Distributed Agile Development at Microsoft patterns & practices

Ade Miller, Microsoft patterns & practices

October 2008

Abstract

Distributed development is a fact of life for many teams. Unfortunately most agile methodologies or approaches assume that the team is located in a single team room. Until recently there has been little guidance about how to apply these approaches with a geographically dispersed team.

Microsoft's patterns & practices group has been following an agile, distributed development approach for the past five years. During this time teams within the group have experimented extensively with different approaches to best address the challenges of distributed agile development. This paper outlines the challenges faced by geographically distributed agile teams and details some proven practices to address these issues and build successful distributed teams.

Contents

Introduction

In *The 2008 State of Agile Development* survey, conducted by VersionOne,[i] 57% of respondents stated that their teams were distributed. Furthermore 41% of respondents said that they were currently using, or plan to combine, agile with outsourced development. When faced with these sorts of numbers it seems that the agile practice of placing the entire team in a single room is at odds with what's actually going on within a large part of the software development community. While colocating your team is clearly the recommended approach many teams are unable to do this and are faced with trying to stick to their agile principles and apply agile practices to distributed development.

This paper describes some of the proven practices about distributed agile development and the specific lessons learned by teams within Microsoft's patterns & practices[1] group over numerous agile projects using distributed teams.

While this paper focuses on geographically distributed agile teams, many of the practices discussed here can also be applied to teams who adopt other methodologies or who are distributed in separate offices in the same building. In both cases teams can use these techniques to improve their communication and productivity.

Why Colocate?

Teams colocate because it maximizes their ability to communicate. Working in the same room is core to all the agile methodologies, including Scrum:

> *"High-bandwidth communication is one of the core practices of Scrum… The best communication is face to face, with communications occurring through facial expression, body language, intonation, and words. When a white board is thrown in and the teams work out design as a group, the communication bandwidth absolutely sizzles."*
> - Ken Schwaber, The Enterprise and Scrum[ii]

Communication represents a significant part of the effort involved in delivering software, so lowering the barriers to communication increases a team's overall efficiency[iii]. Colocation also reduces project risk[iv]. In a recent survey of agile projects, success rates for teams located in the same room were over 20% higher than those for geographically distributed teams[v].

The Reality of Distributed Teams

Why are so many agile teams distributed? Give a choice, almost all agile teams would sit together in a single room with the customer representative or product owner to maximize communication. But in reality there are forces at play that lead to more distributed teams:

> **Global markets**. As businesses expand into new markets they need to gain expertise in those markets, perhaps through mergers and acquisitions or setting up subsidiaries located in these markets.

[1] http://msdn.microsoft.com/practices

Global talent. Increasingly companies are casting their net wider when looking to hire high quality employees. Work visa availability, relocation costs, and the willingness of new hires to relocate are all factors tend to increase team distribution.

Reducing costs. Companies often seek to reduce costs through outsourcing to regions with cheaper development overhead. An offshore service provider may represent an apparent 25% cost saving over a domestic provider[2] although making this calculation isn't trivial. While the hourly costs of individual team members may be lower, reduced productivity and additional travel costs may negate these savings.

It is important to realize that there is a tradeoff being made when the team is distributed. As we will see later the benefits of team distribution come at the cost of reduced performance and potentially greater team dysfunction. Understanding the likely impact of distribution on your team, avoiding distribution when possible, and mitigating its impact when not, is key to success.

Distributed Teams at patterns & practices

Microsoft's patterns & practices group is responsible for delivering applied engineering guidance that helps software architects, developers, and their teams take full advantage of Microsoft's platform technologies in their custom application development efforts. Our teams typically work with subject matter experts (SMEs), customer advisory boards, the larger Microsoft development community, and internal product groups within Microsoft to deliver guidance in the form of written documentation, frameworks and reference implementations.

Unlike many development teams, patterns & practices ships source code and documentation explaining the code which forms a key part of our guidance deliverables. Development projects typically last between three and six months and deliver working software to customers at regular two week intervals.

