# Agile Methods and Virtual Distributed Teams

**Dr. David F. Rico, PMP, CSM**

Are Agile Methods and virtual distributed teams compatible? Aren't Agile Methods and virtual distributed teams polar opposites? Aren't they incompatible? And, if they are compatible, what are the specific techniques for making Agile Methods work for virtual distributed teams?

What does all of this mean? What is this controversy surrounding Agile Methods and virtual distributed teams? Why wouldn't Agile Methods be compatible with virtual distributed teams?

Well, all of this is related to the principles and values of Agile Methods as defined by the Agile Manifesto (http://www.agilemanifesto.org). The sixth of 12 major principles is as follows, "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation." It relates to two of the four major values of Agile Methods: (1) individuals and interactions and (2) customer collaboration. So, taken at face-value, no pun intended, one would be omitting half or 50% of the values of Agile Methods if one didn't use face-to-face conversations with customers and team members. Or, would they?

Face-to-face communication is contextually-rich, and is the preferred method of communication in Agile Methods. In Traditional Methods, statements-of-work, requests for proposals, contracts, requirements, and designs are considered contextually rich. But, nothing beats good old face-to-face communication when it comes to interacting with customers and team members. The creators of Agile Methods realized this and etched face-to-face communication in stone as one of their 12 commandments.

Okay, now, so anyone who can read a book on Agile Methods knows that face-to-face practices are mandatory. That is, anyone except those who are trying to make money in the global software industry. The global software industry is all about connecting or automating the supply chain of goods and services using products from China, computer programming labor from India, project managers from North America, and researchers in Europe. Well, that's simple, just co-locate everyone. Not so fast, why can't everyone stay exactly where they are and use electronic means to communicate? Well, that would involve violating the principles and values of Agile Methods, wouldn't it? In fact, if a team is not face-to-face, it's using Traditional Methods, not Agile Methods. Or, is it?

The creators of Agile Methods had part of the equation correct. Face-to-face communication is contextually rich, and is therefore superior to virtual distributed communication or software documentation in certain circumstances. However, the North American creators of Agile Methods failed to anticipate or acknowledge that 95% of the world wasn't collocated with them, and that North America wasn't even the predominant computer programming market. Didn't they heed the warnings of Ed Yourdon who said that the Indian software market was skyrocketing or how about Michael Cusumano's warning that Japan's Software Factories were taking over the planet? Perhaps they were caught up in the euphoria of Microsoft's and Intel's instant success in the 1990s. Or, the Internet Gold Rush convinced them that North American information technology dominance was here to stay?

I think the only ones who didn't believe the creators of Agile Methods were the 70 or so other countries besides the United States, who benefited from the Personal Computer and Internet revolution as well (along with any enterprising capitalist who wanted to make a quick-buck interconnecting the global supply chain).

The problem is that 99% of the population who knows anything about Agile Methods believes that one isn't being Agile if one isn't using face-to-face communication and all of the associated trappings (e.g., onsite customers, pair programmers, daily standup meetings, etc.).

Well, hang on to your seats, because the world is changing as we speak, and virtual distributed teams for Agile Methods are here to stay. We all have the benefit of being on the leading edge of this phenomenon as part of the next phase of software engineering evolution. I bet most people who are delving into Agile Methods didn't even realize they were on the cutting-edge of software engineering when they endeavored to use Agile Methods? Let's take a look at some of the evidence.

The folks at British Telecom have devised a set of practices for making Agile Methods work with virtual distributed teams (Cannizzo, Marcionetti, & Moser, 2008). They've identified four major practices to help marry Agile Methods with virtual distributed teams: (1) maximize project status visibility to all stakeholders, (2) ruthlessly automate as many development and management processes as possible, (3) ensure effective communications to the maximum extent possible, and (4) provide immediate feedback on every task performed. They use Eclipse IDE for writing code, Fitnesse and Selenium for acceptance testing, CruiseControl with Ant as a build tool, Danube ScrumWorks for user stories, Subversion for version control, and Atlassian Confluence Wiki for document sharing. Other tools include Visual Studio, NetBeans, Capistrano, Live Meeting, Communicator, Meet Me, and a variety of team building tools.

The folks at Yahoo! (Drummond & Unson, 2008) suggest mandatory meetings within time zones, periodic face-to-face meetings across time zones, overlapping virtual distributed meetings where possible, periodic synchronization between international Scrum Masters, the use of Wikis to share photographs of user stories on Post It notes, and ensuring that virtual distributed teams maintain the rank-ordering or priority of user stories. Furthermore, Yahoo! says to emphasize individuals and interactions over processes and tools as much as humanly possible, factor in the needs of the global workforce over individual practices of Agile Methods, and ensure that accurate information is shared in a timely fashion.

