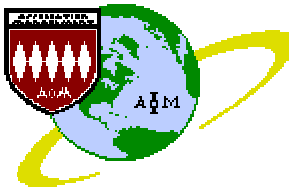

Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

UNDERSTANDING THE ROLE OF SYNCHRONOUS & ASYNCHRONOUS COMMUNICATIONS IN AGILE SOFTWARE DEVELOPMENT AND ITS EFFECTS ON QUALITY

RONZELLE L. GREEN
THE GEORGE WASHINGTON UNIVERSITY, USA
ronzelle@gwu.edu

THOMAS MAZZUCHI
THE GEORGE WASHINGTON UNIVERSITY, USA
mazzu@gwu.edu

SHAHRAM SARKANI
THE GEORGE WASHINGTON UNIVERSITY, USA
sarkani@gwu.edu

ABSTRACT

Through inward perceptions, we intuitively expect distributed software development to increase the risks associated with achieving quality goals. To compound this problem, agile software development (ASD) maintains that face-to-face (synchronous) communication attributed to co-location of the development team is a key success factor. The following study explored the relationship between synchronous and asynchronous communication within ASD teams and its effects on quality. Within two out of four ASD phases (2 out of 6 activities), the analysis supported the opinion that when synchronous communication scores were held constant, asynchronous communication scores predicted quality for distributed teams. In other words, higher asynchronous communication scores resulted in higher quality for distributed teams, but not for co-located teams. Furthermore, the study examined the relationship of distributed ASD teams and challenges encountered through team proximity, frequency of contact, time differences, and language barriers. The analysis identified a positive relationship between proximity, time, and language and asynchronous communication in several ASD phases. Accordingly, distributed teams that are more dispersed, have greater time difference between team members, and use more primary languages to communicate tend to use more asynchronous than synchronous communication techniques. Moreover, ASD teams that communicate more times per day with their dispersed team members tend to use equal amounts of synchronous and asynchronous communication.

Keywords: Agile, distributed software, media richness theory

INTRODUCTION

According to research done by the Standish Group Inc. in 2009, "44% of all projects were challenged (late and overbudget), and/or with less than the required features and functions and 24% failed which are cancelled prior to completion or delivered and never used" [1]. These statistics reflect the state of many software development projects. The Standish Group identified project failure as the measurement of unfavorably meeting three elements: cost, schedule, and performance. One key factor in performance of a software development product is the measurement of quality. The ability of a software development team to produce quality applications requires an understanding of the requirements and consistent communication throughout the software lifecycle. Fred Brooks identified communication as one of the key challenges faced by software engineers [2]. Usually, the inability to accurately communicate and mitigate the effects of unpredictable and changing user requirements and lack of development cohesiveness yield higher costs and defects in software. Therefore, the ultimate goal of ASD is reducing the cost of change (user requirements or other influences) that may engulf a project [3]. Within software development, the strength of agile is the ability to mitigate change. Highsmith and Cockburn note that "teams can be more effective in responding to changes if it can reduce the cost of moving information between people, and reduce the elapsed time between making a decision and understanding the consequences of that decision" [3]. Communication within agile development teams is vital to meeting cost, schedule, and quality goals. Agile development methods recommend co-location of the entire development team. The Agile Manifesto (the cornerstone document of the ASD movement) clearly states that the "most efficient and effective method of conveying information to and within a development team is face-to-face conversation" [4]. In addition, research has shown that having development teams work in the same physical environment improves communication and solidifies clarity [6]. Unambiguous, succinct, and direct communication is important for an ASD team during all activities and phases of development. As corporate entities attempt to benefit from agile and distributed development teams, they must understand the significance of cost effective communication methods within their teams. Furthermore, scholar have identified that the use of geographically dispersed teams continues to grow, outpacing our understanding of their dynamics [5].

AGILE SOFTWARE DEVELOPMENT AND CO-LOCATED TEAMS

In the late 1990s, experts introduced the concept of agile and iterative software development methods. This concept was not completely new, but it readily transformed into a mainstream methodology. Many in the software development community positioned agile methods as a risk mitigation tool to combat the effects of changing user requirements and technology evolution throughout the software development process. The notion of "people over process" reverberated with users and developers alike. A critical element of agile development is iterative software development. Craig Larman, an expert in the field of agile, declares that the iterative approach "allows the user to instantly incorporate feedback into the process to improve functionality" [6]. A very integrated, co-located development team quickly understands and incorporates this feedback into the product. The use of agile development also allows the software developer to adapt to changing and evolving user requirements over the course of the project. Within condensed iterative development efforts, developers incorporate evolving user requirements. The software developers use this feedback mechanism to build optimal information technology (IT) solutions. The team is able to receive, interpret, and execute the desired functionality. This agility helps incorporate new and evolving requirements throughout the process to improve and refine the software product quickly. ASD is the software development community's response to counter unpredictable and changing user requirements through close-knit, co-located teams.

A popular form of ASD is Scrum. Scrum is a "management, enhancement, and maintenance methodology for an existing system or production prototype" [7]. Jeff Sutherland and Ken Schwaber initially developed Scrum. Scrum is an agile, lightweight process used to manage software development processes through iterative and incremental practices. In addition, Scrum provides empirical management and control to manage complex projects using inspection and adaptation to attain the project goals [7]. One of Scrum's guiding principles is to "keep everything visible" and engage everyone in identifying obstacles [8]. Scrum provides a framework that focuses development into "time boxes" usually called sprints. The core practices of Scrum are self-managed teams, sprint planning meetings, backlogs, sprints, daily Scrum meetings, sprint review meetings, and the Scrum-of-Scrums meetings [7].

In general, ASD initiates the idea of co-located development teams and iteration for combating changing and unpredictable requirements. Co-location implies close proximity, face-to-face communication, timely feedback, and informal social interaction [8]. Proximity refers to "the physical distance between people…" [9]. Co-location is one of the key tenants of ASD. Co-location allows teams to react quickly to rapidly changing or ambiguous requirements. Iterative development is the process of building a system within a short period of time [6]. This process of understanding requirements, developing software, and incorporating feedback occurs multiple times until an application meets users' requirements. Usually, during this process, face-to-face, synchronous communication occurs within the development team. Larman also notes another benefit of ASD is reducing the cost of change through precise communication between developers [6].

Unfortunately, ASD has its perceived shortfalls. Opponents of ASD state that it does not scale well in large projects or distributed environments. As Boehm [10] describes, "agile methods are difficult to scale up to large projects because of the lack of sufficient architecture planning, over focusing on early results and low test coverage." Additional research has shown [11] that distributed agile teams face the same pitfalls as many traditional distributed software development teams. These pitfalls include communication shortfalls, culture, and competing organizational norms that add to software project failures.