Five years ago patterns & practices adopted an agile approach to delivering guidance. Our teams use a mixture of Scrum[vi] and Extreme Programming[vii] to deliver software incrementally during the life of each project and to rapidly respond to customer feedback. Projects always have a designated team room, but frequently some of the team members are based in other locations.

Each patterns & practices team is typically made up of a product owner, developers, testers, and documentation writers with additional subject matter experts. There are several key roles on the team, each meet a specific set of responsibilities:

Product Owner. The product owner is responsible for representing the customer and helping the team understand customers' requirements. They and other team members meet frequently with the project's customer advisory board to get feedback on the deliverables from the team's last iteration. The product owner also gathers feedback from the broader community as well as key customers. They use all this information when prioritizing the product backlog and work with the team to pick user stories from it

[2] http://www.informationweek.com/news/services/showArticle.jhtml?articleID=6508358

for each iteration. We place great emphasis on having *one* product owner who represents our customer, owns the product backlog, and is always available to the team.

Coach. The coach or scrum master is primarily responsible for *how* the team executes. They facilitate retrospectives, and provide coaching around agile practices, for example test driven development. Typically this is not a full time role and falls on the senior developer on the team who is usually the person with the most agile experience.

Team Member. The team consists of developers, testers, and writers. While each discipline consists of both full time employees and contractors who report to a functional manager, the team collectively owns delivery of the project. Most teams include a senior Microsoft developer and tester who head up activities in their respective areas.

Subject Matter Experts. Subject matter experts are not usually part of the core team. Typically they act in a consultant role, attending iteration reviews or advisory board meetings to provide feedback on the team's deliverables. They may also visit the team for short or extended periods to give further feedback.

These responsibilities to the team matter far more than job titles. Effective agile teams at patterns & practices have people who meet these responsibilities regardless of their titles. How they are met varies from project to project.

At patterns & practices most teams are distributed to some degree. This allows us to build teams of highly talented people while controlling costs. Figure 1 shows a typical highly distributed team. The core team is located in Redmond, WA while the rest of the team is located offshore[3]; with additional developers in Argentina, a test team located in India and a documentation writer in the UK. Other patterns & practices teams have had distributed members in other parts of the United States, Canada, Costa Rica, and the Ukraine. We try and co-locate our teams as much as possible; this is the preferred approach but not always achievable. When our teams are distributed, they actively work to adapt their practices to compensate.

[3] From the author's perspective in Redmond, WA, offshore is considered to refer to teams outside the continental US. He readily accepts that to a collaborating team in another part of the world he may be considered to be part of an "offshore" team.

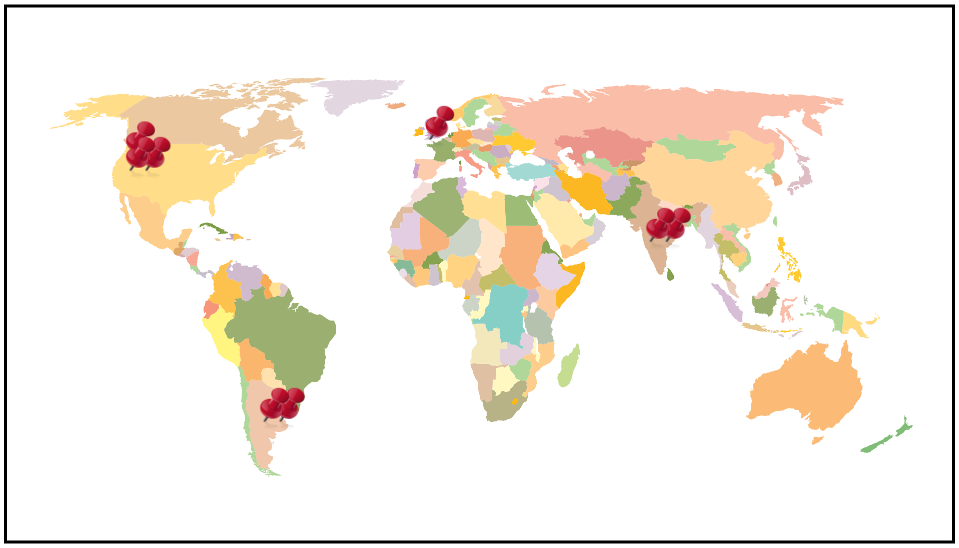


Figure 1

A typical patterns & practices team (red push-pins) may be spread across four widely separated time zones.

