Communication and Quality in Distributed Agile Development: An Empirical Case Study

R. Green, T. Mazzuchi, and S. Sarkani

***Abstract*—**Through inward perceptions, we intuitively expect distributed software development to increase the risks associated with achieving cost, schedule, and quality goals. To compound this problem, agile software development (ASD) insists one of the main ingredients of its success is cohesive communication attributed to collocation of the development team. The following study identified the degree of communication richness needed to achieve comparable software quality (reduce pre-release defects) between distributed and collocated teams. This paper explores the relevancy of communication richness in various development phases and its impact on quality. Through examination of a large distributed agile development project, this investigation seeks to understand the levels of communication required within each ASD phase to produce comparable quality results achieved by collocated teams. Obviously, a multitude of factors affects the outcome of software projects. However, within distributed agile software development teams, the mode of communication is one of the critical components required to achieve team cohesiveness and effectiveness. As such, this study constructs a distributed agile communication model (DAC-M) for potential application to similar distributed agile development efforts using the measurement of the suitable level of communication. The results of the study show that less rich communication methods, in the appropriate phase, might be satisfactory to achieve equivalent quality in distributed ASD efforts.

***Keywords*—**agile software development (ASD), distributed software teams, media richness theory, software development.

I. 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" [13]. 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 (quality). Performance (quality) is especially important. In general, performance (quality) is calculated using the following formula (Equation 1):

$$\frac{Errors}{KSLOC} = Performance / Quality \quad (1)$$

The ability of a development team to produce quality applications requires an understanding of the requirements. Known requirements produce a somewhat consistent and expected baseline cost. Historically, unpredictable and changing user requirements and lack of development cohesiveness yield higher costs and defects in software. On the other hand, the ultimate goal of agile software development (ASD) is reducing the cost of change (user requirements or other influences) that may engulf a project [7]. 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" [7]. Communication within agile development teams is critical to meeting cost, schedule, and quality goals. To meet these goals, agile development methods recommend collocation of the entire development team. The Agile Manifesto (the cornerstone 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" [19]. In addition, research has shown that having development teams work in the same physical environment improves communication and solidifies clarity. Unambiguous, succinct, and direct communication is important for an ASD team during all 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.

This investigation used historical development data from BMC Software's distributed agile development release of Performance Manager 2.3 [21]. The paper analyzed pre-release defect rates, communication mediums, and development phases within the project. The study empirically evaluated the hypothesis that successful distributed ASD teams that use less rich communications techniques (in the appropriate phase) will achieve comparable quality (defect rates) results for pre-release software than collocated agile software teams. This study helps shed light on the degree of communication needed to successfully meet performance

R. Green is a System Engineering doctoral student at The George Washington University, Washington, DC, 20052 USA (e-mail: ronzelle@gwu.edu). This paper is summarized from a dissertation in progress.

T. Mazzuchi is the Chair of the Department of Engineering Management and Systems Engineering at The George Washington University, Washington, DC, 20052 USA (e-mail: mazzu@gwu.edu).

S. Sarkani is a Professor of Engineering Management and Systems Engineering at The George Washington University, Washington, DC, 20052 USA (e-mail: sarkani@gwu.edu).

(quality) objectives by distributed agile development teams. Distributed software development consists of two or more teams working to develop software from different geographical locations [20]. These geographically dispersed teams face time zone and cultural differences that may include, but are not limited to, language, tradition, values, and norms of behavior [20].

II. Agile Software Development And Collocated 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 the 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, declared that the iterative approach "allows the user to instantly incorporate feedback into the process to improve functionality" [9]. A very integrated, collocated 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, collocated teams.

A popular form of ASD is Scrum. Scrum is a "management, enhancement, and maintenance methodology for an existing system or production prototype" [17]. 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 [17]. One of Scrum's guiding principles is to "keep everything visible" and engage everyone in identifying obstacles [16]. 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 [17].

In general, ASD initiates the idea of collocated development teams and iteration for combating changing and unpredictable requirements. Collocation implies close proximity, face-to-face communication, timely feedback, and informal social interaction [8]. The notion of proximity refers to "the physical distance between people…" [8]. Collocation is one of the key tenants of ASD. Collocation 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 [9]. This process of understanding requirements, developing software, and incorporating feedback occurs multiple times until an application meets users' requirements. Usually, during this process, 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 [9].

