By BALASUBRAMANIAM RAMESH, LAN CAO, KANNAN MOHAN, *and* PENG XU

CAN DISTRIBUTED SOFTWARE DEVELOPMENT BE AGILE?

Three organizations studied here suggest the answer is "yes," when the unique characteristics of both environments are successfully blended.

Global sourcing and distributed software development have become a common business reality. Moreover, the current dynamic business environment requires organizations to develop and evolve software systems at Internet speed. As a consequence of these major trends, software development organizations have been striving to blend agile software development methods like Extreme Programming and distributed development to reap the benefits of both. However, agile and distributed development approaches differ significantly in their key tenets. For example, while agile methods mainly rely on informal processes to facilitate coordination, distributed software development typically relies on formal mechanisms.

Can such differences be reconciled to support agile distributed software development? What are the challenges that arise from blending agility with distributed development? How can these challenges be addressed? We investigated these questions by studying three organizations that have adapted their practices to support such development.

The table here summarizes the challenges to distributed development and identifies how adopting an agile approach creates a new set of challenges to agile distributed development.

Challenges in Distributed Development [2,3,5]		Characteristics of Agile Development [1,4]	New Challenges in Agile Distributed Development
Communication challenges	• Difficult to initiate communication • Misunderstanding/miscommunication • Dramatically decreased frequency of communication • Increased communication cost—time, money, and staff • Time difference	• Lack of formal communication • Increased demand for informal communication	*Communication need vs. communication impedance*
Lack of control	• Difficult to control process and quality across distributed teams	• Lightweight process • Ongoing negotiation • Reliance on skilled people	*Fixed vs. evolving quality requirements* *People vs. process oriented control*
Lack of trust	• Lack of trust between distributed team members • Lack of team morale	• Cohesive team • Trust built progressively • Short commitment	*Formal vs. informal agreement* *Lack of team cohesion*

Impact of agility on challenges to distributed development.

Challenges in Agile Distributed Development

Communication need vs. communication impedance. Distributed software development relies on formal mechanisms such as detailed architectural design and plans to address impediments to team communication that result from geographical separation. As agile development relies more on informal interactions than explicit documentation, it poses a real challenge in a distributed environment. How can we achieve a balance in formality of communication in agile distributed environments?

Fixed vs. evolving quality requirements. Due to the limited ability to control activities of remotely located teams, distributed development often relies on fixed, upfront commitments on quality requirements. In contrast, agile development relies on ongoing negotiations between the developer and the customer for determining the acceptable levels of quality at various stages of development. How can we achieve a balance between the fixed and evolving quality requirements?

People- vs. process-oriented control. In distributed development environments, control is often achieved by establishing formal processes. Agile environments, on the other hand, are more people-oriented and control is established through informal processes. What would be the appropriate balance between people- and process-oriented control in agile distributed development?

Formal vs. informal agreement. Contracts in agile environments are loosely and informally defined. In contrast, distributed development relies on explicit targets, milestones, and detailed specification of requirements. What is the appropriate level of formality in developing contractual agreements in agile distributed development?

Lack of team cohesion. In distributed development, participants at different sites are less likely to perceive themselves as part of the same team when compared to co-located participants. Lack of cohesiveness and shared view of goals are problems in such an environment. These problems are even more pronounced in agile development, which emphasizes constant cooperation on all aspects of the project. How can team cohesion be improved given the constraints of a distributed environment?

Project teams involved in agile distributed development must adopt practices that address these questions. We present some successful practices observed in three organizations we refer to here as Telco, Manco, and Consult (see the sidebar for details on how the multi-site case study was conducted). These practices leverage the unique characteristics of agile and distributed development to form a successful blend.

The Balancing Act: Practices to Achieve Distribution and Agility

The practices that may be characterized as agile but disciplined have evolved in the three organizations after repeated experimentation. These practices are in the spirit of "lightweight" methods, but have been adapted to meet the competing demands of distributed development as well. We have classified these practices into five groups.

1. Continuously adjust the process. Instead of strictly following the agile development practices as commonly defined, the companies continuously tweak them to fit the evolving needs of their projects.

Planning iterations to finalize requirements and develop designs. Skeptical of agile development that does not include adequate upfront design, both Manco and Consult devoted the first two or three iterations of a project to finalize critical requirements and develop a high-level architecture. Con-

sult used a SCRUM-like process during two upfront design cycles for this purpose.

Documenting requirements at different levels of formality. Telco and Manco altered their requirements analysis process significantly for agile distributed development. Short use cases and user stories were employed instead of detailed use case specifications. Also, product managers and customer representatives were encouraged to clarify requirements by sketching out acceptance tests. At Telco, the offshore team felt that such minimally documented requirements were more helpful than just informal communication.