The folks in Canada (Robarts, 2008) say to have periodic face-to-face meetings as much as possible (especially during project kickoff), periodically send people back and forth from one country or location to another, ensure domain experts periodically visit one another, ensure delivery teams occasionally visit one another face-to-face, hold virtual distributed daily standup meetings between people in the same time zone, exchange informal notes such as PowerPoint presentations as meeting minutes, and ensure Wikis and other automated tool content is constantly up-to-date. The Canadians remind us to emphasize the use of Agile Methods practices like User Stories (instead of specifications), use video conferences as much as possible, and use portable information technologies such as laptops, cell phones, and personal digital assistants (e.g., BlackBerries). More importantly, proper schedule management is of utmost importance, especially when it comes to building in management reserve, accounting for international Holidays and unplanned events, and keeping the schedules current as well as communicating them. Interestingly enough, we're advised to allow enough latitude for differences in individual Agile Method practices across international boundaries.

The list goes on and on, as the Indians recommend automated dashboards and wikis (Shrinivasavadhani & Panicker, 2008). The Americans emphasize WebEx Meetings, periodic virtual distributed meetings between international Scrum Masters, and a variety of techniques such as balancing power, allowing for variation in practices, detailed User Stories, empowerment, sharing vision statements, executive commitment, and as much communication as possible (Therrien, 2008). Another Canadian team says to use nearshore resources, maintain strict communication plans, share electronic workspaces, use synchronous communications only in adjacent time zones, and use asynchronous retrospectives (Vax & Michaud, 2008). Finally, a group of Americans and Canadians recommend instant messaging, synchronous communication mechanisms, video conferencing, face-to-face virtual distributed meetings, virtual distributed standup meetings, and the list goes on and on (Young & Terashima, 2008). You don't want to skip this last paper, because it contained a lot of helpful advice.

So, what's the bottom line? The creators of Agile Methods had it right, face-to-face communication is contextually rich and face-to-face is the preferred method of communication. And, a lot of face-to-face communication is a key, if not the key, to project success. I speak from experience on this one, because one of the most difficult projects I've ever managed, was success due to frequent, unscheduled face-to-face communication to break down the barriers to personal trust. Once the barriers to trust had been breached through frequent personal interaction, then the project completed successfully. We were given an award by a customer who had never given an award to a consulting firm before. You just don't find that kind of advice in your typical textbook on Traditional Methods. So, how do we duplicate the benefits of face-to-face interaction in virtual distributed teams? Communicate, frequently and often! Outside of video teleconferencing, telephone calls have been one of the best forms of communication over the last century. It's the "unscheduled" telephone calls that provide the most return-on-investment (versus regularly scheduled telephone calls that just don’t seem to have the same effects). Ad hoc telephone calls help alleviate anxiety instantly, as opposed to scheduled phone calls. Intimate, personal communications break down barriers to trust and ensure project success almost every time.

## REFERENCES

Cannizzo, F., Marcionetti, G., & Moser, P. (2008). Evolution of the tools and practices of a large distributed agile team. *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada,* 513-518.

Drummond, B. S., & Unson, J. F. (2008). Yahoo distributed agile: Notes from the world over. *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada*, 315-321.

Robarts, J. M. (2008). Practical considerations for distributed agile projects. *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada*, 327-332.

Shrinivasavadhani, J., & Panicker, V. (2008). Remote mentoring a distributed agile team. *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada*, 322-326.

Therrien, E. (2008). Overcoming the challenges of building a distributed agile organization. *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada*, 368-372.

Vax, M., & Michaud, S. (2008). Distributed agile: Growing a practice together. *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada*, 310-314.

Young, C., & Terashima, H. (2008). How did we adapt agile processes to our distributed development? *Proceedings of the Agile Conference (Agile 2008), Toronto, Canada*, 304-309.