In the last half of the 20th century, Fritz Bauer initially defined software engineering as "the establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines" [12]. Bauer further explains that software engineering is a detailed, systematic process from requirements to delivery of software. Within the last two decades, the perceptions and expectations of software development have evolved. Companies are expected to increase efficiency while maintaining acceptable costs and software quality. The information age has added the critical element of speed to market to this equation [13]. Combined, these factors present a compelling argument to ensure that the development team clearly understands the requirements to make certain that end users are satisfied and corporations achieve cost effective, quality software deliveries. Unfortunately, some requirement specifications are unplanned or change during the developmental process. This situation introduces vagueness for software developers. In many cases, software developers face the challenge of managing efficiency with the necessity of reworking to correct defects in software. Agile development attempts to respond to this conundrum by understanding, analyzing, and prioritizing new requirements within the development team to produce high quality, defect free software. Over the years, organizations have identified unified processes to mitigate unpredictable and changing user requirements; yet, this has been a struggle for the software development discipline.

Four Phases of Agile Development

The agile development process is an integrated, adaptive system that has an ultimate goal of producing working software in an environment of changing requirements and uncertainty. Below are the four systematic phases of the agile development cycle:

1. **Phase I: Release Planning**
   Key Activities: High-Level Design, Architecture
   Phase I centers around overall product planning, architecture, and high-level design. Phase I, at some level, states all desired final product requirements through user stories.
2. **Phase II: Iteration/Sprint Planning**
   Key Activities: User Stories Analysis and Prioritization
   Phase II primarily focuses on user stories for the sprint. The product owner and development team jointly analyze the sprint backlog and prioritize desired features/functionality for the sprint/iteration. Within the development team, members collaboratively estimate the level of effort necessary to implement the desired features.
3. **Phase III: Scrum**
   Key Activities: Software Design and Code, Integration and Test, Scrum
   Phase III begins the scrum session and includes design, coding, and integration and test for the software project. The scrum sessions are important communication mechanisms for the development team. Outside of the daily scrum sessions, the development team collaboratively communications product design, coding techniques, and integration and test procedures.
4. **Phase IV: Product Release and Retrospective**
   Key Activities: Document preparation, software release, retrospective
   Phase IV is a continuation of Phase III. This segment of the agile lifecycle produces a product demonstration, the final release or increment of the software, some documentation, and a team retrospective. During the retrospective, the development team communications challenges,

successes, and other items to enhance future performance. The expected end of this phase is working software.

To successfully integrate and complete the entire process, the development team must uniquely approach each activity and phase. Especially in distributed environments, varying degrees of communication mediums ensure that the correct messages are sent and received, thus resulting in higher quality and fewer defects in software products. Since requirements, customer demands, and expectations are constantly changing, communication between the development team during each phase is crucial.

Each phase and activity is critical, and concatenated with the other phases encompasses the complete agile development lifecycle. As expected, each phase is linked and must be accomplished sequentially in the iteration/sprint for effective results.

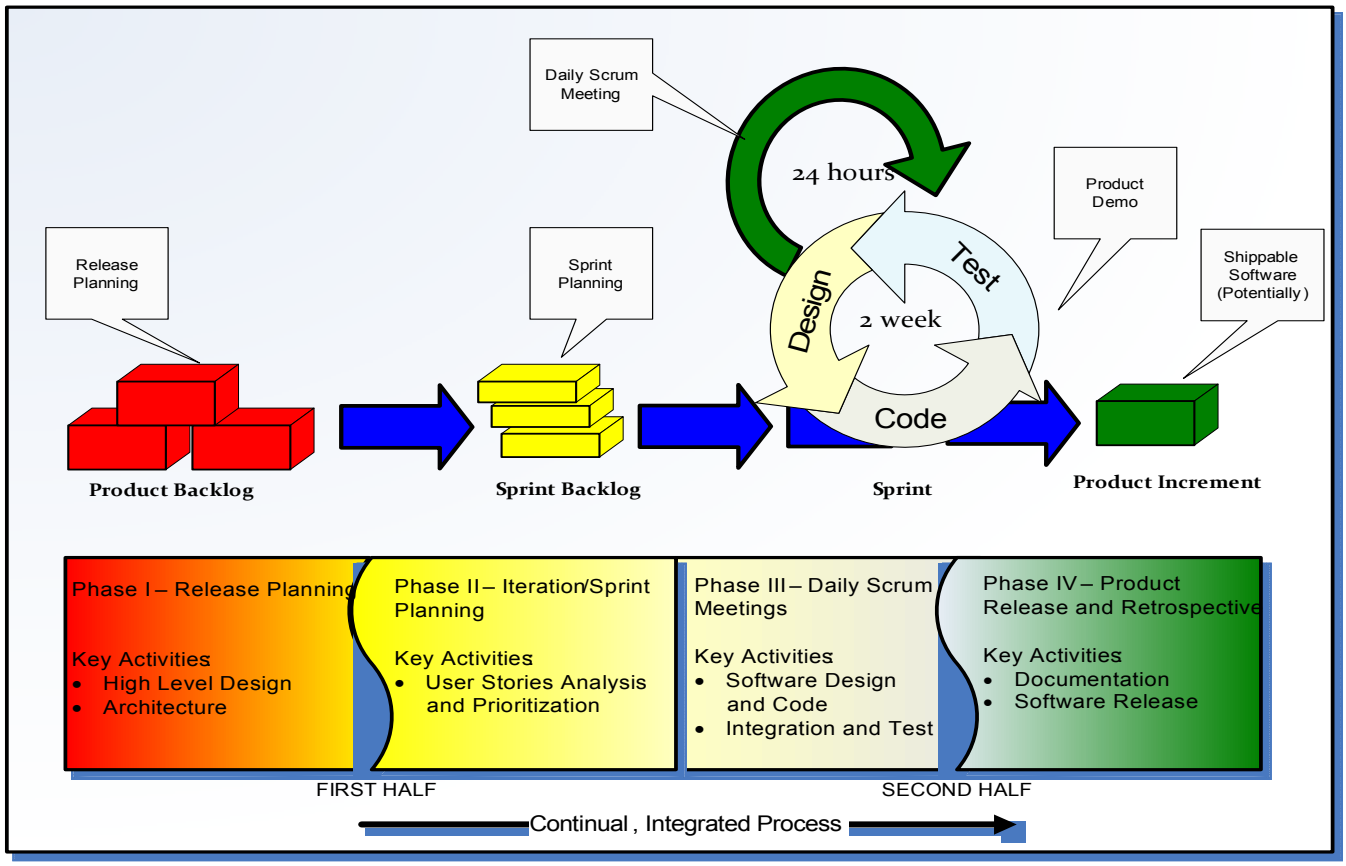

Figure 1: Four Stages of Agile Development adapted from Cohn [28]

