Modified Agile Practices for Outsourced Software Projects.

CONFERENCE PAPER · JANUARY 2006

Source: DBLP

3 AUTHORS, INCLUDING:

Dinesh Batra
Florida International University
32 PUBLICATIONS **619** CITATIONS
SEE PROFILE

Thant Syn
Texas A&M International University
11 PUBLICATIONS **19** CITATIONS
SEE PROFILE

Modified Agile Practices for Outsourced Software Projects

Dinesh Batra
Florida International University
Dinesh.Batra@fiu.edu

Thant Sin
Florida International University
Thant.Sin@fiu.edu

Sheng-Ying Tseng
Florida International University
Sheng_Ying.Tseng@fiu.edu

ABSTRACT

In recent years, agile practices have become popular in the software development industry. Meanwhile, distributed development is becoming an alternative way to develop new systems. The most common mode of distributed development uses outsourcing primarily to reduce costs and ensure timely completion of projects. Agile methods are now being recommended for distributed development. However, the current agile practices break down when subjected to the realities of the outsourced development marked by geographical, language, temporal, social, and cultural barriers. The typical outsourced software project is also larger than the typical agile project. Thus, agile practices need to be modified so as to find application in today's software development environment, which deems outsourcing as an important component. The paper comes up with a collaboration structure that can incorporate modified agile practices for outsourced projects.

Keywords (Required)

Agile practices, outsourcing, software development, geographical barrier, temporal barrier, cultural barrier.

INTRODUCTION

The frustration with the bureaucracy of the disciplined approach has led to the proposal for agile development (Boehm and Turner, 2003). The new approach is defined by its Agile Manifesto (http://agilemanifesto.org/) that values individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Larman, 2004). Examples of agile development include methodologies such as XP (Beck, 2000; Auer and Miller, 2002), Adaptive Software Development (Highsmith, 2000), Crystal (Cockburn, 2005), and Scrum (Schwaber and Beedle, 2002).

Agile development is not just a new approach for software development, but it also indicates a significant cultural shift from the discipline and planning based approaches that focus on process improvement (Crosby, 1979; Deming, 1986; Juran, 1988). The approaches are best exemplified by the Capability Maturity Model (CMM) and Capability Maturity Model Integrated (CMMI), which gained popularity in the 1990s (Ahern, Clouse, and Turner, 2004). Recent work by Curtis, Hefley, and Miller (2002) addresses the critical people issues in organizations.

Agile development does not focus on process improvement; instead it focuses on customer satisfaction and employee empowerment. This is evident from reading the values and principles of the agile manifesto, which include fairly extreme positions on maintaining customer satisfaction such as "welcome changing requirements, even late in development" and "the best architectures, requirements, and designs emerge from self-organizing teams" (http://agilemanifesto.org/). The agile principles (http://agilemanifesto.org/principles.html) may appear somewhat maverick at first sight, but are direct, clear, and generally quite appealing.

A recent study by Holmstrom et al (2006) indicates that agile practices can reduce temporal, geographical, and socio-cultural distance in distributed development projects. They studied agile development between teams located in the US and Ireland, and although they report a positive experience overall, several problems were ascribed to the three distances. For instance, they report the following quote by a Hewlett Packard manager: "Language…it's a really, really difficult problem". This

problem was observed between two English-speaking teams (one in the US and the other in Ireland) belonging to the same company!

Although there are other reported successes of distributed agile (e.g., Kircher et al, 2001), it seems that the projects reported are small, the team members despite the distance are somewhat familiar with each other, and the participants are of high caliber. However, the world is not full of experts or needs only small projects. Fowler (2004) discusses the successful use of agile practices in a scenario that involves a US based office and an India based office. He states that their Bangalore, India based company hires only 1 in 200 developers who apply for a position. However, one cannot generalize the success of an approach that excludes 199 out of 200 developers in a city known for its skilled developers.

In this paper, we attempt to extend distributed agile development from extremely specialized and limited projects to mainstream projects. This implies that we need to drop the strict requirements of expert personnel belonging to the same company and the small size of projects, but still address the geographical, temporal, and cultural distances. A mainstream distributed project in the near future will likely have a significant outsourcing component. To accommodate these constraints, agile practices need to be modified.