# Practices of Agile Methods and Virtual Distributed Teams

| Author | Category | Practice | Technique |
|---|---|---|---|
| 1.0 Cannizzo, Marcionetti, & Moser, 2008 | **1.1 Provide maximum project status visibility** | Enable stakeholders to see the status of the project in any desired detail. | Automation |
| | | Enable managers and customers to see the status of user stories. | Automation |
| | | Enable managers and customers to see user stories and backlogs. | Automation |
| | | Enable teams to see the properties of the code being written. | Automation |
| | | Enable teams to see whether software builds or passes tests. | Automation |
| | | Enable teams to see whether software meets code metric thresholds. | Automation |
| | **1.2 Ruthlessly automate everything** | Automate as many development and management processes as possible. | Automation |
| | | Automate the running of unit tests every time a source file is saved. | Automation |
| | | Automate the running of metrics-gathering tools on the fly. | Automation |
| | | Automate the running of back-end integration tests. | Automation |
| | | Automate deployment of the Web portals to the production environment. | Automation |
| | | Automate tasks around confirming whether something is release-ready. | Automation |
| | **1.3 Institute effective communication** | Enable information exchange with minimal waiting and misunderstanding. | Conferencing |
| | | Enable teams to co-locate three days per week. | Periodic Collocation |
| | | Enable teams to co-locate for one or two iterations per release cycle. | Periodic Collocation |
| | **1.4 Enable teams to obtain immediate feedback** | Enable teams to quickly know outcomes or side effects of completed tasks. | Automation |
| | | Enable teams to address problems as they occur. | Automation |
| | | Enable teams to quickly know whether completed code quality is sufficient. | Automation |
| | | Enable teams to quickly gather feedback on any development activity. | Automation |
| | | Enable teams to make use of IDE plug-ins to gathering of metrics on the fly. | Automation |
| | | Enable teams to easily run only a subset of unit tests. | Automation |
| | | Enable teams to produce integration tests to pinpoint the cause of faults. | Automation |
| | | Enable teams to minimize clashes and merges of code and other artifacts. | Automation |
| | | Enable teams to synchronize source code builds several times per day. | Automation |
| | | Enable teams to negotiate the priority of user stories and evaluate backlogs. | Empowerment |
| | **1.5 Use Eclipse to automate development** | Use Eclipse as a framework for building rich client applications. | Automation |
| | | Use Eclipse with JUnit, Subversion, Emma, CheckStyle, and FindBugs. | Automation |
| | | Use Eclipse as the editor of choice for working with the Java language. | Automation |
| | **1.6 Use Fitnesse and Selenium to automate acceptance testing** | Use Fitnesse and Selenium to implement customer acceptance test suites. | Automation |
| | | Use Fitnesse to test application programming interfaces (APIs). | Automation |
| | | Use Selenium to test web front-ends. | Automation |
| | | Use Fitnesse to allow developers to write tests using wiki syntax. | Automation |
| | | Use Fitnesse to make tests available on web pages to be run on demand. | Automation |
| | | Use Fitnesse to include test suites for specific types of environments. | Automation |
| | | Use Fitnesse to create test suites configured for different environments. | Automation |
| | | Use Fitnesse to run integration tests for both development and production. | Automation |
| | **1.7 Use CruiseControl with Ant to automate software builds** | Use Apache Ant to execute build scripts for Java applications. | Automation |
| | | Use Apache Ant to write complex and flexible build scripts for reuse. | Automation |
| | | Use Apache Ant to manage continuous builds. | Automation |
| | | Use Apache Ant to build projects in different programming languages. | Automation |
| | | Use Apache Ant to provide an administration console. | Automation |
| | | Use Apache Ant to provide reporting for managing builds and build artifacts. | Automation |
| | | Use Apache Ant to automate builds for running performance tests. | Automation |
| | **1.8 Use Danube ScrumWorks to automate user stories** | Use ScrumWorks to support the adoption of the Scrum agile methodology. | Automation |
| | | Use ScrumWorks to allow teams to manage their backlog on a virtual wall. | Automation |
| | | Use ScrumWorks to manage the Scrum backlogs, teams, and products. | Automation |
| | | Use ScrumWorks to provide reporting facilities for accessing historical data. | Automation |
| | | Use ScrumWorks to serve as a reporting front end. | Automation |
| | | Use ScrumWorks to retrieve information during retrospectives. | Automation |
| | **1.9 Use Subversion for version control** | Use Subversion as a software source code control system. | Automation |
| | | Use Subversion as a repository via several protocols. | Automation |
| | | Use Subversion with Tortoise SVN, Subversive, and Subclipse plug-ins. | Automation |
| | **1.10 Use Atlassian for documentation** | Use Atlassian Confluence Wiki to share and produce documents. | Shared Workspaces |
| | | Use Atlassian Confluence Wiki for collaborative editing. | Shared Workspaces |
| | | Use Atlassian Confluence Wiki for rich content using add-ons. | Shared Workspaces |
| | **1.11 Use other tools for automation** | Use Visual Studio (for .NET development) and NetBeans (for Ruby). | Automation |
| | | Use Capistrano to automate deployment of Ruby on Rails applications. | Automation |
| | | Use Microsoft Live Meeting, Microsoft Communicator, and BT Meet Me. | Conferencing |
| | | Use Foosball table, games console, break-out area with sofas, and beanbags. | Process Flexibility |
| 2.0 Drummond & Unson, 2008 | **2.1 Don't fall into the easy trap** | Don't follow the Scrum practices religiously. | Process Flexibility |
| | | Don't require teams in different time zones to attend all Scrum meetings. | Process Flexibility |
| | | Don't follow Scrum to the exclusion of cultural and religious differences. | Cultural Sensitivity |
| | | Don't use Scrum to make international teams feel like second class citizens. | Empowerment |
| | | Don't hinder the adoption of Scrum by alienating international teams. | Empowerment |