Communication Methods and Media Richness Theory

During the eighties, Daft and Lengel produced groundbreaking research introducing media richness theory (MRT). MRT provides a framework for understanding communications requirements and matching those requirements to the capabilities of a given communication medium [14]. MRT categorizes media in a hierarchy of established richness based on feedback; the capacity of the medium to transmit various cues, the use of natural language; and the personal focus of the medium [15]. Furthermore, Daft and Lengel outlined the definition of rich communication. They defined communication rich if it can clear ambiguous and uncertain issues in a timely manner [15]. Moreover, their theory proposed that various forms of communication media possess different ca-

pacities for solving uncertainty and communication ambiguity [15]. Galbraith defined communication ambiguity as the "difference between the amount of communication needed to perform tasks and the amount of information possessed" [16]. In addition, MRT implies that "richer media are more effective for equivocal tasks, and leaner media are better for unequivocal tasks" [16]. Equivocality is the vagueness in tasks caused by unstable and conflicting interpretations, which results in "confusion, disagreement and lack of understanding" [15]. For communication to be efficient and effective in any organization, the richness of the medium used should match the level of message ambiguity [14]. Within this contextual setting, MRT can help evaluate communication media choices. Because of the reduced contextual cues (visual and audio) and less rapid feedback mechanisms, communication media other than face-to-face is considered less rich [14]. The theory suggests that tasks requiring a considerable amount of collaboration require the use of richer media [14], [15]. For these reasons, face-to-face communication has a major advantage over other forms of communication. To achieve the overall goal of reducing ambiguity of communication, one must select the appropriate communication media for the particular task.

Co-located and Distributed Software Development

Co-located and distributed teams face very different communication challenges. Co-located teams are usually in one location and have the ability to send and receive messages within the development team to maximize richness of communication to clear ambiguity. On the other hand, geographically distributed teams face additional obstacles to include [19]:

- Physical distance;
- Overlapping working time;
- Language differences.

Similar obstacles were highlighted by Yadav, et al. in the article on Flexible Global Software Development [20]:

- Control,
- Coordination;
- Communication;
- Culture;
- Technology.

In many cases, these obstacles are significant and are monumental to overcome. Other researchers have documented further challenges [21], [22], [23] of communications within software development project. Carmel identifies five centrifugal forces that have the potential to derail global software projects [23]:

- Loss of communication richness;
- Coordination breakdowns;
- Geographic dispersion;
- Cultural differences;
- Loss of "teamness".

Clearly, these risk factors align closer with distributed than co-located software development. Carmel further explains that distance is the key factor that differentiates distributed and co-located teams [23]. Moreover, he cites that the most "intuitive approach for alleviating distance is to apply communication technologies" [21]. In comparison, Bird's research on the distributed development cycle of Windows Vista, confirms that communication was "the single most referenced problem in globally distributed development" [22]. Proper communication techniques can help alleviate, not eliminate, various challenges faced by distributed software development teams. As expected, approaching distributed software development is quite different from planning activities with co-located teams. Because of various global influences and communication factors, Bird, et al. [22] substantiates that distributed software development is riskier than co-located development. Nevertheless, his investigation of post-release failures developed by distributed verses co-located teams, conclude that there was little difference between each teams' final failure rates for their component of the product. Bird contributed success of the Window Vista project to:

- Positive relationship between distributed sites;
- Overcoming cultural barriers through trust;
- Heavy use of daily synchronous communication;
- Consistent use of similar tools across all teams;
- End to End ownership by all team members;
- Common schedules and deadlines;
- Organizational Integration.

Without a doubt, there are distinct differences between co-located and distributed software development. Yet, if these challenges are properly mitigated, distributed software development can be as efficient as co-located development in achieving quality.

Theoretical Model and Hypotheses

Intrinsically, agile offers a quick response to implicit or explicit change. A provisional element of agile is the ability to provide feedback to the software development team early and often. Research has shown that "projects that performed best were those in which a low-

functionality version of the product was provided to customers at an early stage" [24]. This process limits the cost impact of changing requirements during the software coding and integration phases. This is a progressive approach; yet, developers must coherently understand the new direction and code the software accordingly.

The research hypotheses examined the role of synchronous and asynchronous communication and its influence on quality for ASD teams. The hypotheses fall into two main categories: identifying the linkage between synchronous and asynchronous communication, and quality; and describing the influence of proximity, language, frequency, and time on synchronous and asynchronous communication. The research answers the following questions:

- Can higher levels of asynchronous communication during the agile development lifecycle predict quality for distributed and co-located ASD teams?
- Do proximity, language, frequency, and time have an effect on how distributed agile teams communicate?

The first goal of the research is to determine the effects of synchronous and asynchronous communication of ASD teams and its influence on predicting quality. The hypothesis is as following:

H1: When synchronous communication scores are held constant for co-located and distributed teams, asynchronous communication scores will predict quality such that higher asynchronous communication scores will result in higher quality for distributed teams, but not for co-located teams.

The second goal is examining the effects of proximity, language, time, and frequency on communication methods of distributed ASD teams. The hypotheses are as following:

H2a: For distributed ASD teams, proximity scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

H2b: For distributed ASD teams, communication frequency scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

H2c: For distributed ASD teams, time difference scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

H2d: For distributed ASD teams, primary language scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

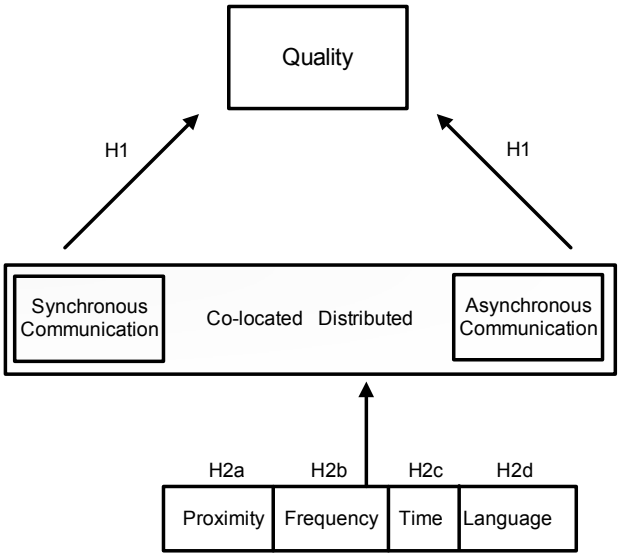

Figure 2: Theoretical Context Diagram

METHODOLOGY

Research Sample

The objective of selecting the research sample is to elicit a national and international representation of the agile development community. As such, an online survey instrument was developed and virtual notifications were sent through diverse national and international email distribution groups. In addition, the survey link was posted on a well-visited agile community resource website. After examination of the data, the research sample achieved a random sample of member from the international and national agile software development community. Over 337 participants started the survey; however, only 60% completed all questions. Partially answered surveys were excluded from the sample. The final sample consisted of 99 (66.4%) co-located agile team members and 50 (33.6%) distributed agile team members.