2. Facilitate knowledge sharing. Knowledge sharing between and within teams was supported to enhance developers' shared understanding of the application and business domains. Processes and tools were developed to minimize the overhead to participants in knowledge-sharing activities.

Maintain product/process repository. Rather than relying exclusively on informal means for project tracking and monitoring, Manco and Telco developed a database to help teams report issues, assign priorities, and track project status. However, in the spirit of an agile process, only minimal documentation was created.

Focus on well-understood functionality rather than critical new functionality. Agile methods emphasize delivering value to the customer early in the process and therefore often advocate the development of features prioritized as critical by the customer. However, in Manco, the project manager first wanted to build an atmosphere in which both the developers and the client representatives were acclimatized to the processes, tools, and the application. Therefore, the team worked on well understood functionality in the early iterations to ease the learning curve rather than on complex requirements even when they were prioritized to be more critical.

Short cycle but not time-boxed development. A common characteristic of most agile development projects is time-boxed development. Telco and Consult realized early on that this approach, while successfully practiced in their co-located teams, was not very successful with distributed development. In Telco, early iterations in three previous projects encountered difficulties because the development team did not have sufficient understanding of the business needs of the customers. In Consult, critical requirements identified by the clients in the first iteration were too broad in scope to be implemented within a short iteration. Both the organizations adopted a flexible short-cycle approach in which two to three development cycles were allowed to take two-to-four weeks each, depending on the complexity of the functionality and the setup time needed to understand the business domain.

HOW THE STUDY WAS CONDUCTED

We conducted detailed case studies of agile, distributed development in three companies (referred to here as Telco, Manco, and Consult) that have software development operations in India supporting customers located in the U.S. The objective was to understand the difficulties faced in managing such a development process and the practices designed to address them.

Company Characteristics			Typical Project Characteristics				
Company	Domain	Number of completed/ongoing agile distributed projects	Average project duration (months)	Application complexity	Number of development sites	Number of customer locations	Average development team size (people)
Telco	Telecommunications	6	6	Moderate	2	2	16
Manco	Supply chain	4	10	High	3	5	14
Consult	CRM	3	7	Moderate	3	3	15

Details about the companies and projects studied.

Data collection was carried out through open-ended (semi-structured) interviewing. We interviewed a total of 18 software developers, project managers, and customer representatives—one to three times each—from both locations in these companies over a period of nine months. We used grounded theory methodology [6] to analyze the data. This analysis identified five groups of practices and how they address challenges in distributed, agile development.

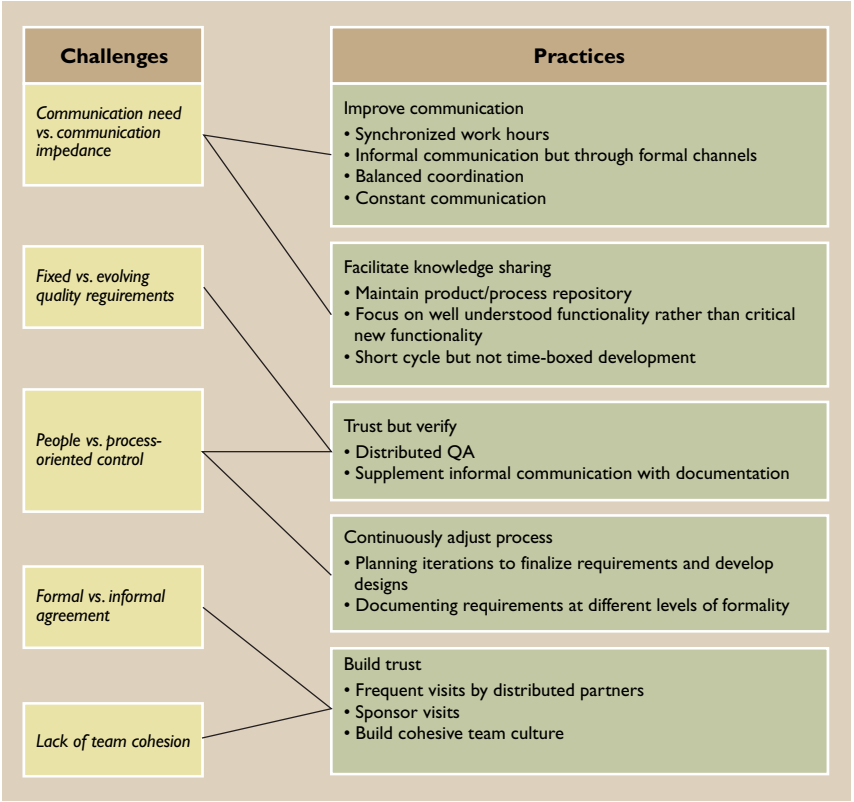
Telco is a large telecommunication company that is embarking on large-scale offshore development and maintenance of its applications. At the time of the study, the organization was completing several pilot projects aimed at identifying practices that can be rolled out throughout the organization.

