, though the majority of work on certain pieces of code was often done by one pair. Only the lead developers took part in planning meetings, which departs from standard XP practice. This was primarily due to scheduling reasons, as the planning meetings took place at the end of the Czech workday and often extended well-after closing hours.

Throughout the project, the project management personnel and the development lead took part in release and iteration planning meetings. During the first month of the project, these meetings served as the primary means of communication between the two groups. The planning 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. The absence of the development team during planning meetings runs contrary to XP process [7]. 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 project management personnel explained system components on a whiteboard, which the development lead could not see. XP advocates these informal, whiteboard meetings, but the information could not be shared with the development team.

The team's primary means of project management was through the XPlanner tool. XPlanner contained electronic versions of user stories and a mechanism for tracking the amount of effort developers spent on each user story. This information was used to analyze the team's velocity, which is the amount of effort (in hours) the team completed in the previous iteration. The velocity was used to determine the number of features that should be scheduled for the next iteration. The XPlanner tool was accessible on-line by the entire team, including customers. Developers used the tool every 1-3 days to update the time spent working on user stories. During iteration planning meetings, the discussion centered on the information contained in XPlanner, which was updated by the project tracker during the meeting. He would then prompt the development lead (via speakerphone) to reload the current page whenever a change had been made. Change tracking was not available in XPlanner. As discussed in greater detail in Section 5.3, the team used also used a mailing list for project management activities. The 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 team also used daily status meetings between the development manager and the development team to discuss progress and technical issues. Maznevski and Chudoba [39] advocate the "regular rhythm" of such meetings for effective distributed teams. During these short status meetings (in the form of "stand up meetings" or Scrum meetings [48]), the team would have voice communication either via speakerphone or Voice Over IP (VOIP) teleconferencing software. Later in the project, these meetings also used whiteboard software so that the development manager and the development team could synchronously view and edit whiteboard contents. The meetings lasted from 15-30 minutes and involved each developer briefly recounting 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 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.

4.3. Project factors

This section describes the characteristics of the project 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. The actual amount of work involved in each user story varied greatly, which is also partially a result of the team's unfamiliarity with XP. For example, one user story in this project involved creating a graphical representation of hardware components and took 45 hours, while another story that involved validating an input 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. 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. The amount of effort varied primarily due to the addition and removal of personnel. The team started with four developers, and 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 feature development. The development company had a small number of developers, and thus was forced to switch personnel between projects as needed to meet imminent customer demands. Iterations were 10 working days in length.

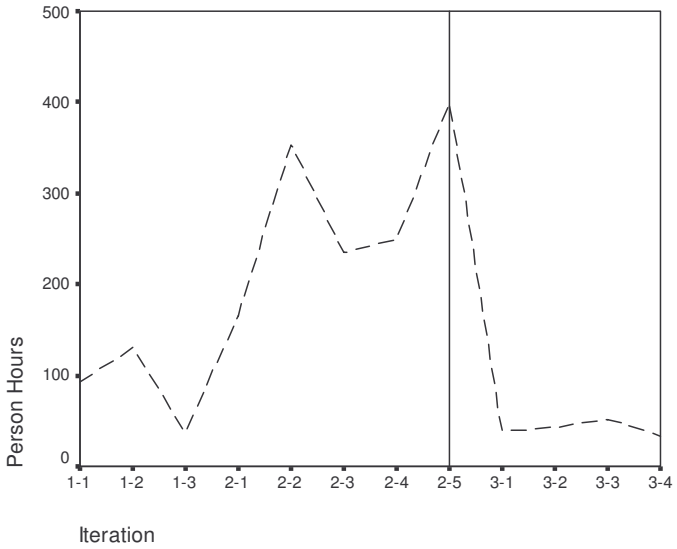

Figure 1. Effort distribution [2]

4.4. Project outcome

This section discusses the F-15 project outcome measures 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 project outcome measures are summarized in Table 4.