Unfortunately, ASD has its perceived shortfalls. Opponents of ASD state that it does not scale well in large projects or distributed environments. As Boehm [2] 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 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 originally 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" [11]. Bauer further explains a detailed, systematic process from requirements to delivery of software. Within the last two decades, the perceptions about 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 [1]. 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 requirements 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.

III. 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 systematically segregated, four

distinct phases of the agile development cycle:

A. Phase I: Planning, Architecture, High Level Design

Phase I centers around overall product planning, architecture, and high-level design. Phase I states all desired product requirements at some level. User stories capture these requirements.

B. Phase II: Analysis & Prioritization

Phase II focuses on the individual sprint backlog. The product owner and development team analyze the sprint backlog and prioritize desired features for the sprint/iteration. Within the development team, members collaboratively estimate the level of effort necessary to implement the desired features.

C. Phase III: Design & Code

Phase III is a continual evolution of design and coding for the development team. This phase also encompasses the daily Scrum sessions for the entire team.

D. Phase IV: Integration, Test, Documentation, & Release

Phase IV is a continuation of the Phase III design and code session. This portion of the cycle integrates testing. In addition, documentation and software release/deployment, sprint review, retrospective, and product demonstration occur in this phase. The expected end of this phase produces working software.

To successfully integrate and complete the entire process, the development team must uniquely approach each 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 stage is critical, and concatenated with the other phases encompasses the complete agile development lifecycle. As expected, each stage is linked and must be accomplished sequentially for effective results

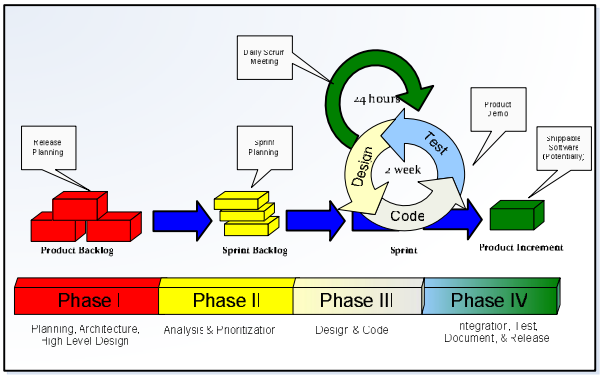


Fig. 1 Four Stages of Agile Development

IV. 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 medium [5]. MRT categorizes media in a hierarchy of established richness based on the "availability of instant feedback; the capacity of the medium to transmit multiple cues such as body language, voice tone, and inflection; the use of natural language; and the personal focus of the medium" [4]. Daft and Lengel deem communication rich if it can clear *ambiguous* and *uncertain* issues in a timely manner. In addition, their theory proposed various forms of communication media possessing different capacities for solving uncertainty and ambiguity [4]. MRT implies that "richer media are more effective for equivocal tasks, and leaner media are better for unequivocal tasks" [6]. MRT presumes that people are motivated to overcome equivocality and that various forms of communication media each have optimal uses [6]. Daft and Lengel defined communication ambiguity as the difference between the amount of communication needed to perform tasks and the amount of information possessed by the organization. Equivocality is the ambiguity in tasks caused by varying, perhaps conflicting interpretations of a situation by groups or individuals [4]. For effective communication to occur, the richness of the medium should match the level of message ambiguity [5]. In this context, MRT helps evaluate communication media choices. Because of the reduced contextual cues and less rapid feedback mechanisms, media other than face-to-face is considered less rich [5]. The theory suggests that tasks requiring a considerable amount of collaboration require richer media. Face-to-face communication is a richer media than any type of computer-mediated communication [22]. Face-to-face communication has major advantages over other forms of communication. As Vrasidas and McIsaac explain, "a major disadvantage of text-based CMC is the lack of visual and auditory cues" [15]. These visual and auditory cues can be translated in body language that may provide additional meaning [15]. The overall goal is to help reduce ambiguity of communication through the appropriate selection of communication media. This theory was analyzed within the context of agile system development within development teams.