Data Collection Procedures

Using a 5-point Likert scaling technique, data was collected through a survey with the following requirements:

1. Each question directly related to the research questions and hypotheses;
2. The survey asked short, succinct questions that could be completed within 5-7 minutes;
3. The survey was easily accessible by worldwide

users;
4. The survey contained clear, simple language to be easily understood by either native or non native English speakers;
5. The survey provided participants clear instructions;
6. The survey used common vocabulary understood by novice or expert agile community members.

The survey was open for approximately 3 months and was facilitated by a professional online survey vendor. At the onset of the study, a small group of agile professionals piloted the survey, and feedback was incorporated to update the design. The data characteristics are captured in Table 1.

Table 1: Descriptive Characteristic, Skewness, and Kurtosis for all variables by Communication Techniques

	Communication Technique	*N*	*M*	*STD*	*Skewness*	*Kurtosis*
Quality	Synchronous & Asynchronous	149	2.90	1.04	-0.85	-0.37
Proximity	Synchronous & Asynchronous	50	2.52	0.65	0.40	-0.20
Frequency	Synchronous & Asynchronous	50	1.76	1.12	1.69	2.40
Time	Synchronous & Asynchronous	50	3.02	1.17	-0.37	0.03
Language	Synchronous & Asynchronous	50	2.12	0.85	1.02	1.92
Release Planning	Synchronous	149	2.78	.759	-.263	1.73
	Asynchronous	149	2.61	.952	-.150	-.051
Sprint Planning	Synchronous	149	2.74	.649	.398	.703
	Asynchronous	149	2.50	.971	-.042	-.407
Scrum	Synchronous	149	2.53	.676	-.064	.626
	Asynchronous	149	1.78	.964	.933	.491
Design and Code	Synchronous	149	2.59	.684	-.373	1.43
	Asynchronous	149	2.47	.966	.405	-.484
Integration and Test	Synchronous	149	2.46	.728	-.394	.602
	Asynchronous	149	2.49	.934	.263	-.489
Retrospective	Synchronous	149	2.53	.708	-.659	1.66
	Asynchronous	149	1.77	.994	1.02	.446

Note: N=149

Dependent and Independent Variables

The survey instrument consisted of independent and dependent variables. The independent variables selected are locations and communication techniques. These values were chosen because they were independent of the participants' behavior [25]. Location is composed of distributed and co-located. Communication techniques include synchronous and asynchronous communication between team members throughout the agile development lifecycle. The dependent variable is the observed value that is recording during the research. When stimulated by the independent variable, changes in the dependent variable were observed and measured to determine the level of causality. The dependent variables are quality, proximity, time, frequency, and language.

ANALYSIS AND RESULTS

For Hypothesis 1, multiple linear regression (MLR) analysis were used to compare communication methods with quality for distributed and co-located agile development teams for each phase during the lifecycle. For Hypotheses 2, a combination of Pearson and Spearman Correlation were used to understand the relationship between communication techniques and proximity, language, time, and frequency.

First, each regression model verified the following: 1) normal distribution, 2) a linear relationship between independent and dependent variables, and 3) homoscedasticity of residuals. To confirm linear relationship and homoscedasticity, the studentized residuals were plotted against the standardized predicted scores for the dependent variable to identify a curvilinear patterns or patterns indicative of nonrandomly scattered residuals [26]. Each plot was adequately scattered around zero and appeared to be random.

Each regression model followed the same analysis method. As the first step in computing interaction terms for continuous independent variables, Synchronous and Asynchronous communication scores were mean centered [26]. Team was coded as 0 = Co-located and 1 = Distributed. Next, interaction terms were computed between Team and Synchronous communication scores as well as Team and Asynchronous communication scores. For each regression, the dependent variable (Quality) was regressed on Team, the Synchronous score, the Asynchronous score, the Team × Synchronous score interaction, and the Team × Asynchronous score interaction. The interaction term was added to the model to test the hypothesis that the relationship between asynchronous communication and quality is different for co-located than distributed ASD teams.

Release Planning

For Release Planning, Quality was regressed on Team, the Synchronous release planning communication score, the Asynchronous release planning communication score, the Team × Synchronous release planning score interaction, and the Team × Asynchronous release planning score interaction.

The overall model was not significant, $F(5, 143) = 2.25$, $p < .01$, explaining 7% of the variance in Quality. There was not a significant effect of Team ($\beta = -.10$, $p > .05$), Synchronous release planning communication ($\beta = .22$, $p > .05$), Asynchronous release planning communication ($\beta = -.07$, $p > .05$), Team × Asynchronous release planning communication ($\beta = .15$, $p > .05$), nor Team × Synchronous release planning communication ($\beta = .001$, $p > .05$). Thus, Hypothesis 1 was not supported for the Release Planning activity.

Sprint Planning

For Sprint Planning, Quality was regressed on Team, the Synchronous sprint planning communication score, the Asynchronous sprint planning communication score, the Team × Synchronous sprint planning score interaction, and the Team × Asynchronous sprint planning score interaction.

Results are presented in Table 2. The overall model was significant, $F(5, 143) = 3.99$, $p < .01$, explaining 12.2% of the variance in Quality. There was a significant effect of Team ($\beta = -.74$, $p < .05$), indicating that co-located teams had higher quality when sprint planning communication scores were held constant. There was a significant effect of Synchronous sprint planning communication ($\beta = .39$, $p < .001$), indicating that more synchronous communication resulted in higher quality when Team and Asynchronous sprint planning communication were held constant. There was a significant effect of Asynchronous sprint planning communication ($\beta = -.24$, $p < .05$), indicating that more asynchronous communication resulted in *lower* quality when Team and Synchronous sprint planning communication were held constant. Finally, there was a significant interaction effect between Team and Asynchronous sprint planning communication ($\beta = .84$, $p < .01$), indicating that the relation between Asynchronous sprint planning communication scores and quality was positive for distributed teams, but negative for co-located teams. In other words, for co-located teams, when Synchronous communication was held constant, more Asynchronous sprint planning communication resulted in lower quality, but for distributed teams, when Synchronous communication was held constant, more Asynchronous sprint planning communication resulted in higher quality. The significant interaction is depicted in Figure 3. Thus, Hypothesis 1 was supported for the Sprint Planning phase.