Table 4: Project outcome

Outcome measure	**F-15 project**
Pre-release Quality (test defects/KLOEC)	N/A
Post-release Quality (post-release defects/KLOEC)	1.62 defects/KLOEC
Customer Satisfaction (interview)	Capability – Neutral Reliability – Satisfied Communication – Very Satisfied
KLOEC/person month	1.22 KLOEC/PM 2.32 KLOEC/PM (including test code)
User stories/person month	8.53 stories/PM
Putnam Productivity Parameter [46]	15782.72

[2] The timeframe is presented roughly corresponding to releases. From iteration 1-1 through 1-3, the team had no definite customer. Iterations 2-1 through 2-5 represent the time period from when a customer joined the project through final system delivery. Iterations 3-1 forward represent the maintenance phase of the project. The reference line at 2-5 represents the project's delivery deadline.

We use two measures of quality related to defects in the system. A pre-release quality measure (a surrogate measure of quality [31]) was not available since results of pre-release testing were not recorded. Just prior to release, the system was tested by the customer at a functional/system level, and 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 post-release bugs he encountered as user stories in XPlanner. The post-release defect numbers for the F-15 project are lower than published industry averages for new software projects [3]. In an interview, the customer also stated that he was satisfied with the reliability of the product, noting only one fatal crash that was 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. Due to the immovable 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 [3]. The results of this comparison may be influenced by the team's use of a 4th generation programming language as well as extensive unit testing by the developers. When the test code created by the team is included in productivity calculations, the productivity is on par with published standards. We acknowledge that KLOC-based measures of productivity are oftentimes inaccurate and unreliable, but we include them in this study for comparison purposes. The user stories/PM metric (a measure of how many features delivered) and the Putnam Productivity Parameter (a LOC-based metric that controls for project size and duration) can also help understand the team's productivity. 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."

The quality and productivity (including test code) indicate that this project was on par or better than published standards. While the customer would have preferred more capabilities in the product, he acknowledged that the team achieved much in a short amount of time. The developers, the project management personnel, and the customer all considered the project a success. In summary, the GSD XP team was successful in delivering a high-quality product with adequate functionality on time. However, our results indicate the team's productivity may be have been negatively affected by the challenges.

5. Conjectures and recommendations

In this section, we present four conjectures on the enabling communication factors that created a communication-rich environment needed for a successful XP project despite geographical, temporal, and linguistic hurdles. We do not focus on the individual practices of XP and how they were used in this particular project. These conjectures serve as an initial basis for understanding the challenges facing XP GSD teams and are based on grounded theory observations. We base these conjectures on the data that we have collected in our case study as well as our own rationale, but they will bear further substantiation. When writing the conjectures, there are three parts that we address: the conjecture statement; an explanation of the statement with supporting evidence and references to related research results available in the literature; and a recommendation for other distributed teams who wish to use a communication-centric process.

5.1. A definitive customer role for requirements management activities

> *Conjecture 1: In a globally-distributed XP team, a well-defined customer authority is essential for effective decision making and a clear requirements statement.*

In XP methodologies, the customer role is particularly important for requirements management activities. The "on-site customer" practice of XP states than an active customer should be continuously available, physically present (ideally) with the developers to answer developer questions, to make planning decisions, and to provide customer acceptance test cases (CATs). This need for customer contact is compounded by the fact that the development team was working in a new problem domain, a situation typical in GSD [15]. Furthermore, requirements management activities are primarily impacted by distance [14, 45]. Thus, as discussed in this section, the customer role becomes essential for effective requirements management in a distributed XP project to succeed.

The customer role and its associated responsibilities proved to be initially the greatest source of difficulty with the new software process. 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 built. 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 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.

Initially, the F-15 project was to be a simulator for the entire signaling system, a "grandiose" goal according to the project tracker. There was no clear purpose of the usage of the system. From a requirements standpoint, this made it difficult to identify necessary, concrete features for final system implementation. Both the development lead and one of the proxy customers pointed out that the lack of upfront defined project scope caused problems. While user stories are high level and written in natural language, they must contain enough information to provide some bounds of the feature's scope so that the user story can be broken down into readily-identifiable, concrete tasks during the iteration planning meeting [8]. However, user stories with high-level goals but ill-defined features pervaded the early stages of development. Furthermore, XP requires customers to provide 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. The initial requirements difficulties experienced by the team were a result of unfamiliarity with the XP process. The distribution of the project management and development teams compounded this problem, since the developers did not have easy access to a readily-identifiable resource who could resolve any outstanding technical issues.