The next section sets the context of the paper by discussing key issues of outsourced projects. This is followed by an analysis first by the interaction of agile values and outsourced development, and then by the interaction of agile principles and outsourced development. This evaluation enables a discussion of a collaboration structure to facilitate modified agile practices for outsourced projects. Finally, we discuss how the modified practices address the problems encountered in distributed agile development.

KEY ISSUES OF OUTSOURCED PROJECTS

Although outsourcing is not equivalent to offshoring a software development project, the availability of skilled labor and economic advantages of considerably lower wages have made the two equivalents for all practical purposes. Outsourcing has a bright future (Brown and Wilson, 2005), and several studies (e.g., http://www.acm.org/globalizationreport/) have shown that it improves productivity and does not lead to loss of jobs. The primary motive of offshoring is to keep costs down (Coar, 2003), but there are several other noteworthy benefits such as the ability to undertake larger projects and expand markets (Sahay, 2003). Software development wages in the US, Canada, and European countries are considerably higher than most Asian countries such as India and China, and hence these countries are likely to be involved as outsourcing clients and vendors, respectively. However, some European countries (e.g., Ireland) are significant outsourcing vendors because of the quality of their developer pool.

When a software development project is outsourced, generally at least three parties are involved (Figure 1): the customer, the outsourcing client, and the outsourcing vendor. Let us assume that a customer in the US needs an information system. A software division in the same company or another software development company in the US may be asked to develop the system. This unit may outsource a portion of the project to a vendor in India. The software development unit, thus, become an outsourcing client.

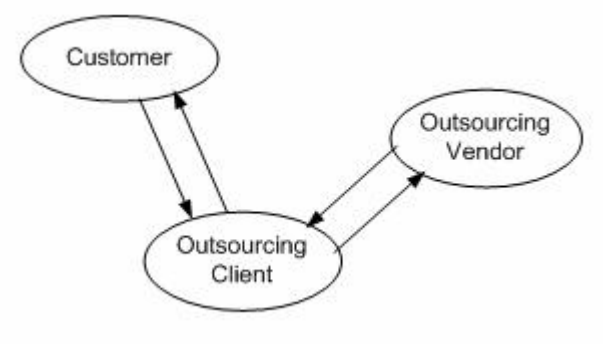

Figure 1. Outsourcing Parties

In applying agile practices to outsource projects, we need to consider several issues. The first issue is that of the size of the project. Other issues include the cultural distance, the temporal distance, and the geographical distance.

Size of the project: Boehm and Turner (2003) observe that agile development is suitable only for small teams. Beck (2000) states: "Size clearly matters. You probably couldn't run an XP project with a hundred programmers. Not fifty. Nor twenty, probably. Ten is definitely doable." It is evident that agile development is meant for small projects only. It is very plausible that the principle of self-organizing teams is successful in agile development – you are not organizing a hundred people team. Members of the teams are expected to be experts in design and programming, and have teamwork skills. The customer is available on a daily basis to guide development. Thus, it is not surprising that agile development leads to successful projects. After all, experts in design, technology, and teamwork can deliver a small project successfully even in the face of dynamic requirements.

Business needs dictate projects of all sizes, not just small projects. In fact, there is practically no need to outsource small projects since the overhead cost will likely exceed the savings from the labor cost. Since a larger project can be broken down into smaller pieces (Sahay, 2003; Nickerson and Zenger, 2004), we posit that each piece can be completed using agile development.

Cultural distance: Agile development denotes a significant shift in culture from discipline-based development. In agile development, there are no prescribed processes or tools; instead, the values and principles govern software development. In a distributed environment, the cultural values need to be shared by the outsourcing company, say based in the US, with the outsourced company, say based in India. In general, the US based team will easily adopt the agile culture because the US work culture in general is less bureaucratic and more flexible. The Indian team may have more difficulty in adopting to the agile values because most companies are hierarchical and structured, and the people value rank and class. In fact, adoption rates for People CMM, a discipline-based approach, appear to be highest in India (Curtis, Hefley, and Miller, 2002).