Table 2: Regression Model for Predicting Quality from Team Type, Sprint Panning (SP) Communication, and Interactions

Predictors	*B*	*SE B*	*β*	*t*	*p*	R^2
Team	-1.47	0.57	-.740*	-2.56	.011	.122
SP synchronous	0.36	0.10	.388***	3.56	.000	
SP asynchronous	-0.23	0.11	-.242*	-2.04	.043	
Team × SP synchronous	-0.28	0.18	-.168	-1.51	.134	
Team × SP asynchronous	0.52	0.20	.845**	2.66	.009	

Note. Synchronous and Asynchronous scores were mean centered. Team was coded such that 0 = Co-located and 1 = Distributed. The overall model was significant, $F(5, 143) = 3.99$, $p < .01$. The constant for the model = 2.93.
*$p < .05$. **$p < .01$. ***$p < .001$.

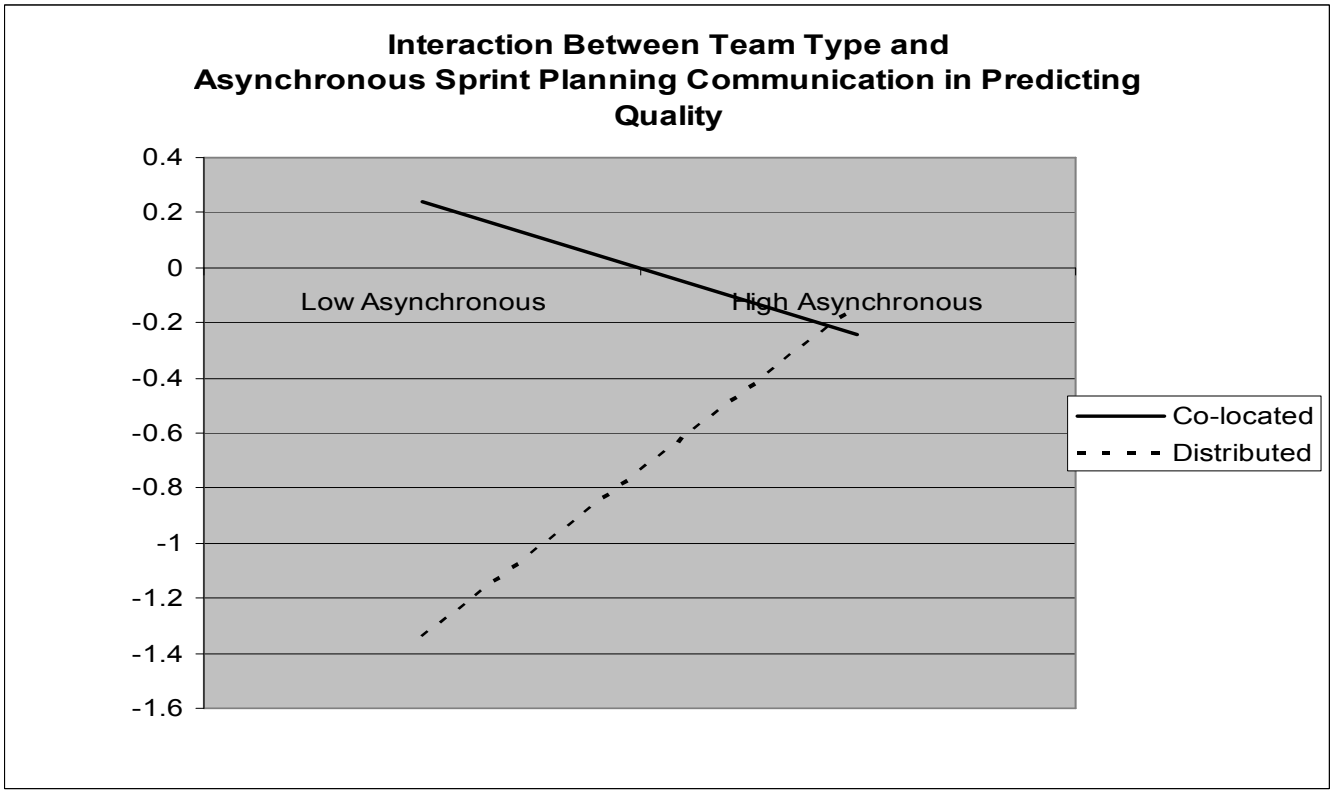

Figure 3: Interaction between Team type and Asynchronous sprint planning communication in predicting quality while holding Synchronous sprint planning communication constant

Agile Communication during Phase III & Phase IV

Scrum Sessions

For Scrum, Quality was regressed on Team, the Synchronous Scrum communication score, the Asynchronous Scrum communication score, the Team × Synchronous Scrum score interaction, and the Team × Asynchronous Scrum score interaction.

Results are presented in Table 3. The overall model was significant, $F(5, 143) = 2.21$, $p < .01$, explaining 7.2% of the variance in Quality. There was a significant interaction effect between Team and Asynchronous Scrum communication ($\beta = .42$, $p < .01$), indicating that the relation between Asynchronous Scrum communication scores and quality was positive for distributed teams, but negative for co-located teams. In other words, for co-located teams, when Synchronous communication was held constant, more Asynchronous Scrum communication resulted in lower quality, but for distributed teams, when Synchronous communication was held constant, more Asynchronous Scrum communication resulted in higher quality. The significant interaction is depicted in Figure 4. Thus, Hypothesis 1 was supported for the Scrum phase.

Table 3: Regression Model for Predicting Quality from Team Type, Scrum Communication, and Interactions

Predictors	*B*	*SE B*	*β*	*t*	*p*	R^2
Team	-.261	.192	-.131	-1.360	.176	.072
Scrum synchronous	.133	.103	.141	1.287	.200	
Scrum asynchronous	-.238	.125	-.253	-1.898	.060	
Team × Scrum synchronous	-.198	.187	-.126	-1.063	.290	
Team × Scrum asynchronous	.587	.197	.423**	2.982	.003	

Note. Synchronous and Asynchronous scores were mean centered. Team was coded such that 0 = Co-located and 1 = Distributed. The overall model was not significant, $F(5, 143) = 2.21$, $p > .05$. The constant for the model = 2.94.
**$p < .01$.