After this initial troublesome month, a new stakeholder joined the team as the project's customer. The customer had a strong interest in creating the system for a particular purpose. At this point, the goal of the project became to create a simulator that would be used to teach a class on using the signaling system. This also placed an immovable delivery deadline on the project. A clear goal now enabled the creation of well-scoped and concise user stories based upon consultation with a highly-motivated customer and upon the course description available in training materials. 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 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 also wrote scripts 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 customer was unavailable for these questions (due to travel or other commitments), the proxy customers could make decisions with the aid of the training course documentation.

The problems experienced by the F-15 were largely 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 [35, 36]. However, research has shown that distance impacts requirements management activities [14, 45], thus making the customer role in a GSD XP an even more critical component to project success. The impact of the customer role was greatly increased in this project since the team was working in an unfamiliar problem domain, a situation typical of GSD teams [15]. A new problem domain increases the need for active customer involvement as the customer not only has to provide requirements, but may have to spend a significant amount of additional time clarifying technical aspects of the problem.

Recommendation: Define a person to play the role of the customer up front. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.

The contrast between initial difficulties followed by concrete progress solidified the need for a definite customer. A definite customer helps to settle the difficulties of requirements management that are amplified in a globally-distributed project and provides the developers with an identifiable resource to answer questions about features and project scope. This recommendation is similar to what XP requires of all its teams, yet other studies have shown that XP teams can be successful without consistent customer involvement [50] [35]. We believe that active, consistent customer involvement is essential for the success of an XP GSD and cannot be overlooked.

5.2. *Bridgehead*

Conjecture 2: In a globally-distributed XP team, having a key member of one team physically located with the other team can provide an essential two-way communication conduit.

The notion of an individual used to bridge technical, linguistic, and managerial gaps between GSD teams is not novel [9, 22]. However, this person's role is essential when communication between sites is required on a daily basis. In XP, communication between developer and customer is so critical that the use of an "on-site" customer, physically located with the development team, is recommended [7]. In this section, we discuss the development manager, the technical lead on the project who worked closely with both developers and project management personnel. In many ways, he was considered an "on-site customer" for the developers (with regards to technical issues) and an "on-site developer" for the project management personnel.

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 been personally familiar with all of the developers involved in the project since 2002, with the exception of the development lead who he had known for many years prior to that. The time spent at Tekelec allowed the development manager to become familiar with the company, its technologies, and its needs. Consequently, he had a solid understanding of the system being built, far more so than the remote developers. Furthermore, his close ties with the developers allowed him to informally communicate many technical issues to the development team. According to the 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 a technical and cultural liaison between the development team and the project management personnel. The language barrier presented challenges early in project development. 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 [12] as well as of the cultural and technical liaison in global software projects [9, 33]. However, there is risk associated with a project that becomes overly dependent upon any small number of individuals [11].

Recommendation: When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all languages involved.

It is necessary to establish an individual (or possibly a group of individuals) within the team who will straddle both the development and project management realms. This individual is responsible for playing several key roles that emerged as essential in the success of our case study. Ideally, this person is located that group that is not his original organization.

5.3. *Short, asynchronous communication loops*

Conjecture 3: In a globally-distributed XP 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 [25]. Creating effective communication directly between developers and customers is also essential in XP. However, the geographic distances between customers and developers in the F-15 project made face-to-face and telephone communication cost-prohibitive. This section discusses how the team used asynchronous communication (email) to establish communication channels between customer and developer to manage XP user stories.

A significant portion of the project's 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. The team considered using free VOIP technologies, but installing them for all project management personnel was deemed to be too costly for the small amount of time they would be used during the short project. The team also used instant messaging (IM) systems to facilitate some communication between the developers and customers. The IM systems were subject to the same time zone difficulties as telephone conversations, and also were predominantly one-on-one conversations between developer and customer to understand specific technical issues. IM systems were not used when discussing user requirements since the team desired all stakeholders to be aware of such conversations, and the one-on-one nature of instant messaging prevented this information from being distributed to the whole team. This restricted ability to communicate synchronously is particularly troublesome when the XP methodology is used because interactive communication between customer and developers is needed due in large part to the informal nature of XP user stories

The daily status meetings and iteration/release planning meetings (described in Section 4.2) 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 email 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 email 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.