Manco is a large manufacturing company that had just finished a project in its offshore development center. The project extended the functionality of a complex supply chain system.

Consult is a veteran in the global delivery model using offshore development. It was working with a U.S.-based client to develop a CRM system to support a complex customer network. It had dedicated a small facility exclusively for the U.S. customer, with the expectation of a significant long term partnership. The table here lists some of the salient characteristics of the companies and projects studied. C

3. Improve communication. The companies adopted several ways to improve the quality of communication among the team members.

Synchronized work hours. While 24x7 development is sometimes claimed to be a benefit of distributed development, this was far from reality in the three organizations. Separation by nine time zones created serious communication bottlenecks. Asynchronous work was found to be quite inconsistent with the constant communication needed for agile development. In order to "share the pain" of synchronizing work schedules, early morning meetings for onshore customer representatives and product managers and late evening shifts for offshore developers were the norm in all the three organizations.



Mapping between challenges and practices that address them.

Informal communication but through formal channels. There were several incidents of miscommunication among the distributed teams in Manco. These were attributed to over reliance on informal communication. To minimize this problem, one project lead was designated as the primary point of contact for each location. They were responsible for facilitating communication across the teams. Also, the organizational culture in Telco that values formal channels for communication demanded a very similar structure. In both organizations, informal communication among the team members was facilitated through formal channels.

Balanced coordination. Whereas typical agile development teams rely on minimal coordination of the team's activities by project managers, in all the three organizations, the coordination roles of project managers/leads were significantly more important. In Telco, the management maintained a formal structure of reporting and responsibility. Project leads and champions were expected to coordinate the activities of the offshore and onshore teams to help achieve project goals.

Constant communication. A variety of mechanisms were used in each organization to maintain constant communication. At Telco, short morning meetings were held each workday to identify issues, track project status and invite ideas and critiques. Also, the teams used online chat and Short Message Service (SMS) extensively. Project leads and champions at Consult were on call almost round-the-clock via their Blackberries. The instant availability, while considered beneficial, also was a significant burden to these individuals who were the designated gatekeepers of communication channels. Further, at both Telco and Manco, senior managers used videoconferencing on a weekly basis to initiate new development cycles, assess progress at the end of each cycle, and discuss critical issues.

4. Build trust. The companies realized that trust between teams is very essential in an agile distributed environment because there was minimal formal control. Several practices were used to build the trust between the teams.

Frequent visits by distributed partners. All three companies organized regular visits by customers or surrogates (product managers) with the development team. Consult instituted a practice of having a small group of analysts and developers in their customer sites. Telco and Manco organized visits to customer sites by developer representatives. In Manco, the complexity of the application required many more and longer visits than either group had anticipated. In Telco, key customer representatives spent several weeks in their offshore development center. In fact, one group stayed at the development center for four complete iterations. These visits were more intense in

THE PRACTICES THAT FACILITATED KNOWLEDGE SHARING HELPED DEVELOPERS GAIN A SHARED UNDERSTANDING OF THE BUSINESS DOMAIN AND THE FUNCTIONALITIES REQUIRED.

the early development cycles as well as when major changes were implemented.

Sponsor visits. Senior managers of the sponsoring organizations also visited the development teams at the beginning of the project to establish ground rules and finalize contractual arrangements (in Consult), and evaluate project progress at predetermined critical milestones (in Telco and Manco). These visits also helped establish trust among the senior members of the teams.

Build cohesive team culture. Telco created a cohesive team culture by requiring that each team was made up of members who had developed prior working relationships with each other and collectively possessed all the required expertise. The management insisted on such an arrangement to minimize what they perceived as chaos in agile approaches. Consult adopted a similar strategy and created high-performance teams. In all three organizations, the tremendous demand for qualified developers posed a significant challenge in retaining a cohesive team throughout the project.

5. Trust but verify. Deviating from agile methods, the companies introduced several practices to formally verify process and product quality.

Distributed QA. The senior management of Manco insisted that the onshore QA team be involved in ensuring that the practices followed by the offshore development team were of acceptable quality. During each iteration, this QA team reviewed the test procedures created by the offshore development team. In Telco, the onsite technical personnel participated in design reviews to ensure that the offsite development met their quality standards.