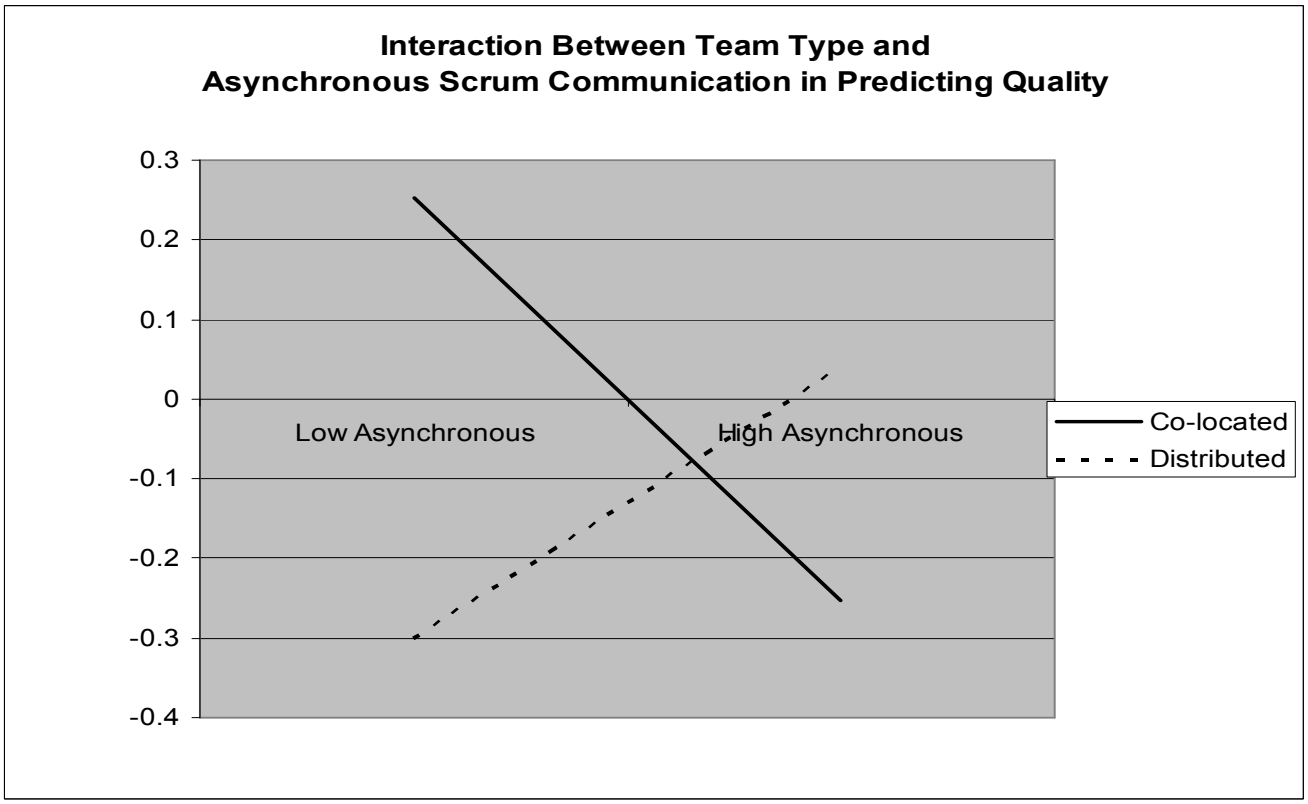

Figure 4: Interaction between Team type and Asynchronous Scrum communication in predicting quality while holding Synchronous sprint planning communication constant

Design & Code

For Design & Code, Quality was regressed on Team, the Synchronous design & code communication score, the Asynchronous design & code communication score, the Team × Synchronous design & code score interaction, and the Team × Asynchronous design & code score interaction.

The overall model was not significant, $F(5, 143) = 1.73$, $p > .05$, explaining 5.7% of the variance in Quality. There was not a significant effect of Team ($\beta = -.13$, $p > .05$), Synchronous design & code communication ($\beta = .09$, $p > .05$), Asynchronous design & code communication ($\beta = -.04$, $p > .05$), Team × Synchronous design & code communication interaction ($\beta = .07$, $p > .05$), nor Team × Asynchronous design & code communication interaction ($\beta = .17$, $p > .05$). Thus, Hypothesis 1 was not supported for the design & code activity.

Test & Integration

For Test & Integration, Quality was regressed on Team, the Synchronous test & integration communication score, the Asynchronous test & integration communication score, the Team × Synchronous test & integration score interaction, and the Team × Asynchronous test & integration score interaction.

The overall model was not significant, $F(5, 143) = 1.58$, $p > .05$, explaining 5.2% of the variance in Quality. There was not a significant effect of Team ($\beta = -.12$, $p > .05$), Synchronous test & integration communication ($\beta = .14$, $p > .05$), Asynchronous test & integration communication ($\beta = -.11$, $p > .05$), Team and Synchronous test & integration communication ($\beta = .00$, $p > .05$), nor Team and Asynchronous test & integration communication ($\beta = .21$, $p > .05$). Thus, Hypothesis 1 was not supported for the test & integration phase.

Retrospectives

For the Retrospective phase, Quality was regressed on Team, the Synchronous retrospective communication score, the Asynchronous retrospective communication score, the Team × Synchronous retrospective score interaction, and the Team × Asynchronous retrospective score interaction.

The overall model was not significant, $F(5, 143) = 1.11$, $p > .05$, explaining 3.7% of the variance in Quality. There was not a significant effect of Team ($\beta = -.09$, $p > .05$), Synchronous retrospective communication ($\beta = .12$, $p > .05$), Asynchronous retrospective communication ($\beta = -.07$, $p > .05$), Team × Synchronous retrospective communication interaction ($\beta = .01$, $p > .05$), nor Team × Asynchronous retrospective communication interaction ($\beta = .15$, $p > .05$). Thus, Hypothesis 1 was not supported for the retrospective activity.

Results Summary for Hypothesis 1

Through the examination of Hypothesis 1, the results supported (2 out of 6 ASD activities) the belief that when synchronous communication scores were held constant for co-located and distributed teams, asynchronous communication scores will predict quality such that higher asynchronous communication scores will result in higher quality for distributed teams, but not for co-located teams. Accordingly, to achieve higher quality goals, distributed ASD teams should use higher forms of asynchronous communication during Sprint Planning and Scrum phases of development. For distributed ASD teams, synchronous communication techniques throughout the agile development process may produce comparable results, however, this approach could be cost prohibitive and may not be as effective to convey richness of information for worldwide disperse teams.