What is culture and why is it so important? Culture is the deposit of knowledge, experience, beliefs, values, attitudes, meanings, hierarchies, religion, notions of time, roles, spatial relations, concepts of the universe, and material objects and possessions acquired by a group of people in the course of generations through individual and group striving (Samovar and Porter, 1995). It is a pattern of shared basic assumptions that the group (social units of all sizes) learned as it solved its problems of external adaptation and internal integration, that has worked well enough to be considered valid and, therefore, to be taught to new members as the correct way to perceive, think, and feel in relation to those problems (Schein, 1992).

Work that takes place over long distances means that communication will often involve different cultures (Olson & Olson, 2003). Participants may be surprised by such interactions because they have not considered various cultural differences and how they impact the daily work of long-distance teams. These differences matter. When Chrysler and Daimler Benz merged in 1999, it became an exercise in reconciling American innovation and informality with German structure and bureaucratic form, which resulted in confusion, and organizational inconsistencies with little synergy (Keyton, 2005).

Temporal distance: In sharp contrast to the popular image of software developers as relatively introverted and isolated, they, in fact, spend a large proportion of their time communicating (Herbsleb & Moitra, 2003). The temporal distance (Boehm, 2002) is likely to mitigate communication, which is a critical ingredient of agile practice. When the time is 9 am in New York, it is around 7:30 pm in New Delhi!

Geographical distance: This factor can severely affect communication. Surely, the widespread adoption of email and other communication mechanisms has enhanced virtual communication. However, these mechanisms still largely facilitate business communication, not organizational communication. Business communication includes formalizes and planned messages, codified in letters, memos, reports, websites, and advertising campaigns. Organizational communication includes business communication as well as informal and day-to-day interactions among organizational members (Keyton, 2005). Information conversations include personal stories, gossip, rumors, and socializing that also reveal important cultural information (Ibarra and Andrews, 1993), especially when informal or causal talk is intertwined with task talk, such as when casual conversation begins or ends professional meetings (Boden, 1994). Organizational communication is a complex and continuous process through which organizational members create, maintain, and change the organization (Keyton, 2005). Agile practices, as currently prescribed, rely on organizational, not business communication. Geographical distance, thus, impacts agile practices.

In the next section, we evaluate the interactions between agile practices and outsourced development. This is followed by an evaluation of the interaction between agile principles and outsourced development.

THE INTERACTION OF AGILE VALUES AND OUTSOURCED DEVELOPMENT

The agile manifesto lists four fundamental values that should govern software development. The advantages and the problems encountered in subscribing to a value in an outsourcing environment are listed in Table 1.

Individuals and interactions over processes and tools: In co-located settings, agile practices improve organizational communication by facilitating informal interactions. Distributed agile development can rely mainly on business communication, but not on organizational communication. In distributed agile development, the informal communication is drastically broken down. One wonders if the face to face communication can be substituted by virtual communication and still produce the same team feeling. For example, the assumption of tacit knowledge is challenged, and the pair programming practice is likely to be drastically affected.

Working software over comprehensive documentation: The basic premise of outsourcing is that a project can be done faster and cheaper. Thus, the goal of working software is likely to be achieved faster. However, documentation is likely to be important in an outsourced project. If the project is actually a piece of a bigger development task, documentation is necessary so that the pieces can be integrated. As the project becomes bigger, one cannot rely on tacit knowledge alone. Since temporal factors may reduce communication, documentation will be required to enhance understanding and prevent ambiguities.

Agile Manifesto Values	**Advantages in an Outsourcing Environment**	**Problems in an Outsourcing Environment**
Individuals and interactions over processes and tools.	May improve organizational communication	Temporal distance may impede communication Both outsourcing and outsourced parties are likely to employ business (formal) instead of organizational (formal and informal) communication Likely to conflict with the cultural values of the vendor party
Working software over comprehensive documentation.	Software can be developed quicker	Documentation is essential for avoiding ambiguities
Customer collaboration over contract negotiation.		Contract between client and vendor parties may not easily be renegotiated
Responding to change over following a plan.	More resources available to handle changes	Vendor may not welcome changes

Table 1. Support for Current Agile Values in Outsourcing Environment

Customer collaboration over contract negotiation: The client and vendor are likely to be involved in a contract with service level agreements. Although some companies have opened offices at offshore sites and can avoid a contract, the remote may not be large enough to support projects on a routine basis. Customer collaboration will be affected by client-vendor contract, which cannot be renegotiated on an ad-hoc basis. This value, therefore, is likely to be mitigated.