| Author | Category | Practice | Technique |
|---|---|---|---|
| | | Don't limit international interactions to phone and video meetings. | Periodic Collocation |
| | | Don't ask international teams to make unnecessary personal sacrifices. | Cultural Sensitivity |
| | **2.2 Bridge the distributed divide** | Use Scrum retrospectives to identify issues affecting international teams. | Empowerment |
| | | Use Scrum retrospectives to smoke out inequitable practices. | Empowerment |
| | | Use informal chats with Scrum Masters to augment retrospectives. | Empowerment |
| | | Use face-to-face, onsite training to further identify inequitable practices. | Periodic Collocation |
| | | Have Scrum Masters and Agile Coaches regularly communicate. | Touch Points |
| | **2.3 Go local** | Adopt "go-local" rule to remove having to attend early or late meetings. | Process Flexibility |
| | | Allow each team to do their meetings at their own time zones. | Process Flexibility |
| | | Institute key touch points between senior members of the team. | Touch Points |
| | | Have managers at multiple locations meet two or three times a week. | Touch Points |
| | | Alternate the schedule of who will wake up early and stay up late each week. | Process Flexibility |
| | | Remove the pain of having the teams attend lengthy late or early meetings. | Process Flexibility |
| | **2.4 Institute periodic face-to-face meetings** | Foster bonding and collaboration by having face-to-face, quarterly meetings. | Periodic Collocation |
| | | Alternate location of face-to-face, quarterly release planning meetings. | Periodic Collocation |
| | | Have everyone together in one location for at least one occasional sprint. | Periodic Collocation |
| | | Equalize the pain of having to travel 24 hours back and forth across sites. | Periodic Collocation |
| | | Allow people to connect the faces to the voices they hear on the phone. | Periodic Collocation |
| | **2.5 Make some immediate adaptations** | Adopt a strict time limit on sprint planning and review meetings. | Conferencing |
| | | Use the first hour to review prior deliverables and results of retrospectives. | Conferencing |
| | | Allow exchange of stakeholder feedback between managers and developers. | Empowerment |
| | | Reserve the second hour for discussing the details of the product backlog. | Conferencing |
| | | Allow the teams to review backlog items and ask detailed questions. | Empowerment |
| | | Enforce meeting end times, as well user stories, priorities, and commitments. | Conferencing |
| | | Repeat the planning meeting at remote sites to identify tasks and estimates. | Touch Points |
| | | Synchronize any local adjustments to sprint backlogs at the end of the day. | Process Flexibility |
| | **2.6 Use local proxies** | Synchronize communications between product owners and Scrum masters. | Touch Points |
| | | Discuss the sprint progress and backlog status at least three times a week. | Touch Points |
| | | Use local Scrum Masters as alternate product owners to answer questions. | Empowerment |
| | **2.7 Get everyone on the same page up-front** | Provide training in agile methods to the whole team at beginning of project. | Coaching |
| | | Use the same agile methods trainer to maximize consistency. | Coaching |
| | | Embed agile coaches within each team at each location. | Coaching |
| | | Hold daily meetings among the agile coaches at each location. | Coaching |
| | | Hold frequent voice and email exchanges among agile coaches. | Coaching |
| | **2.8 Replicate information radiators** | Use highly visible post-it notes for communicating sprint information. | Shared Workspaces |
| | | Use shared easily editable team Wiki pages to communicate progress. | Shared Workspaces |
| | | Keep shared information constantly updated to maintain interest in activities. | Shared Workspaces |
| | | Produce current status and information using simple web page formats. | Shared Workspaces |
| | | Distribute digital photographs of information radiators on wikis. | Shared Workspaces |
| | | Use understandable identifiers and labels for user stories and tasks. | Shared Workspaces |
| | | Make shape of task board and movement of stickies easy to follow. | Shared Workspaces |
| | | Emphasize use of visible user stories and tasks on post it notes. | Shared Workspaces |
| | | Produce frequent feedback by instituting two instead of four week sprints. | Process Flexibility |
| | | Synchronize end-of-sprint reviews across international locations if possible. | Periodic Collocation |
| | | Use Adobe Connect for voice and video conferencing services. | Conferencing |
| | | Use localized retrospectives and sharing the results with all teams. | Touch Points |
| | **2.9 Miscellaneous practices** | Maintain the priority of user stories by disallowing cherry picking. | Empowerment |
| | | Separate backlogs dedicated to regional customizations and considerations. | Process Flexibility |
| | | Use Scrum of Scrums for upper-level coordination of multiple backlogs. | Process Flexibility |
| | | Standardize communication, increasing visibility, and automate releases. | Process Flexibility |
| | | Increase communication speed by providing less detail at the task level. | Process Flexibility |
| | | Provide consistent and authoritative training messages from coaches. | Coaching |
| | **2.10 Avoid common pitfalls** | Focus on individuals and interactions over processes and tools. | Process Flexibility |
| | | Don't allow teams to fall into the "Scrum-by-the-book" syndrome. | Process Flexibility |
| | | Don't cause resentment by having meetings at inconvenient times. | Process Flexibility |
| | | Don't use Scrum practices as a "big-stick" to alienate teams. | Process Flexibility |
| | | Make a genuine effort to factor in the needs of people. | Process Flexibility |
| | | Adapt Scrum processes, minimize pain, and place value on people. | Process Flexibility |
| | | Institute measures to close gaps in customer interaction and collaboration. | Empowerment |
| | | Identify individuals who have the abilities to succeed in distributed teams. | Personnel Selection |
| 3.0 Robarts, 2008 | **3.1 Use face-to-face visits and rotations** | Plan for rotations through each site on a regular basis. | Periodic Collocation |
| | | Get the entire delivery team together to kick off the project. | Periodic Collocation |
| | | Choose a location convenient for the majority of the team for the kickoff. | Periodic Collocation |
| | | Use face-to-face kickoff to meet, establish a rapport, and understand project. | Periodic Collocation |
| | | Use a face-to-face kickoff to help everyone feel like part of the same team. | Periodic Collocation |
| | | Invite the client or the customer to the face-to-face kickoff as well. | Periodic Collocation |
| | | Use a face-to-face kickoff to help the client become more trusting. | Periodic Collocation |
| | | Schedule periodic face-to-face exchanges in the case of a limited budget. | Periodic Collocation |