Table 4: Summary of Hypothesis 1

Activities/Phases	*Hypothesis 1*
Release Planning	Not Supported
Sprint Planning	Fully Supported
Scrum	Fully Supported
Design and Code	Not Supported
Integration and Test	Not Supported
Retrospective	Not Supported

Agile Communication techniques for distributed teams

Distributed ASD teams face additional challenges with proximity, frequency of communication with other disperse team members, time differences, and potential language barriers. Moreover, these barriers contribute to poor team performance, increase delivery schedules, and miscommunications that often lead to rework and poor software quality. Below, the variables proximity, frequency, time, and language were examined to investigate the relationship between these variables and asynchronous and synchronous communication for distributed ASD.

Table 5: Pearson Correlations with Dependent Variable (Proximity, Time, Frequency, Language) for Distributed ASD Teams using Asynchronous and Synchronous Communication

	Proximity		*Time*		*Frequency*		*Language*	
	Async	Sync	Async	Sync	Async	Sync	Async	Sync
Release Planning	.098	-.128	.312*	.065	-.028	.159	.175	.057
Sprint Planning	.309*	-.115	.334*	-.148	.019	.151	.077	-.174
Scrum	.189	-.072	.321*	-.107	.165	.190	.267	-.156
Design and Code	.349*	-.121	.471***	.000	.069	.055	.308*	-.024
Integration and Test	.363**	-.027	.500***	.048	.148	.172	.340*	-.110
Retrospective	.156	-.010	.316*	.086	.139	.104	.132	.017

Note. $N = 50$.
*$p < .05$. **$p < .01$. ***$p < .001$.

Proximity

H2a: For distributed teams, proximity scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

Proximity refers to the physical distance that a particular ASD team is from each other. The greater the proximity score, the wider disperse (number of locations) the ASD is located. As explained in the results in Table 4, there are significant positive relations between proximity scores and asynchronous communication in three activities (two phases): Sprint Planning ($r = .309$, $p < .05$), Design & Code ($r = .349$, $p < .05$), and Integration & Test ($r = .363$, $p < .01$). The correlations for the other asynchronous phases were positive, but not significant. Furthermore, the data supported negative correlations between distributed ASD teams that are in numerous locations (higher proximity scores) and synchronous communication (less synchronous communication) although the correlations were not significant. In other words, distributed ASD teams that are in more location across the world tend to use more asynchronous communication over synchronous communication. Thus, Hypothesis 2a was supported for three activities, and the trend was in the right direction for the other three activities.

Time

H2b: For distributed teams, time difference scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

Time refers to the time differences of ASD teams that are widely disperse throughout the world. Greater time scores related directly to the difference in time (hours & minutes) for different project members of distributed teams. As shown in table 4, there are significant positive relationships between time scores and asynchronous communication in all six activities (four phases): Release Planning Sprint ($r = .312$, $p < .05$), Sprint Planning ($r = .334$, $p < .05$), Scrum ($r = .321$, $p < .05$), Design & Code ($r = .471$, $p < .001$), Integration & Test ($r = .500$, $p < .001$), and Retrospective ($r = .316$, $p < .05$). In comparison, very weak nonsignificant relationships were found between time scores and synchronous communication for distributed ASD teams. Thus, distributed ASD teams that have greater time differences are more likely to use asynchronous communication over synchronous communication. Thus, Hypothesis 2b was fully supported.

Frequency

H2c: For distributed teams, communication frequency scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

Frequency refers to the number of information exchanges that ASD team members perform in one day with other members in distributed sites. The greater the frequency number, the more communication exchanges distributed ASD teams had in a particular day. Kurtosis for frequency was 2.6; therefore, the variable showed a slight deviation from normality. Consequently, the Pearson (parametric) and Spearman (nonparametric) correlations were both applied to the data in case the deviation from normality distorted the parametric results. There was no difference between the results. The results in Table 4 identified a positive relationship (except during release planning) between frequency scores and asynchronous communication for distributed teams, but the correlations were small and nonsignificant. The data also showed a positive relationship between frequency and synchronous communication techniques, but these were also very weak and nonsignificant. In other words, for distributed ASD teams, frequency does not appear to be related to either asynchronous or synchronous communication. Thus, Hypothesis 2c was not supported such that frequency was not positively related to asynchronous communication.

Language

H2d: For distributed teams, primary language scores are positively related to asynchronous communication scores, but not related to synchronous communication scores.

Language referred to the number of primary languages an ASD team may use in each locality. As research has shown [11], language barriers can affect the productivity and efficiency of any project, especially agile software development. The greater the language scores, the more primary languages, which are used within various locations. As shown in Table 4, a significant positive relationship was found between language scores and asynchronous communication for: Design & Code ($r = .308$, $p < .05$), and Integration & Test ($r = .340$, $p < .05$). The correlations for the other activities were in the correct direction, although nonsignificant. The data also confirmed weak, negative relationships between language scores and synchronous communication techniques, none of which was significant. In practical terms, distributed ASD teams that have more primary languages tend to us asynchronous communication. Thus, Hypothesis 2d was

partially supported for two activities and the trend was in the correct direction for the other four activities.

Results Summary for Hypotheses 2

After testing Hypothesis 2, the results supported the concept that within distributed ASD teams, proximity, frequency, time, and language scores are positively related to asynchronous communication scores, but not related to synchronous communication scores (except for frequency). Through correlation techniques, the data confirmed a positive relationship between proximity, time, and language and asynchronous communication for several phases. Consequently, distributed teams that are more dispersed, have greater time difference between team members, and use more than one primary languages to communicate tend to use more asynchronous communication techniques. Likewise, ASD teams that communicate more times per day with their dispersed team members tend to use equal amounts of synchronous and asynchronous communication. As discussed in Table 6, (3 out of 4) hypotheses were fully or partially supported.

Table 6: Summary of Hypothesis 2

Variable	*Hypothesis 2*
Proximity	Partially Supported
Frequency	Not supported
Time	Fully Supported
Language	Partially Supported

CONCLUSION

New environmental, organizational pressures, and competitive markets are forcing companies to produce quality software with shorter developmental cycles. As one could expect, these goals could be counterintuitive, however, many corporate entities are leveraging fertile international software engineering resources. These technical professionals are permitting continuous, 24-hour software development cycles at lower human capital costs. Within this new paradigm, software engineers still must mitigate the effects of unstable and changing user requirements to produce quality products. Co-located ASD teams have favorably showed value in responding to dynamic and evolving user requirements that ultimate affect the products' final levels of quality. Thus, in this new dispersed environment, software engineers must be able to succinctly communicate within the development team to achieve the common purpose of quality software, on time deliveries, and within cost projects. Therefore, distributed ASD teams must proficiently translate synchronous communication requirements into appropriate levels of asynchronous communication functionality. Once achieved, distributed ASD teams can consistently enjoy similar successes realized by co-located ASD teams.