We examined archived data from the F15 team's email listserv to identify messages concerning user story clarifications and specification, acceptance tests, bug identifications, planning and scope definitions, prototypes of layouts, and more. The classification and quantification of the email communications between developers and project management personnel was conducted through a multi-hour session involving the lead author, the development manager, the project manager, and the project tracker. Figure 2 provides the breakdown of the various types of emails according to iteration. Points on the chart represent the frequency of a message type during that iteration, e.g. in iteration 2-5, there were approximately 40 clarification messages and 30 bug-related messages.

This chart provides many insights into the progression of the project. 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 most numerous type of message is a clarification message, which represents developer-customer dialogues that clarify the user stories and elucidate specification details. 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 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.

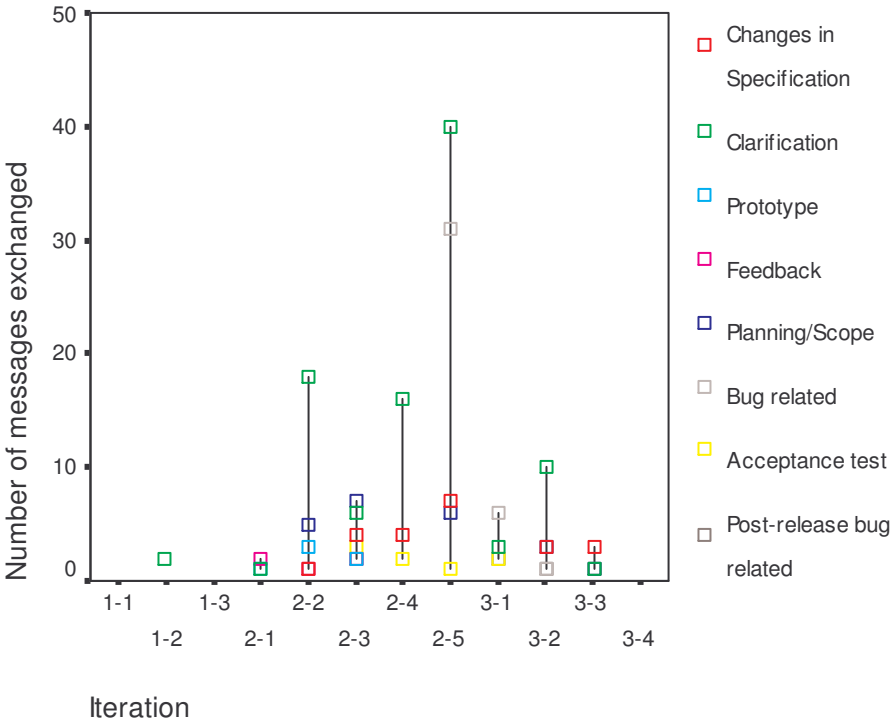

Figure 2. Email message type frequencies

This email analysis provides us with two important insights. First, the change-in-specification and scoping/planning messages indicate that the customer was influencing the process on a daily basis. Second, communication between developers and customers was occurring on a regular basis and not just at critical points during the lifecycle. These observations seem to suggest that it is possible for email communication to sufficiently exhibit the characteristics implicit in face-to-face communication in XP, despite claims by XP authors that face-to-face communication is necessary [7, 8]. While face-to-face communication may be the richest form of communication [17], this data suggests that email communication may serve as a sufficient replacement when face-to-face (or telephone) interaction is cost-prohibitive or restricted due to security concerns or company policy. Furthermore, it has been noted in a previous study of an XP team that email communication between the developers and project management was deemed adequate [52].

Recommendation: When face-to-face, synchronous communication is infeasible, use an email listserv to increase the chance of a response and encourage prompt, useful, and conclusive responses to emails.