| Author | Category | Practice | Technique |
|---|---|---|---|
| | | Use face-to-face release planning with clients, product owners, and leads. | Periodic Collocation |
| | | Invest in face-to-face meetings by personnel who represent everyone. | Periodic Collocation |
| | | Schedule visits for product owners and team leads throughout each release. | Periodic Collocation |
| | | Use periodic face-to-face meetings to establish norms, rules, and protocols. | Periodic Collocation |
| | | Use face-to-face meetings to increase awareness of dedication and loyalty. | Periodic Collocation |
| | | Arrange for members of the delivery team to rotate between locations. | Periodic Collocation |
| | | Rotate team members between locations to share practices and customs. | Periodic Collocation |
| | | Learn international visa rules and limitations before release planning begins. | Periodic Collocation |
| | **3.2 Communicate progress with conference call standups** | Hold conference calls every day within the same time zone. | Conferencing |
| | | Hold conference calls every other day across different time zones. | Conferencing |
| | | Hold conference calls with only the leaders of very large teams. | Conferencing |
| | | Hold conference calls by rotating the leaders of very large teams. | Conferencing |
| | | Hold conference calls by sharing knowledge among leaders in large teams. | Conferencing |
| | | Hold conference calls with strict time limits. | Conferencing |
| | **3.3 Follow up in writing** | Follow up verbal messages with a written versions (using PowerPoint). | Process Flexibility |
| | | Update all project artifacts, wikis, and tracking tools prior to a status calls. | Shared Workspaces |
| | **3.4 Communicate business needs by writing less** | Use video conferences to present a high-level vision of the application. | Conferencing |
| | | Provide a local installation of the application for demonstration purposes. | Process Flexibility |
| | | Provide video taped messages recorded by subject matter experts. | Coaching |
| | | Provide wireframes and lo-fi prototypes to communicate requirements. | Process Flexibility |
| | **3.5 Build contingency reserves into schedules** | Take international holidays into account to minimize disruptions. | Cultural Sensitivity |
| | | Take international vacations into account to minimize disruptions. | Cultural Sensitivity |
| | | Take international customs, cultures, habits, and behaviors into account. | Cultural Sensitivity |
| | | Take international seasonal weather events and phenomenon into account. | Cultural Sensitivity |
| | | Take international personal time needs and preferences into account. | Cultural Sensitivity |
| | **3.6 Build common understanding of practices** | Assess differences in Agile Methods terms and practices across locations. | Process Flexibility |
| | | Set up wiki pages to capture terminology, references, and standard practices. | Shared Workspaces |
| | | Provide training in Agile Methods and standard practices for new members. | Coaching |
| | | Allow teams to select their standard practices from a repository of templates. | Process Flexibility |
| 4.0 Shrinivasavadhani & Panicker, 2008 | **4.1 Use a product roadmap and stakeholder meetings** | Develop a product roadmap and circulate it to all of the stakeholders. | Process Flexibility |
| | | Constantly evaluate new features and align them with the roadmap. | Process Flexibility |
| | | Hold weekly meetings with the stakeholders to update them on the status. | Touch Points |
| | | Share feedback from weekly stakeholder meetings with the rest of the team. | Touch Points |
| | | Refine and update Wiki contents prior to all discussions and meetings. | Shared Workspaces |
| | **4.2 Enhance collaboration and productivity with heavy automation** | Perform early integration and daily builds using Maven and Continuum. | Automation |
| | | Gradually adjust build frequency to suit the team's pace within the test bed. | Automation |
| | | Enhance collaboration by using automated modeling tools. | Process Flexibility |
| | | Minimize effort by using automated modeling tools for documentation. | Process Flexibility |
| | | Use tools such as project websites, Bugzilla, and source code repositories. | Automation |
| | | Use WaccPlanner and Xplanner for planning, tracking, metrics, and reports. | Automation |
| | | Use Wiki and Wink to facilitate discussions within the teams. | Shared Workspaces |
| | **4.3 Split up the work between teams with greater and lesser skills and capabilities** | Split up the work to help lesser skilled teams gain some momentum. | Process Flexibility |
| | | Perform release planning to divide the work into non-interdependent parts. | Process Flexibility |
| | | Allow lesser skilled remote teams to work on lower priority tasks. | Process Flexibility |
| | | Use separate branches of a common source code repository. | Process Flexibility |
| | | Gradually merge more and more code between disparate branches. | Process Flexibility |
| | | Use identical project environments to facilitate seamless integration. | Automation |
| | | Use remote mentoring as an effective way to ramp up a distributed team. | Coaching |
| | **4.4 Select remote mentors** | Select remote mentors based on expertise with the product features. | Coaching |
| | | Select remote mentors based on their abilities to effectively communicate. | Coaching |
| | | Select remote mentors based on their abilities to maintain active visibility. | Coaching |
| | | Select remote mentors based on technical expertise and domain knowledge. | Coaching |
| | | Select remote mentors based on their ability to perform quality assurance. | Coaching |
| | | Select remote mentors based on their ability to focus on broad issues. | Coaching |
| | **4.5 Select local mentors** | Select local mentors based on technical capabilities and domain knowledge. | Coaching |
| | | Select local mentors based on ability to participate in pair programming. | Coaching |
| | | Select local mentors based on their ability to interface with remote mentors. | Coaching |
| | | Select local mentors based on ability to participate in face-face discussions. | Coaching |
| | | Select local mentors based on ability to focus only the local team's release. | Coaching |
| | **4.6 Establish a remote mentoring process** | Establish a remote mentoring process to help lesser skilled remote teams. | Coaching |
| | | Use remote mentoring to level skill sets, build trust, and create confidence. | Coaching |
| | | Collocate teams for kickoff, to discuss goals, and establish relationships. | Periodic Collocation |
| | | Assign mentors based on knowledge gained from working with all teams. | Coaching |
| | | Ensure that remote mentors have full visibility into the work of the teams. | Coaching |
| | | Assign about 20% of the effort for remote mentors to perform their roles. | Coaching |
| | | Use written documents and diagrams to bridge culture and language barriers. | Process Flexibility |
| | | Post documents, models, and diagrams in Wikis to enhance communications. | Shared Workspaces |
| | **4.7 Use remote and** | Use a project website to maintain dashboards that contain details of teams. | Automation |