Responding to change over following a plan: The customer does not directly work with the vendor, so there is less motivation on vendor's part to respond to changes proposed by the customer and transmitted through the outsourcing client. Vendor's response to proposed changes will depend on various factors such as the relationship with the outsourcing client, to what extent is the agile culture prevalent, what are the costs involved in responding to changes and how are these recovered, and how much future business is expected.

THE INTERACTION OF AGILE PRINCIPLES AND OUTSOURCED DEVELOPMENT

The agile manifesto lists twelve principles that should govern software development. The advantages and the problems encountered in subscribing to a value in an outsourcing environment are listed in Table 2.

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software: Early delivery is certainly possible because outsourcing can provide the extra labor force. However, continuous delivery may not be possible because of complications in writing a contract between the client and the vendor.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage: Since outsourcing creates a larger labor pool, changing requirements can be addressed. However, the vendor

may not like the fixed price contract as is often the case with agile development, and is likely to protest "welcoming changing requirements late in development".

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale: Since this paper does not regard small projects typically associated with agile development as cost effective for distributed development, one may wonder about the usefulness of delivering software in two week intervals. Even if the developer team is handling a piece of the overall project, two week delivery packets are unlikely to integrate with other developing pieces. A somewhat longer time frame is certainly feasible.

Agile Manifesto Principle	**Advantages in an Outsourcing Environment**	**Problems in an Outsourcing Environment**
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Early delivery is feasible	Continuous delivery may be difficult
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	More resources available	Outsourced party may not accept late changes
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	More resources available	A couple of weeks may be more difficult; a somewhat longer frame is certainly possible
Business people and developers must work together daily throughout the project.		Temporal and geographic distances may make it more difficult to work with the vendor
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.		Difficult to control motivation at the vendor's site Outsourcing client and vendor may not have the trust
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.		Practically impossible between the outsourcing client and vendor
Working software is the primary measure of progress.	Very feasible	
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Feasible	Fatigue because of temporal distance is inevitable if practices like pair programming are instituted
Continuous attention to technical excellence and good design enhances agility.		Technical excellence is limited by the labor pool
Simplicity – the art of maximizing the amount of work not done – is essential.	Very feasible	
The best architectures, requirements, and designs emerge from self-organizing teams.		Culture disparities between the outsourcing client and vendor may prevent self-organization
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.		There is not adequate motivation for the outsourced team to adjust behavior

Table 2. Evaluation of Current Agile Principles in Outsourcing Environment

Business people and developers must work together daily throughout the project. A daily collaborative effort between the outsourcing client and vendor is difficult if the temporal and geographic distances are large. For example, pair programming may be impractical between teams in the US and in India, somewhat practical between the US and Ireland or between the UK and India, but quite feasible between the UK and Ireland or the UK and Russia.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. This kind of a culture is likely to ensue in a small co-located team. It is difficult to expect that a remote site in a different country will imbibe the same cultural values. Although selective hiring can lead to success in stray cases, the magnitude of change is unlikely to happen at grass roots level. It also requires a high level of trust, which takes a reasonable period of time.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. This mode of communication between the outsourcing client and vendor is ruled out. Even virtual communication that mimics face to face working may be difficult because of cultural, technical, and temporal constraints. For example, a person in India may slightly shake his head when implying agreement; the person at the US end may take it as denial.

Working software is the primary measure of progress. This principle is very feasible, and the vendor party should not have a problem accepting it.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. Outsourcing spreads the work and makes the overall development objective quite feasible. However, temporal distance can lead to fatigue if techniques like pair programming are taken too seriously. The touted 40-hour work week can be ruled out, and some degree of fatigue and burnout is inevitable.