A timely response to developer inquiries will prevent development from slowing or progressing down an undesired path while awaiting a definitive answer [43].

5.4. Process visibility and control

Conjecture 4: In a globally-distributed XP team, providing the team with continuous access to process and product information can help to improve process control and plan effectiveness.

This section describes how the F-15 team used the XPlanner tool to effectively manage their project. 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 GSD, process visibility may become an issue when the project management team is not present to observe first-hand the progress made on the project [42]. 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 [42]. In GSD, a universally-accessible knowledge base that documents project status can help in achieving project control through

high project visibility. The use of XPlanner to store electronic versions of user story cards provides visibility into the progress reported per user story.

In XPlanner, both customers and developers accessed electronic versions representations of user story cards (see Figure 3 for an example). 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.

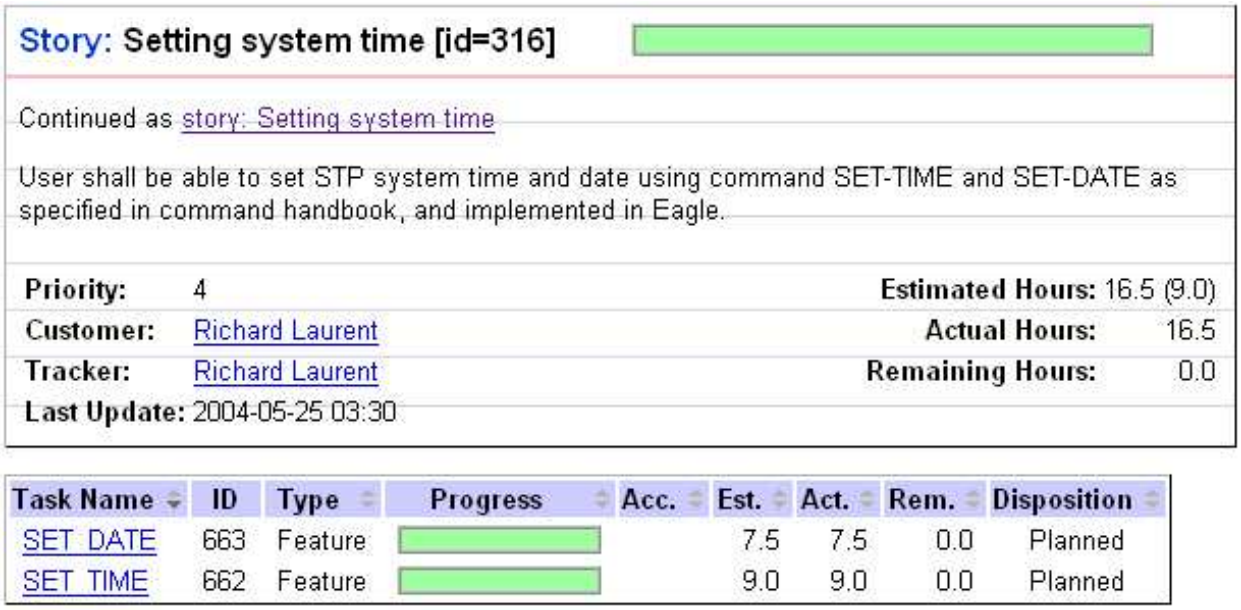

Figure 3. Example of user story from XPlanner

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%. This suggests that, while the team's schedule adherence was not perfect, they avoided continuous underestimates that may have led to unanticipated reductions in project scope at the delivery date. The XPlanner tool was cited by both the development manager and the customers as being essential to the successful management of the project.