| Author | Category | Practice | Technique |
|---|---|---|---|
| | **local mentoring activities** | Post the features for all releases in the project website for both teams. | Shared Workspaces |
| | | Hold multiple joint rounds of collaborative meetings to establish user stories. | Periodic Collocation |
| | | Hold multiple joint rounds of collaborative meetings to validate the scope. | Periodic Collocation |
| | | Use the centralized project website to store and retrieve user stories. | Shared Workspaces |
| | | Use local mentors to develop, refine, and post development tasks. | Coaching |
| | | Use local mentors to ensure collaboration and frequent update of status. | Coaching |
| | | Use local mentors to send updated status using WaccPlanner (tracking tool). | Coaching |
| | | Use remote mentors to identify issues and technical flaws early on. | Coaching |
| | | Use local mentors to ensure the quality of code under development. | Coaching |
| | | Establish and hold daily standup meetings at each location. | Process Flexibility |
| | | Ensure teams communicate on a daily basis and use instant messaging. | Conferencing |
| | | Post all clarifications using wikis in the form of diagrams instead of text. | Shared Workspaces |
| | | Use local mentors upload task breakdown into Waccplanner for user stories. | Coaching |
| | | Regularly update the status of teams in Waccplanner to provide visibility. | Automation |
| | | Do not require daily stand-ups if there are significant language barriers. | Process Flexibility |
| | | Check the deliverables into a source code repository to increase visibility. | Automation |
| | | Have both mentors take corrective action based on early visibility of status. | Coaching |
| | | Use updated project status, velocity, and test status to gauge the progress. | Automation |
| | | Use wikis to capture details of the look and feel of the diagrams. | Shared Workspaces |
| 5.0 Therrien, 2008 | **5.1 Get team buy-in at local and remote locations** | Include both local and remote teams in backlog scrubbing. | Empowerment |
| | | Allow all teams to participate in critical activities such as backlog scrubbing. | Empowerment |
| | | Allow all team members time to achieve an optimum level of comfort. | Cultural Sensitivity |
| | | Allow for cultural adaptations for consensus vs. individual decision making. | Cultural Sensitivity |
| | | Allow all teams to interface product owners, clients, and stakeholders. | Empowerment |
| | **5.2 Adapt Scrum meetings for working with local and remote teams** | Be flexible until a balance of home and work life is achieved. | Process Flexibility |
| | | Ensure that product owners attend at least two stand-up meetings per week. | Touch Points |
| | | Ensure that teams participate in backlog scrubbing meetings each week. | Empowerment |
| | | Allow teams to ask product owners for clarification on as-needed basis. | Empowerment |
| | | Ensure product owners update content in tools such as Version One. | Automation |
| | | Keep backlog scrubbing meetings down to a manageable time. | Conferencing |
| | **5.3 Adapt Scrum Master roles** | Charge local Scrum Masters with ensuring active participation by all teams. | Empowerment |
| | | Allow all teams to identify, estimate, and managing user stories and tasks. | Empowerment |
| | | Have product owners focus on long term planning, vision, and strategy. | Process Flexibility |
| | | Allow product owners to host Sprint planning and review meetings. | Touch Points |
| | **5.4 Adapt Product Owner roles** | Ask product owners to provide additional requirements (if necessary). | Process Flexibility |