Continuous attention to technical excellence and good design enhances agility. Technical excellence is limited by the quality of personnel available in the market. If you need 5 excellent designers, you may be able to hire them. If you need 50 excellent designers, however, you may have to improvise the meaning of the term excellent. As Boehm and Turner (2003) point out, approximately 49.99% of designers are below average.

Simplicity – the art of maximizing the amount of work not done – is essential. This is an excellent principle in a dynamic world regardless of the development approach. It underscores the YAGNI (you aren't going to need it) principle.

The best architectures, requirements, and designs emerge from self-organizing teams. The ability of a team to dynamically and optimally structure and restructure itself based on the requirements is a complex skill and needs experience, trust, and a conducive culture. Organizations in countries like India and China favor a hierarchic culture. Thus, culture disparities between the outsourcing client and vendor may prevent self-organizations.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. This is related to the previous point. The ability to tune a structure or behavior is an advanced skill and needs adequate motivation on the vendor's part. The ability to dynamically tune team behavior requires a culture that welcomes change. When the culture and the skill set are missing, the team may actually become less effective due to haphazard adjustment.

MODIFIED AGILE PRACTICES FOR OUTSOURCED PROJECTS

In coming up with modified agile practices for outsourced projects, the underlying guideline is that the teams at the outsourcing client and the outsourcing vendor need not be configured the same way or perform the same functions. Currently, the literature suggests a configuration shown in Figure 2. Although working software is the measure of progress in agile development, the coding aspect need not be done at the client end. Software is more than just code and the customer cares about functionality, not about some mundane lines of code. We recommend a structure (see Figure 3) that attempts to minimize the problems associated with distributed agile while maintaining the essence of the approach.

The outsourcing client needs to stay reasonably close to the customer. Hence the client should outsource the activity farthest from the customer. This activity, indeed, is coding. Note that coding, not software per se, is outsourced. The outsourcing vendor should be primarily responsible for coding. The vendor's team should essentially be a programming team under a group leader. The idea is that communicating with a team that is culturally, temporally, and geographically distant is likely to create rather than solve problems since tacit knowledge is practically impossible to achieve in such a configuration. However, one can train a select number of personnel in agile practices. For example, the programming group leader needs to undergo training in agility principles and to convey it to the programming team. Testing and integration can happen at the client end on an ongoing basis.