V. Distributed ASD At BMC Software

With its headquarters in Austin, TX, BMC Software primary business focus area is system management. The software BMC creates monitors applications, networks, and infrastructure for data centers and other facilities. BMC's customer base includes many of the Fortune 1000 global companies. In 2004, BMC strategically decided to quickly enhance and consolidate their flagship products into Performance Release Manger in an attempt to increase future revenues and decrease time to market. BMC's first expected

delivery of the product was within one year of this decision. BMC decided to use ASD methods to meet its goals. Moreover, BMC was determined to use a non-traditional agile best practice to accomplish this feat. BMC used a large (92 people) distributed agile development in three countries, six locations, over eleven time zones:

TABLE 1
DISTRIBUTED LOCATION & TIME ZONES

11pm	1am	8am	11:30am
Silicon Valley, CA	Austin, TX Houston, TX	Tel Aviv, Israel Tel Haifa, Israel	Pune, India

BMC would face potential communication and quality challenges using a collaborative software development technique to rapidly enhance their product with a highly distributed development workforce. Historical data and best practices insist collocation of the development team produces concise communication, which results in higher productivity and quality.

VI. PROBLEM STATEMENT, RESEARCH QUESTIONS (RQ), AND HYPOTHESES

A. Problem Statement

ASD uses techniques of dividing projects into smaller manageable deliveries, while utilizing cross-functional teams that collaboratively plan, analyze, design, develop, test, and integrate [18]. Successful agile development teams rely on cohesive and interactive environments to produce quality software while absorbing continuously changing user requirements. Historically, to achieve this goal, agile teams were usually collocated. Nevertheless, as corporations strive to decrease time to market and lower development costs (utilize people resources throughout the world and continuous development efforts), agile has become a prime software development candidate for implementation globally. Organizations increasingly disperse due to cost-cutting measures, acquisition of global talent, and a general belief that this type of organizational structure may result in heightened productivity surpassing that of face-to-face teams [14]. Regardless, the key principles of ASD are consistent, synergic communication with users and within the development team. Precise communication is important during the four agile development phases. This project determined the level of communication necessary for agile distributed development teams to achieve similar quality achieved by a collocated agile development project. At the conclusion of the study, the author will recommend the level of communication necessary to mimic the success of collocated agile development teams in distributed environments. Once accomplished, projects may be able to reduce overall development costs, increase speed-to-market, and maintain quality while achieving successful deliveries.

B. Research Questions

Previously, IT professionals have strived to deliver cost efficient, relevant, and defect free software to their users. In most cases, these solutions are in direct response to user needs and requirements. Overall, many software projects do not succeed because "vague problem statements and imprecise scope definition lead to unstable user requirements that result in an unstable application development environment" [12]. Also, problems with quality arise. Along with user requirements, scope definition is a very important ingredient. If scope definition is uncertain, user requirements may also be unclear. Within this paradigm, it is difficult to achieve derived, stable user requirements. Unfortunately, adding to this complexity is communication effectiveness of the development team, technology maturity, and environmental factors that directly affect usability, functionality, and effectiveness of systems. This paper examines the feasibility of distributed ASD methods and its ability to produce quality software.

Inherently, 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 distributed to customers at an early stage" [10]. 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 goal of the research is to determine the optimal richness of communication needed within the development team to produce quality software. The paper will address the four Research Questions below:

1) Identify communication factors that affect the success or failure of ASD projects in distributed environments.
2) Within the agile development lifecycle, identify the level of communications necessary to enable successful agile projects in distributed agile development teams.
3) Determine the effect of distributed ASD on pre-release software defect rates.
4) Determine media instruments and communication frequency needed to improve defect rates and software quality in distributed agile projects.

C. Hypotheses

The current study will examine the following major, minor, and null hypotheses. It will use empirical data from BMC software to evaluate each of the hypotheses below. The goal of the investigation is to support the major and minor hypotheses. The study will attempt to reject the null hypothesis. The hypotheses are as follows:

Major Hypothesis: Successful distributed ASD teams that use less rich communications techniques (in the appropriate phase) will achieve quality (defect rates) results for pre-release software comparable to those of collocated agile software teams.

Minor Hypothesis: Distributed ASD teams that use richer communication techniques within Phase I & II will achieve quality (defect rates) results for pre-release software comparable to those of ASD teams that are collocated.

Null Hypothesis: Distributed ASD teams that use less rich communications techniques (in the appropriate phase) will not achieve quality (defect rates) results for pre-release software comparable to those of collocated agile software teams.