| | | Ask product owners to document acceptance criteria (if necessary). | Process Flexibility |
| | | Ask product owners to exchange emails for sizing, scoping, and estimating. | Process Flexibility |
| | | Ask product owners to add additional content to tools such as Version One. | Process Flexibility |
| | **5.5 Adapt Product Owner interactions** | Ask product owners to conduct quarterly product roadmap meetings. | Process Flexibility |
| | | Ask product owners to communicate vision using WebEx and PowerPoint. | Process Flexibility |
| | | Ask product owners to communicate requirements as categories and epics. | Process Flexibility |
| | | Ask product owners to facilitate brainstorming and improvement sessions. | Empowerment |
| | | Ask product owners to travel to all locations to ensure project progress. | Periodic Collocation |
| | **5.6 Facilitate communication with tools** | Facilitate communications by using automated tools such as Version One. | Automation |
| | | Supplement user stories with requirements documents in automated tools. | Process Flexibility |
| | | Use WebEx and conference calls as the primary methods of interaction. | Conferencing |
| | **5.7 Maintain an equitable balance of power** | Adapt product owner and Scrum Master roles to maintain balance of power. | Empowerment |
| | | Don't disempower teams with localized product owners and Scrum Masters. | Empowerment |
| | | Allow teams to enjoy a sustained work pace and normal work hours. | Empowerment |
| | | Allow teams to fully participate in the Scrum process. | Empowerment |
| | **5.8 Adapt Scrum practices and guidelines for local and remote teams** | Adapt Scrum processes and practices to meet the needs of all teams. | Process Flexibility |
| | | Supplement user stories with detailed specifications (when necessary). | Process Flexibility |
| | | Provide detailed requirements to minimize dependency on product owners. | Process Flexibility |
| | | Use consultants to facilitate retrospectives and identify critical issues. | Coaching |
| | | Invest in training and coaching on an as-needed basis. | Coaching |
| | | Enforce commitment by regular interaction with product owners. | Touch Points |
| | | Use visuals, mockups, diagrams, online meetings, and video conferencing. | Process Flexibility |
| | | Satisfy user needs and delight customers, not follow Scrum religiously. | Process Flexibility |
| 6.0 Vax & Michaud, 2008 | **6.1 Adapt Agile Methods for a distributed environment** | Allow teams to use documentation and multiple communication channels. | Process Flexibility |
| | | Allow teams to use status tracking and reporting tools and other meetings. | Automation |
| | | Allow teams to be initially collocated and then allow them to separate. | Periodic Collocation |
| | | Use shared source code management tools and a bug tracking systems. | Shared Workspaces |
| | | Use standardized iteration planning and daily status update processes. | Process Flexibility |
| | **6.2 Use near shore resources as Scrum Masters** | Use near shore resources who can meet face-to-face at regular intervals. | Process Flexibility |
| | | Keep lines of communication open between leads and Scrum Masters. | Touch Points |
| | | Provide overlapped communication time between leads and Scrum Masters. | Touch Points |
| | | Allow Scrum Masters to coordinate acceptance testing activities. | Touch Points |
| | **6.3 Implement a** | Hold Scrum meetings three times per week between international teams. | Process Flexibility |