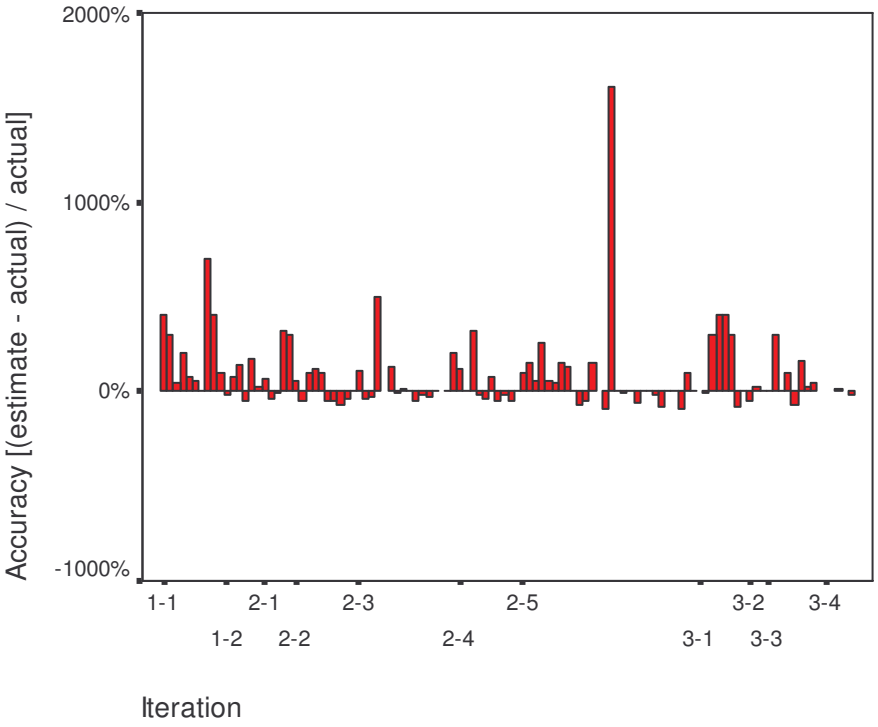

Figure 4. Estimation accuracy of user stories

Recommendation: Use globally-available project management tools to record and monitor the project status on a daily basis.

Using an online project management tool can provide an accurate gauge of project status for all stakeholders. The tool is only as useful as its users make it, so it is essential that the information contained in such a tool is accurate and up-to-date. The recommended use of such a tool is certainly not novel. However, we feel that it is particularly important communication tool in an XP GSD team.

6. Case study limitations

Yin [54] 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 [35, 36, 52], and its measurements were constructed using Basili's Goal-Question-Metric approach [4]. 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 [5, 30]. 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. The personnel turnover during development may also have influenced the
- **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 [44] 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 email 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 counting procedure with the same personnel would yield some different classifications. Also, the interviews conducted with project stakeholders are likely to yield different results based on the individual style of the interviewer.

7. Conclusion

This paper represents a case study of a GSD XP team and the challenges faced by the project management team in the USA and the development team in the Czech Republic. The XP development methodology requires informal communication between customer and developer. Despite geographic, technical, temporal and linguistic hurdles, the team was able to create an environment rich with the informal communication that is essential to XP. Dialogues between customers and developers over feature details became common, changes in user story specifications were made throughout the project, and technical questions were resolved quickly and efficiently. Thus, many of the hallmarks of the face-to-face communication advocated in XP were adequately approximated through the team's use of three essential communication practices: an email listserv, globally-available project management tool, and an intermediary development manager who played a strong role in both groups.

Through a systematic analysis of our findings, we suggest four success factors for globally-distributed XP projects working in a new problem domain. These include the invaluable role played by the development manager, who acted as a communication bridgehead 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. An identifiable customer authority is necessary to manage and resolve requirements-related issues, and high process visibility was important for the customers to guide the project effectively while the developers were working on a new and unfamiliar problem. While some of these conjectures refer to practices that could be beneficial to any software development team, our results indicate these practices are essential for distributed teams employing a communication-rich methodology.

Our findings are a first step toward a more formal investigation into using communication-centric methodologies such as XP in GSD. We hope that others will build upon this research the possible replication of this case study by investigations of development organizations with similar characteristics. 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. We also acknowledge the feedback provided by the NCSU software engineering reading group.

References

[1] P. Abrahamsson, "Extreme Programming: First Results from a Controlled Case Study," presented at 29th IEEE EUROMICRO Conference, Belek, Turkey, 2003.

[2] 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.