REFERENCES

[1] The Standish Group International, Inc. "Extreme chaos", http:// www.vertexlogic.com/process Online/processData/documents/pdf/extreme_chaos.pdf , March 2009.

[2] Brooks, F. "No silver bullet; essence and accidents of software engineering ", *Computer* 20:4, April 1987, 10-1.

[3] Highsmith, J. and Cockburn, A. "Agile software development: the business of innovation," *IEEE Computer*, 2001.

[4] Manifesto for agile software development., http://www.agilemanifesto.org, 2001.

[5] Ramsubbu, N., Krishnan, M., and Kompalli, P. "Leveraging Global Resources: A Process Maturity Framework for Managing Distributed Development. "IEEE Software 22, no. 3, May 1,2005, pp. 80-86.

[6] Larman, C., *Agile & Iterative Development*: *A Manager's* G*uide*. Boston, MA: Addison-Wesley, 2004.

[7] Schwaber, K., *Agile project management with Scrum*. Redmond, WA: Microsoft Press, 2004.

[8] Schwaber, K. and Beedle, M. "Agile software development with Scrum," in Series in agile software development, Upper Saddle River, NJ: Prentice Hall, 2002, pp. xvi-158.

[9] Hinds, P. and Kiesler, S., Eds., *Distributed Work*. Cambridge, MA: Massachusetts Institute of Technology, 2002.

[10] Boehm, B. "Get ready for agile methods, with care," *Computer,* vol. 35, no. 1, pp. 64, Jan. 2002.

[11] Dubé, L. Paré, G. "Global Virtual Teams." *Communications of the ACM* 44, no. 12, December 2001, pp. 71-73.

[12] Pressman, R., *Software Engineering: A Practioner's Approach*, 2nd ed. New York: McGraw-Hill Book Company, 1987.

[13] Baskerville, R., Balasubramaniam, L., Levina, J., Pries-Heje, and S. Slaughter. "Is internet-speed software development different?," *IEEE Software,* vol. 20, no. 6, pp. 70-77, 2003.

[14] Daft, R.. Lengel, R. and Trevino, L. "Message equivocality, media selection, and manager performance: Implications for information systems,"

MIS Quarterly, vol. 11, no. 3, Sept. 1987, pp. 355-366.

[15] Daft, R.. Lengel, R. "Organizational information requirements, media richness and structural design," *Management Science*, vol. 32, no. 5, pp. 554-571, May 1986.

[16] Galbraith, J., *Strategies of Organizational Design,* Reading, MA: Addison–Wesley, 1973.

[17] Barkhi, R., Jacob, V. and H. Pirkul. "An experimental analysis of face to face versus computer mediated communication channels," *Group Decision and Negotiation,* vol. 8, 1999, pp. 325-347.

[18] Vrasidas, C. and McIsaac, M."Principles of pedagogy and evaluation for web-based learning," *Educational Media International,* vol. 37, no. 2, 2000, pp. 105-111.

[19] Sosa, M.E.; Eppinger, S.D.; Pich, M.; McKendrick, D.G.; and Stout, S.K. "Factors that influence technical communication in distributed product development: an empirical study in the telecommunications industry," *Engineering Management, IEEE Transactions on* , vol.49, no.1, Feb 2002, pp.45-58.

[20] Yadav, V.,M.Adya,V., Sridhar, and D. Nath. "Flexible Global Software Development (GSD): Antecedents of Success in Requirements Analysis. "*Journal of Global Information Management* 17, no. 1, January 1, 2009: 1-31.

[21] Carmel, E. and Agarwal, R. "Tactical approaches for alleviating distance in global software development. "*IEEE Software,* vol. 18, no. 2, March 2001, pp. 22-29.

[22] Bird, Christian, et al. "Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista." *Communications of the ACM* 52, no. 8, August 2009, pp.85-93.

[23] Carmel, E., *Global Software Teams: Collaborating across Borders and Time Zones*. Prentice Hall, 1999.

[24] MacCormack, A, Verganti, R. and Iansiti, M. "Developing products on internet time: the anatomy of a flexible development process," *Management Science*, Jan 2001.

[25] Bordens, Kenneth, Abbott, S. and Bruce, B., *Research Designs and Methods: A process Approach.* Mayfield Publishing Company; Mountain View, CA, 1996, pp. 54-60.

[26] Aiken, L. and West, S., *Multiple Regression: Testing and interpreting interactions*, Newbury Park, CA: Sage, 1991.

[27] Cohen, J., Cohen, P., West, S. and Aiken, L., *Applied multiple regression/ correlation analysis for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum, 2003.

[28] Cohen, M. "The Scrum Process", http://www.mountaingoatsoftware.com/topics/scrum, 2005.

Note: This paper is abstracted from a working dissertation at The George Washington University, and is a revised paper published in the proceedings of the WASET 2010 International Conference on Computer, Electrical, and Systems Science, and Engineering, Cape Town, South Africa, January 29-31.

ACKNOWLEDGEMENTS

Authors would like to thank Dr. Tom Holzer, Scott Ambler, Walter Bodwell, Brian Drummond, and Brad Swanson for their support and constructive insights.

AUTHOR BIOGRAPHIES

Ronzelle L. Green is a doctoral student in the Engineering Management and Systems Engineering Department in the School of Engineering and Applied Science at the George Washington University, Washington, D.C. He has held management and engineering positions within large and small Agile development projects. He's also a member of IEEE.

Thomas Mazzuchi, D.Sc. received a B.A. (1978) in Mathematics from Gettysberg College, Gettysberg, PA, a M.S. (1979) and a D.Sc. (1982), both in Operations Research from the George Washington University, Washington DC. Currently, he is a Professor of Engineering Management and Systems Engineering in the School of Engineering and Applied Science at the George Washington University, Washington, D.C. He is also the Chair of the Department of Engineering Management and Systems Engineering at the George Washington University where he has served as the Chair of the Operations Research Department and as Interim Dean of the School of Engineering and Applied Science.

Shahram Sarkani, Ph.D., P.E., is a Professor of Engineering Management and Systems Engineering at The George Washington University. Professor Sarkani has engaged in engineering research, technology development, and engineering education since 1980. He is author of over 150 technical publications and presentations. He remains engaged with important ongoing research in the fields of engineering management, systems engineering, civil engineering, and logistics management. Since 1987, he has conducted sponsored research with such organiza-

tions as NASA, the National Institute of Standards and Technology, the National Science Foundation, the U.S. Agency for International Development (in association with Zagazig University, Egypt), the U.S. Department of Interior, the U.S. Department of Navy, the U.S. Department of Transportation, and Walcoff and Associates Inc.