| Author | Category | Practice | Technique |
|---|---|---|---|
| | strict communication plan | Allow teams to work from home by using common a computing toolset. | Automation |
| | | Use tools such as instant messaging and VoIP teleconferencing. | Conferencing |
| | | Always provide instant informal feedback by email within 12 hours. | Conferencing |
| | 6.4 Use shared electronic work spaces | Provide a persistent workspace for critical project data such as SharePoint. | Shared Workspaces |
| | | Use SharePoint for status, announcements, discussions, and documents. | Shared Workspaces |
| | | Use common source code control systems and bug tracking systems. | Automation |
| | | Provide local system support for shared development servers. | Shared Workspaces |
| | 6.5 Manage work effectively across sites | Divide work across no more than two time zones. | Process Flexibility |
| | | Involve the entire team of a single time zone in synchronous retrospectives. | Process Flexibility |
| | | Involve only the leads and Scrum masters in synchronous retrospectives. | Process Flexibility |
| | 6.6 Get the right people communicating | Assign product owners and Scrum masters according to their abilities. | Personnel Selection |
| | | Allow Scrum Masters to talk and developer-to-developer communications. | Touch Points |
| | | Do not restrict the flow of communications to the organizational hierarchy. | Empowerment |
| | 6.7 Implement the appropriate planning mechanisms | Budget for occasional face-to-face travel, meetings, and communications. | Periodic Collocation |
| | | Provide comprehensive training, coaching, instruction, and mentoring. | Coaching |
| | | Refactor the code base for more modularity and independence among teams. | Process Flexibility |
| | | Base teams on personality, temperament, experience, skills, and education. | Personnel Selection |
| 7.0 Young & Terashima, 2008 | 7.1 Place an emphasis on effective communication mechanisms | Use instant messaging for quick feedback as much as possible. | Conferencing |
| | | Use Skype or iChat to have spontaneous meetings to discuss issues. | Conferencing |
| | | Make an effort to overlap work schedules across all the teams. | Conferencing |
| | | Establish a goal of being available synchronously as much as possible. | Conferencing |
| | | Use desktop sharing combined with video conferencing for discussing code. | Conferencing |
| | | Use Virtual Network Computing (VNC) for pair programming sessions. | Conferencing |
| | | Use VNC to leverage file-indexing, bookmarking, and debugging. | Conferencing |
| | | Communicate using video conferencing for code-related reviews. | Conferencing |
| | | Take breaks to ensure everyone is following along with code reviews. | Conferencing |
| | | Use video conferences for conversation, participation, and high context. | Conferencing |
| | | Have virtual stand up meetings as much as possible (if schedules permit). | Process Flexibility |
| | | Hold weekly meetings with customers to negotiate priority of user stories. | Touch Points |
| | | Use wikis to post weekly meeting agendas along with the meeting minutes. | Shared Workspaces |
| | | Use asynchronous communication tools such as wikis. | Shared Workspaces |
| | | Use bug management systems, newsgroups, and email to log project history. | Automation |
| | | Create documents to describe the effort and time line for each team. | Process Flexibility |
| | | Communicate the business value of new features and changes to all teams. | Process Flexibility |
| | | Use a wiki page to merge and prioritize all of the work into one backlog. | Shared Workspaces |
| | | Enter all tasks into a shared bug ticketing system. | Automation |
| | 7.2 Build a sense of trust with remote team members | Use face-to-face kickoffs and sprints to allow the teams to get acquainted. | Periodic Collocation |
| | | Incorporate a short greeting period at the beginning of each meeting. | Conferencing |
| | | Discuss ordinary subjects such as the weather, wellness, and other small talk. | Conferencing |
| | | Incorporate physical surroundings into video conferences to ensure realism. | Conferencing |
| | | Allow everyone to relax as a pleasant way of starting meetings. | Conferencing |
| | | Have clear agenda and an expected duration time for all meetings. | Conferencing |
| | | Frequently pause for questions and clear up any misconceptions. | Conferencing |
| | | Strive to maintain real-time two-way communication and for realism. | Conferencing |
| | | Bring all participants into video conferences to resolve miscommunications. | Conferencing |
| | | Provide an introductory video conference on Agile Methods practices. | Coaching |
| | | Be mindful of etiquette rules associated with different cultures. | Cultural Sensitivity |
| | | Respect the holidays of all teams and don't disturb them in these periods. | Cultural Sensitivity |
| | | Learn as much as you can about the various cultures involved in projects. | Cultural Sensitivity |
| | | Arrange meetings at reasonable times because of time zone differences. | Process Flexibility |
| | | Obtain consensus on scheduling of subsequent follow-up meetings. | Empowerment |
| | 7.3 Software Architecture, Tools, and Design Approaches | Maintain a wiki to document the application programming interfaces (APIs). | Shared Workspaces |
| | | Create a set of functional tests to verify the API against specifications. | Automation |
| | | Use functional tests for continuous integration to provide quick feedback. | Automation |
| | | Hold video conferences with all teams to discuss changes and updates. | Conferencing |
| | | Use a single code base for all teams that can be locally customized. | Process Flexibility |
| | | Program to interfaces and not to implementations when using the code base. | Process Flexibility |
| | | Enforce strict coding standards that require coding components to interfaces. | Process Flexibility |
| | | Leverage Spring Framework containers to help loosely couple systems. | Process Flexibility |
| | | Use polymorphism as a way of extending components when necessary. | Process Flexibility |
| | | Collaborate on tasks together when working on cross-country requirements. | Touch Points |
| | | Use Maven 2 for development and Ant as an auxiliary Java utility. | Automation |
| | | Use Eclipse as and IDE and with plug-ins for Maven and Ant. | Automation |
| | | Use Ant to run any command on multiple operating systems. | Automation |
| | | Ensure everything is automated and repeatable using Ant or Maven. | Automation |
| | | Help newcomers by using an automated development environment. | Automation |