[3] Bangalore Benchmarking Special Interest Group, "Benchmarking of Software Engineering Practices at High Maturity Organizations," Bangalore Software Process Improvement Network, 2001.
[4] 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.
[5] V. Basili and S. Green, "Software Process Evolution at the SEL," *IEEE Software*, vol. 11, pp. 58-66, 1994.
[6] K. Beck, *Extreme Programming Explained: Embrace Change*, Second ed. Reading, Mass.: Addison-Wesley, 2005.
[7] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, MA: Addison-Wesley, 2000.
[8] K. Beck and M. Fowler, *Planning Extreme Programming*. Boston, MA: Addison-Wesley, 2001.
[9] E. Carmel and R. Agarwal, "Tactical Approaches for Alleviating Distance in Global Software Development," *IEEE Software*, vol. 18, pp. 22-29, 2001.
[10] A. Cockburn, *Agile Software Development*. Reading, Mass: Addison-Wesley Longman, 2001.
[11] J. O. Coplien and N. B. Harrison, *Organizational Patterns of Agile Software Development*. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.
[12] B. Curtis, "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, vol. 31, pp. 1268-1287, 1988.
[13] D. Damian, "Global Software Development: Growing Opportunities, Ongoing Challenges," *Software Process: Improvement and Practice*, vol. 8, pp. 179-182, 2003.
[14] D. Damian and D. Zowghi, "Requirements Engineering Challenges in Multi-site Software Development Organizations," *Requirements Engineering*, vol. 8, pp. 149-160, 2003.
[15] J. M. Earl, "The Risks of Outsourcing IT," *Sloan Management Review*, vol. 37, pp. 26-32, 1996.
[16] C. Ebert and P. D. Neve, "Surviving Global Software Development," *IEEE Software*, vol. 18, pp. 62-69, 2001.
[17] M. El-Shinnawy and L. Markus, "Acceptance of Communication Media in Organizations: Richness or Features?" *IEEE Transactions on Professional Communication*, vol. 41, pp. 242-253, 1998.
[18] M. Fowler, "Using an Agile Process with Offshore Development," vol. 2005, 2004.
[19] A. French and P. Layzell, "A Study of Communication and Cooperation in Distributed Software Project Teams," presented at International Conference on Software Maintenance, Bethesda, MD, 1998.
[20] G. Glaser and L. Anselm, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicagom, IL: Aldine de Gruyter, 1967.
[21] R. E. Grinter, "Recomposition: Putting It All Back Together Again," presented at Conference on Computer Supported Cooperative Work, Seattle, Washington, USA, 1998.
[22] S. Heeks, S. Krishna, B. Nicholsen, and S. Sahay, "Synching or Skinking: Global Software Oursourcing Relationships," *IEEE Software*, vol. 18, pp. 54-60, 2001.
[23] J. D. Herbsleb, "Hard Problems and Hard Science: On the Practical Limits of Experimentation," *IEEE Computer Society Software Process Newsletter*, pp. 18-20, 1998.
[24] J. D. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering*, vol. 29, pp. 481-494, 2003.
[25] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An Empirical Study of Global Software Development: Distance and Speed," presented at International Conference on Software Engineering, Toronto, Ontario, Canada, 2001.
[26] J. D. Herbsleb and D. Moitra, "Global Software Development," *IEEE Software*, vol. 18, pp. 16-20, 2001.
[27] R. Jeffries, A. Anderson, and C. Hendrickson, *Extreme Programming Installed*. Upper Saddle River, NJ: Addison Wesley, 2001.
[28] T. D. Jick, "Mixing Qualitative and Quantitative Methods: Triangulation in Action," *Administrative Science Quarterly*, vol. 24, pp. 602-611, December 1979.
[29] C. Jones, "Strategies for Managing Requirements Creep," *Computer*, vol. 29, pp. 92-94, 1996.
[30] S. Kan, *Metrics and Models in Software Quality Engineering*, Second ed. Boston, MA: Addison Wesley, 2003.
[31] B. Kitchenham, *Software Metrics: Measurement for Software Process Improvement*. Cambridge, MA: Blackwell, 1996.
[32] R. E. Kraut and L. A. Streeter, "Coordination in Software Development," *Communications of the ACM*, vol. 38, pp. 69-81, 1995.
[33] S. Krishna, S. Sahay, and G. Walsham, "Managing Cross-cultural Issues in Global Software Outsourcing," *Communications of the ACM*, vol. 47, pp. 62-66, 2004.