VII. Methodology And Data Collection

The research methodology used an empirical case study approach of BMC Software. The investigation consists of historical quantitative and qualitative data. The quantitative data leverages data and research received through Cutter Consortium and QSM Associates (Michael Mah) [21]. QSM develops and maintains a database with 7,500 completed projects. This database has productivity statistics and trends for cost, schedule, and quality for worldwide software projects. This data assists with identifying baseline defect rate for collocated agile projects, and vital statistical information for BMC's distributed agile project. The case study analyzed BMC's Performance Manager Release (PMR) 2.3. BMC developed PMR using Scrum methods across six distributed locations with time differences as much as 11 hours. The project yielded over 837,375 lines of new and modified code

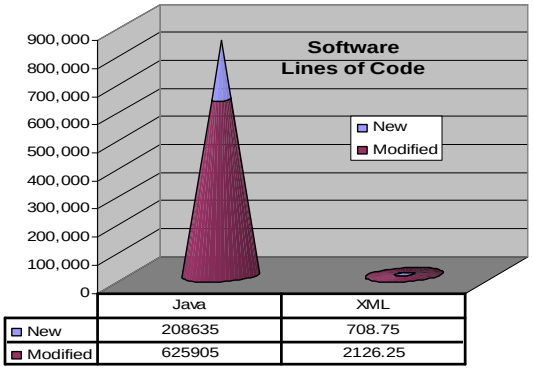


Fig. 2 New vs. Modified Software

[21]. The majority of the code was modified Java. A small portion of the code was XML.

Other historical data show the number of people on the team, time to complete the project, iteration lengths, number of iterations, distributed teams, and user stories (requirements) completed. See Table II below for vital statistics of BMC's Performance Manager Release 2.3 [21]:

Table II
BMC Performance Manager Release 2.3

Distributed Agile Teams	6
Persons on development team	92
Months to Complete	5.5 months
Number of Iterations	11
Iteration Length	2 weeks
User stories (requirements)	918
Communication Mediums	Face-to-Face (H), Video Teleconference (HM), Teleconference/Phone/Skype/Instant Messenger (M), Podcast/Recorded Webcast/Web-based Tracking Tool (ML), Email/Documentation/Wikis (L)

This project collected additional data through structured interviews with members of the project team and various consultants. The result of this investigation measured the levels of successful communication modes during the agile development process in a distributed environment. It is expected that this model could be mimicked in other distributed environments with similar factors. Industry averages for defects for agile projects in collocated environments are as follows:

Table III
Defect Rates & KSLOC

Projects	Code Size (KLOC Java & XML)	Pre-release Defects	Defect Rate (Defect/KLOC)
Performance Manager Rel 2.3	837	635	0.76
*Other Projects - QSM Agile Industry Average (Collocated)	**700**	**713**	**1.02**

*** Data from 2005 - 21 projects (average)**

Table III describes lines of code for Performance Manager Release 2.3, pre-release defects, and defect rates for final full integration and regression testing. In addition, the table shows the industry average lines of code, pre-release defects, and defect rates for collocated agile projects as of 2005 in QSM's database [21]. The defect rates account for the final integration and complete regression testing after software coding ended.

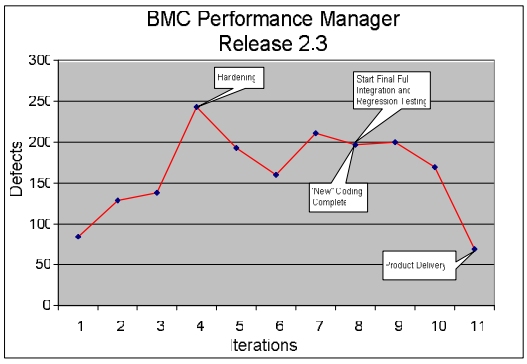


Fig. 3 Defect by Iterations

This study collected data by phone interview, documentation, and historical research. The interviewees consisted of consultants, the BMC Performance Manger 2.3 program manager, and the former Director of Engineering for BMC.