Supplement informal communication with documentation. In the spirit of agile development, even Telco, which traditionally favors a formal structure, adopted an informal atmosphere to facilitate collaboration among the geographically distributed team members. However, the management insisted on supplementing informal communication with documentation of critical artifacts. A similar practice was also considered necessary at Consult to formalize the vendor-client relationship. Often many documents were developed after the actual work was completed; for example, requirements documents were finalized toward the end of the development cycles rather than at the beginning so that they represent the actually implemented functionality rather than the initially conceptualized requirements that invariably changed during development.

Mapping Challenges to Practices

Each group of practices discussed here addresses the challenges faced while incorporating agility in distributed software development. Even though each group of practices addresses each of the challenges to a varying degree, here we illustrate how each challenge is addressed by the most relevant groups of practices. A summary of this mapping is provided in the figure on the previous page.

The communication need vs. communication impedance challenge is addressed by two groups of practices: improve communication and facilitate knowledge sharing. Improved communication achieved through a wide range of balanced channels for anytime anyplace access was the key to mitigating this challenge. These practices helped increase the efficiency and effectiveness of informal communication that was a challenge in a distributed environment. A limited amount of structure imposed on this communication was also useful in enhancing coordination among team members. The practices that facilitated knowledge sharing helped developers gain a shared understanding of the business domain and the functionalities required.

The fixed vs. evolving quality requirements challenge is addressed primarily by the trust but verify

THE TRUST BUILT BETWEEN THE TEAMS HELPED IN LIMITING THE FORMALITY WITH WHICH AGREEMENTS WERE SPECIFIED AND THUS ENABLED THE DEVELOPMENT TEAMS TO RAPIDLY ADAPT TO THE CHANGING NEEDS OF THE PROJECT.

group of practices. Distributed QA provides control over the quality of the system without a detailed contract upfront. Also, documentation of critical deliverables and processes helped ensure the evolving needs of quality were adequately addressed by the flexible process.

The people- vs. process-oriented control challenge is addressed by two groups of practices: continuously adjust the process and trust but verify. Continuous adjustments to the processes help ensure they are "just enough" for the current needs, without unduly constraining the development to a fixed process. Planned iterations helped in setting direction and in understanding the initial set of requirements. Requirements were documented at different levels of formality to provide a balance between the need for structure demanded by the distributed nature of development and the needs of agile project management.

The trust but verify practice allowed the organizations to keep a level of control that was considered essential to satisfy organizational norms and policies while at the same time providing necessary flexibility. The three organizations allowed the development to progress with the level of informality demanded by high-speed development, but built checks and balances to ensure that the process was under control.

The formal vs. informal agreement challenge is addressed by the build trust group of practices. The trust built between the teams helped in limiting the formality with which agreements were specified and thus enabled the development teams to rapidly adapt to the changing needs of the project.

The lack of team cohesion challenge is addressed by the build trust group of practices. While the site visits by sponsors and distributed partners added a significant overhead, they helped the customers and developers understand and trust the informal processes followed during development and build a cohesive team. Also, efforts to foster a cohesive team culture helped the teams operate with a common purpose even though they were geographically distributed.

Conclusion

Returning to the questions that guided our study, we conclude that careful incorporation of agility in distributed software development environments is essential in addressing several challenges to communication, control, and trust across distributed teams. The practices presented here demonstrate how a balance between agile and distributed approaches can help meet these challenges.

References

1. Beck, K. *Extreme Programming Explained: Embrace Change.* Addison-Wesley, Boston, 2000.
2. Ebert, C. and Neve, P.D. Surviving global software development. *IEEE Software 18,* 2 (Mar./Apr. 2001), 62–69.
3. Herbsleb, J.D. and Mockus, A. An empirical study of speed and communication in globally distributed software development. *IEEE Trans. on Software Engineering 29*, 6 (June 2003), 481–494.
4. Highsmith, J. and Cockburn, A. Agile software development: The business of innovation. *IEEE Computer 34,* 9 (Sept. 2001), 120–122.
5. Matloff, N. Offshoring: What can go wrong? *IT Professional* (July/Aug. 2005), 39–45.
6. Strauss, A. and Corbin, J. Grounded theory methodology: An overview. *Handbook of Qualitative Research.* N.K. Denzin and Y.S. Lincoln, Eds. Sage, London, 1994.

Balasubramaniam Ramesh (bramesh@gsu.edu) is a professor in the Department of Computer Information Systems, Georgia State University, Atlanta.
Lan Cao (lcao@odu.edu) is an assistant professor in the Department of Information Technology and Decision Sciences, Old Dominion University, Norfolk, VA.
Kannan Mohan (Kannan_mohan@baruch.cuny.edu) is an assistant professor in the Department of Computer Information Systems, Zicklin School of Business, Baruch College, New York City.
Peng Xu (peng.xu@umb.edu) is an assistant professor in the Department of Management Science and Information Systems, University of Massachusetts, Boston.