[34] C. Larman and V. Basili, "Iterative and Incremental Developments: A Brief History," *IEEE Computer*, vol. 36, pp. 47-56, 2003.
[35] L. Layman, L. Williams, and L. Cunningham, "Exploring Extreme Programming in Context: An Industrial Case Study," presented at 2nd IEEE Agile Development Conference, Salt Lake City, UT, 2004.
[36] L. Layman, L. Williams, and L. Cunningham, "Motivations and Measurements in an Agile Case Study," presented at Proceedings of the Workshop on Quantitative Techniques for Agile Processes (QUTE-SWAP '04), Newport Beach, CA, 2004.
[37] L. Layman, L. Williams, D. Damian, and H. Bures, "Conjectures of Informal Communication-Centric Practices Observed in a Distributed Software Development Team," North Carolina State University, Raleigh, NC, TR-2005-13, February 28, 2005.
[38] F. Maurer and S. Martel, "Extreme Programming: Rapid Development for Web-Based Applications," *IEEE Internet Computing*, vol. 6, pp. 86-90, 2002.
[39] M. L. Maznevski and K. Chudoba, "Bridging Space Over Time: Global Virtual Team Dynamics and Effectiveness," *Organizational Science*, vol. 11, pp. 473-492, Sept./Oct. 2000.
[40] A. Mockus and J. Herbsleb, "Challenges of Global Software Development," presented at Seventh International Software Metrics Symposium (METRICS 2001), London, England, 2001.
[41] NASSCOM Report, "The IT Software and Services Industry in India Strategic Review 2001," NASSCOM, 2001.
[42] M. Paasivaara and C. Lassenius, "Collaboration Practices in Global Inter-organizational Software Development Projects," *Software Process: Improvement and Practice*, vol. 8, pp. 183-200, 2003.
[43] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "People, Organizations and Process Improvement," *IEEE Software*, vol. 11, pp. 69-81, 1994.
[44] C. Potts, "Software Engineering Research Revisited," *IEEE Software*, vol. 10, pp. 19-28, 1993.
[45] R. Prikladnicki, J. L. N. Audy, and R. Evaristo, "Global Software Development in Practice: Lessons Learned," *Software Process: Improvement and Practice*, vol. 8, pp. 267-281, 2003.
[46] L. H. Putnam and W. Myers, *Measures for Excellence: Reliable Software on Time, Within Budget*. Englewood Cliffs, NJ: Yourdon Press, 1992.
[47] S. Sahay, "Global Software Alliances: The Challenge of 'Standardization'," *Scandinavian Journal of Information Systems*, vol. 15, pp. 3-21, 2003.
[48] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Englewood Cliffs, NJ: Prentice Hall, 2001.
[49] A. Tiwana, "Beyond the Black Box: Knowledge Overlaps in Software Outsourcing," *IEEE Software*, vol. 21, pp. 51-58, 2004.
[50] L. Williams, W. Krebs, L. Layman, A. Antón, and P. Abrahamsson, "Toward a Framework for Evaluating Extreme Programming," presented at Empirical Assessment in Software Eng. (EASE) 2004, Edinburgh, Scot., 2004.
[51] 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.
[52] L. Williams, W. Krebs, L. Layman, and A. Antón, "Toward a Framework for Evaluating Extreme Programming," presented at 8th International Conference on Empirical Assessment in Software Engineering (EASE 04), 2004.
[53] Y. Xiaohu, X. Bin, and H. Zhijuen, "Extreme Prgoramming in Global Software Development," presented at Canadian Conference on Electrical and Computer Engineering, Niagra Falls, Ontario, Canada, 2004.
[54] R. K. Yin, *Case Study Research: Design and Method*, vol. 5, Third ed. Thousand Oaks, CA: Sage Publications, 2003.
[55] M. V. Zelkowitz and D. R. Wallace, "Experimental Models for Validating Technology," *IEEE Computer*, vol. 31, pp. 23-31, 1998.

Appendix

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?