The information collected reflects the communication media used during each phase of the development process and resulting defects. BMC used a variety of communication methods. Each method proved to have pros and cons. A scale developed for the assortment of communication tools BMC used in the development of Performance Manager 2.3 reflects standard 1-5 categorization. The scale ranges from high (face-to-face: 5) rich communication to low (email: 1) less rich communication medium. This study applied this scale to evaluate the communication methods per iterations.

Face-to-Face (H), Video Teleconference (HM), Teleconference/Phone/Skype/Instant Messenger (M), Podcast/Recorded Webcast/Web-based Tracking Tool (ML), Email/Documentation/Wikis (L)

H(5), HM(4), M(3), ML(2), L(1)

	Phase I	Phase II	Phase III	Phase IV
Iteration 0	5	5	N/A	N/A
Iteration 1	5,2,1,1	5,3,3,2,2	5,5,3,3,3,3,1,1,1	5,5,3,3,3,2,1,1
Iteration 2	5,2,1,1	5,3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 3	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 4	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 5	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	5,3,3,3,2,1,1
Iteration 6	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 7	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 8	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 9	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 10	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Iteration 11	5,2,1,1	3,3,2,2	5,5,3,3,3,3,1,1,1	3,3,3,2,1,1
Average	2.86	2.85	2.78	2.30

Fig. 4 Communication Mediums by Phase

To explore the uses and effects of communication mediums within agile development teams distributed around the world, the case study uses quantitative and qualitative data collection and analysis techniques.

VIII. Conclusion

The conclusion reflects the significant role of communication mediums within the agile development process. One of the key tenants of agile is frequent and continuous communication between developers and users. Agile anticipates change, and developers must be poised to accept it and adeptly take action. Within the agile development process, it is of certainty that best practices in many successful distributed projects involve richer synchronous communication mediums. The research shows that this communication is particularly indispensable during the beginning of the development project (Figure 5). Despite the considerable power of today's asynchronous technologies for dispersed work, there are still powerful reasons for synchronous communication [3]. Regardless, as organizations desire to reduce development timelines, deliver their products to market faster, and leverage cheaper software development resources across the world, each organization must utilize a combination of communication mediums for successful agile implementations.