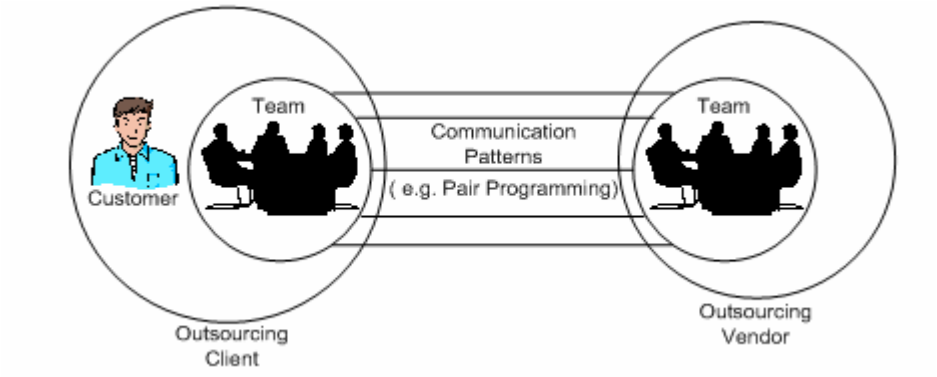

Figure 2. Current Agile Structure

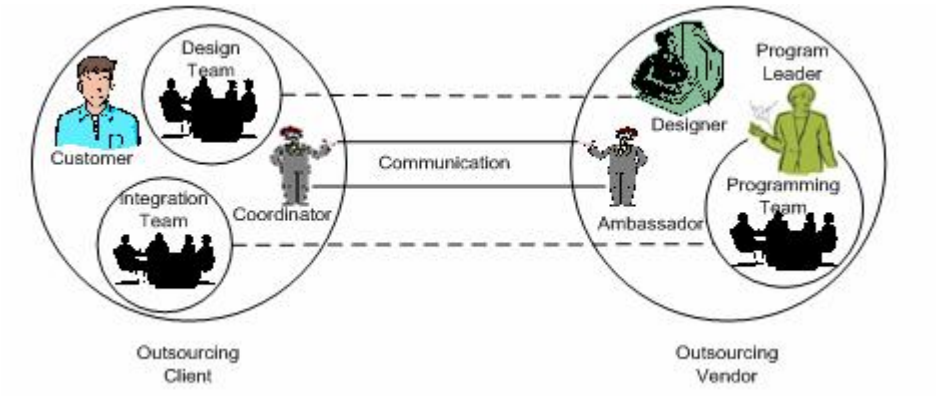

Figure 3. Modified Agile Structure

The outsourcing vendor ostensibly is there to make a profit. The espoused notion of a fixed price contract should not apply to the outsourcing vendor especially at the beginning of an outsourcing relationship. If the outsourcing client needs to keep the customer happy, then responding to changes can become frustrating for the vendor under a fixed price contract. Thus, the contract pricing is likely to be based on actual hours worked. This raises the issue of control.

To ensure control, the outsourcing client needs to send an ambassador to the vendor's site. The ambassador is also the main communication link and the mediator between the two parties. Typically, the ambassador should be someone who spent a reasonable period of her life in the vendor country. Say, the vendor is based in Hyderabad, India, a popular outsourcing region. Since a significant number of developers of Indian origin and currently working in the US have studied in Hyderabad or nearby areas, it is not difficult to find a person who can serve as an ambassador and "cultural mediator" given that many such persons do not mind working in India on the US kind of salaries.

The vendor team will have other roles, too. A useful role is that of the designer, who can help the ambassador interpret design specifications to be coded. The ambassador, the designer, and the programming group leader need to work closely so that software can be developed quickly and meet customer requirements. Although either one of the three can communicate with the client, it is evident that the ambassador will be the focal point of communication.

To prevent bewildering the ambassador, there needs to be a contact point on the outsourcing client side. This role is the team coordinator, who is the bridge between the customer and the outsourcing vendor. The client side needs to have two important groups. The agile manifesto lays stress on design, and so there needs to be a design group. It is assumed that the same group does analysis, too. However, the analysis and design here is continuous and is based on iterative design as well as responding based on feedback from the customer as well as from the emerging product. The agile manifesto also lays stress on working software, so there is the need for an integration group. This group, consisting mainly of programmers, continuously tests and integrates software so that the programming group at the vendor's end gets daily feedback. At the client end, however, the structure is fluid and self-organizing. As long as there are roles to fulfill the essential agile principles, the structure can adapt to the requirements.

ADDRESSING CULTURAL, TEMPORAL, AND GEOGRAPHICAL DISTANCES

The proposed modifications to the agile practices for outsourced projects significantly eliminate the problems caused by distances. Consider the temporal problem, for instance. It is practically impossible to work out a solution to the temporal distance if the client and vendor teams need to work in concurrent manner. In such a scenario, it is evident that the project will proceed during the day time at the client's convenience, which implies that it will be a tedious night shift for the vendor. This will eventually lead to burnout and turnover at the vendor end. Further, the practice would migrate from a partnership to a dehumanizing superior-subordinate relationship. The solution proposed considers a partnership relationship and facilitates daily communication. The client team can evaluate, test, and integrate programs written by the vendor team, and continue the design work on a daily basis. The cultural and geographical problems are solved through an ambassador, who is an interface between the two cultures.

CONCLUSION

The agile concept and values are very relevant in today's dynamic environment. However, agile development has shown success only in small projects. To scale the development, a large project can be divided into smaller parts with clearly defined interfaces. A project or a piece of the project can be outsourced for controlling cost in today's competitive global environment. However, outsourcing in the software industry generally tantamounts to offshoring from Western to Asian markets. Outsourcing brings its baggage of cultural, temporal, and geographic distances. As the analysis in this paper indicates, these problems mitigate the advantages of agile practices.

The modified agile practices address these problems by giving up on some practices while maintaining the essence of the agile values and principles, and providing a new collaboration structure. The new structure is not bureaucratic, but can be adjusted based on the needs and trust levels of the organizations. Eventually, the collaborations can lead to permeable, trusting organizations (Ashkenas et al, 2002). This is intrinsically the goal of agile manifesto.

REFERENCES

1. Ahern, D. M., Clouse, A., & Turner, R. (2003). Cmmi distilled: A practical introduction to integrated process improvement. Boston: Addison-Wesley.
2. Ashkenas, R. N. (2002). The boundaryless organization: Breaking the chains of organizational structure (2nd ed.). San Francisco, CA: Jossey-Bass.
3. Auer, K., & Miller, R. (2002). Extreme programming applied: Playing to win. Boston: Addison-Wesley.
4. Beck, K. (2000). Extreme programming explained: Embrace change. Reading, MA: Addison-Wesley.
5. Boden, D. (1994). The business of talk: Organizations in action. London; Cambridge, Mass.: Polity Press.
6. Boehm, B. (2002). Get ready for agile methods, with care. IEEE Computer, 35(1), 64-69.
7. Boehm, B. W., & Turner, R. (2003). Balancing agility and discipline: A guide for the perplexed. Boston: Addison-Wesley.
8. Brown, D., & Wilson, S. (2005). The black book of outsourcing: How to manage the changes, challenges, and opportunities. Hoboken, N,J.: John Wiley & Sons, Inc.
9. Coar, K. (2003). The sun never sits on distributed development. ACM Queue, 1(9), 32-39.
10. Cockburn, A. (2005). Crystal clear: A human-powered methodology for small teams. Boston: Addison-Wesley.
11. Crosby, P. B. (1979). Quality is free: The art of making quality certain. New York: McGraw-Hill.
12. Curtis, B., Hefley, W. E., & Miller, S. A. (2002). The people capability maturity model: Guidelines for improving the workforce. Boston: Addison-Wesley.
13. Deming, W. E. (1986). Out of the crisis. Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering Study.
14. Fowler, M. (2004, April 2004). Using an agile software process with offshore development. from http://martinfowler.com/articles/agileOffshore.html
15. Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering, 29(6), 481-494.
16. Highsmith, J. A. (2000). Adaptive software development: A collaborative approach to managing complex systems. New York: Dorset House Pub.

17. Holmstrom, H., Fitzgerald, B., Ågerfalk, P. J., & Conchúir, E. O. (2006). Agile practices reduce distance in global software development. Information Systems Management, Forthcoming.
18. Ibarra, H., & Andrews, S. B. (1993). Power, social influence, and sense making: Effects of network centrality and proximity on employee perceptions. Administrative Science Quarterly, 38(2), 277-303.
19. Juran, J. M. (1988). Juran on planning for quality. New York, London: Free Press; Collier Macmillan.
20. Keyton, J. (2005). Communication & organizational culture: A key to understanding work experiences. Thousand Oaks, Calif.: Sage Publications.
21. Kircher, M., Jain, P., Corsaro, A., & Levine, D. (2001). Distributed extreme programming. Paper presented at the XP2001 Conference, Sardinia, Italy.
22. Larman, C. (2004). Agile and iterative development: A manager's guide. Boston: Addison-Wesley.
23. Nickerson, J. A., & Zenger, T. R. (2004). A knowledge-based theory of the firm-the problem-solving perspective. Organization Science, 15(6), 617-632.
24. Olson, J. S., & Olson, G. M. (2003). Culture surprises in remote software development teams. ACM Queue, 1(9), 52-59.
25. Sahay, S. (2003). Global software alliances: The challenge of standardization. Scandinavian Journal of Information Systems, 15, 3-21.
26. Samovar, L. A., & Porter, R. E. (1995). Communication between cultures (2nd ed.). Belmont, Calif.: Wadsworth.
27. Schein, E. H. (1992). Organizational culture and leadership (2nd ed.). San Francisco: Jossey-Bass.
28. Schwaber, K., & Beedle, M. (2002). Agile software development with scrum. Upper Saddle River, NJ: Prentice Hall.