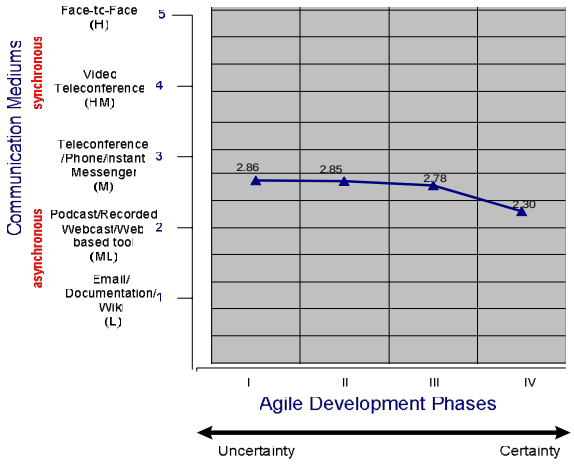


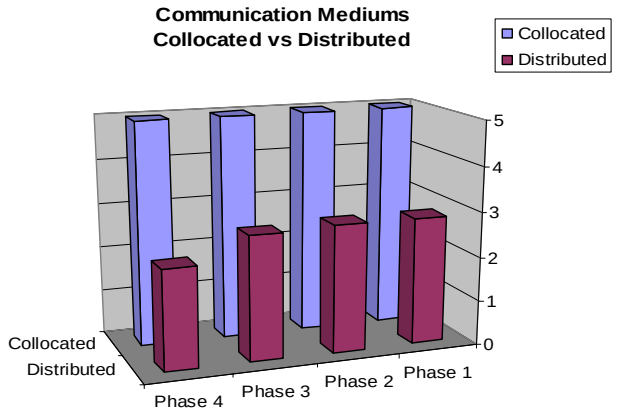
Fig. 5 Communication Method by Phase

1) *Phase I and II of a distributed agile software development project tend to require richer communication and less communication channels.* The data supports the notion that the initial stages of a distributed agile development project need richer communication. As expected, these phases are most turbulent initially, and lend themselves to increased uncertainty. Thus, development teams must understand emerging and changing requirements to produce software that meets user needs and quality goals. The figure above shows that ASD requires richer communication at the start of the project.
2) *The product demonstration at the end of each iteration is a key method to ensure communication within the development teams is synchronized*. As user requirements transform, distributed development teams do not always have the luxury of understanding the complexity of the new direction. In any event, the demonstration at the end of the iteration is a key communication instrument for the development teams to resynchronize their vision and direction. This product demonstration at the end of each sprint level sets the development teams in all locations.
3) *Focused face-to-face communication in the beginning of a distributed agile project may suffice for face-to-face collaboration throughout the iterations.* Even with a large team, BMC was able to effectively communicate within their development team to accomplish admirable results. In part, their "iteration 0" builds a vision, team cohesiveness, and foundational artifacts for their distributed teams. Of note, BMC had a large portion of their development team centrally located (design and code). Regardless, BMC realized the benefits of iteration "0" throughout the 11 iterations and teams.
4) *The use of other synchronous techniques may supplement continuous face-to-face and proximity shortfalls*. The observations of this study show that other less rich communication methods may substitute for face-to-face communication. As explained, visual cues in communication may not be necessary for a successful agile development project. In any case, the study shows that synchronous communication techniques are critical during high uncertainty periods. With the lack of abundant documentation, clear, concise communication is necessary.
5) *Quality requires a combination of richer and less rich communication mediums during the entire agile development process.* Within a distributed agile development project, face-to-face communication is costly and most likely not feasible. Thus, mechanisms must be in place to support communication requirements during the optimal periods to mimic quality results of collocated teams. Apply a blend of synchronous and asynchronous methods for use throughout the development. Given the correct mixture, distributed agile development teams may be as effective as collocated agile teams.

BMC's overall approach yields a potential model for agile

professionals to apply within distributed development processes that may produce similar software quality results with projects of comparable parameters.

By isolating communication mediums, the study shows



Method	Phase 1	Phase 2	Phase 3	Phase 4	Defects	SLOC
Collocated	5	5	5	5	713	700,000
Distributed	2.86	2.85	2.78	2.30	635	837,375

Fig. 6 Collocated vs Distributed

distributed agile development projects with similar constraints may produce equivalent quality results in terms of pre-release defects.

The author believes one can measure the proper amount of communication contact needed between distributed agile development teams within phases to construct a reasonable initial model. The study shows a pattern between communication methods (richness), and the degree of software quality measured in pre-release defects. Confidently, the author believes other factors do play a role in successful distributed ASD projects. Moreover, many agile professionals demand the prerequisite of collocation for successful agile projects. Nevertheless, BMC Software's method proves that distributed ASD can be effective if the proper communication methods and tools are in place for the development teams. These tools can be tempered according to necessity clear and concise communication within the development project. ASD is no longer a niche development process, but a bona fide method to produce quality software. The author believes the bounds of proximity no longer restrict ASD. Agile can be truly successful in a distributed environment if credence is given to the communication needs of the development team during the optimal phase of the software lifecycle.

Acknowledgment

R. Green thanks Michael Mah, Mike Lunt, and Walter Bodwell for their support in gathering data and insights for this paper.

References

[1] R. 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.

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

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

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

[5] R. L. Daft, R. H. Lengel, and L. K. Trevino, "Message equivocality, media selection, and manager performance: Implications for information systems," *MIS Quarterly*, vol. 11, no. 3, pp. 355-366, Sept. 1987.

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

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

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

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

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

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

[12] R. Ram, "Is your IT project OA?," *InformationWeek*, pp. 162, November 29, 1999.

[13] The Standish Group International, Inc. (2009). "Extreme chaos." [Online]. Available: www.vertexlogic.com/processOnline/processData/documents/pdf/extreme_chaos.pdf (accessed August 13, 2009).

[14] S. Townsend, "Over the waterfall," *ITNow*, vol. 49, no. 1, pp. 9, 2007.

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

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

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

[18] R. Holler (2006). "Mobile application development: a natural fit with agile methodologies." [Online]. Available: www.versionone.com/pdf/MobileDevelopment.pdf. (accessed June 2009).

[19] "Manifesto for agile software development." (2001) [Online]. Available: www.agilemanifesto.org

[20] E. Carmel, *Global Software Teams: Collaboration Across Borders and Time Zones*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[21] M. Mah, *How Agile Projects Measure Up, and What This Means to You*. Arlington, MA: Cutter Consortium, 2008.

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