The patterns & practices teams are not typical of product development teams at Microsoft in general. Microsoft product teams are more likely to be in the same building[viii] and use a wide variety of approaches to software development including agile.

Challenges

While there may be business reasons for distributing a team, distribution will also contribute to team dysfunction by inhibiting communication. Agile teams rely on intensive person to person communication, both within the team and with the customer. This allows them to forego some of the additional process—like writing detailed specifications—usually associated with a more waterfall approach. Casual conversations within the team room, in hallways, and other shared spaces make a significant contribution to the team's collective understanding. Remote team members miss these and consequently their understanding suffers.

Agile approaches like Extreme Programming rely on a set of mutually supporting practices. Choosing to drop one practice will weaken the remaining ones. Therefore when the communication bandwidth is reduced teams must supplement weakened practices or replace them. Simply abandoning them — because they seem to hard—will reduce the team's level of agility.

For example, a story card[4]—a note card containing a brief summary of the functionality being developed—is really a placeholder for future conversations. These conversations between developers, testers, and the product owner will flesh out the specifics of a particular piece of functionality. These two things, the card and the conversations replace a more formal specification writing process with something that results in less ambiguous working software sooner. If the lack of rich conversations

[4] http://www.mountaingoatsoftware.com/article_view/27-advantages-of-user-stories-for-requirements

makes information on story cards insufficient or ineffective for remote team members then the team needs to supplement them with more detail.

Pair programming, where two team members sit side by side and work on the same code, is another example of a challenging practice for distributed teams. When pair programming simply isn't possible it needs to be adapted or replaced by another equivalent practice.

Teams that span time zones have further challenges when it comes to communication. There are times during each person's working day when other members of the team are not available to them. This can be mitigated somewhat by shifting core hours to better align the team members. Inevitably work will be delayed because clarification is needed or rework will need to be done because one person's best guess didn't pan out when reviewed by the team. Regular reworking in this way can also impact team health; morale suffers when people are repeatedly asked to redo work.

Reduced communication has even more impact when coaching new remote team members. Many of the practices used by agile teams, like test driven development[ix], are learned most rapidly by demonstration in a one on one coaching situation. Similarly, understanding of specific customer problem domains, system architecture and design is best learned by taking part in discussions with the team and solving problems together.

Conway's Law[5] states that any piece of software reflects the organizational structure that produced it. A distributed team will naturally tend to divide the work according the team's distribution. The end result is a product architecture that reflects the team's distribution or has artifacts that accommodate the distribution. Unfortunately your users and customers care mostly about user stories, *not* components nor architecture. They shouldn't have to care about your team's distribution and the artifacts it may introduce. Agile teams should focus on the delivering the value stated in user stories, *not* on the individual components which they manipulate and orchestrate into a solution.

Effective distributed agile development is about minimizing the impact of distribution. The emphasis is on *minimizing*. There is no substitute for two or three people having a face to face discussion at a whiteboard, or two programmers sitting together and working on the same piece of code.

Proven Practices

Creating an effective distributed agile team is largely about compensating for the barriers to communication added by distribution. In fact many distributed teams falter because they try to behave as though their team is colocated and do not effectively address the additional communication burdens placed on them. Many of the core communication challenges faced require both commitment on the part of the team to improve and the support of additional practices and tools.

The following sections outline the practices in five key areas and gives specific details of how patterns & practices teams employ many of them.

[5] http://en.wikipedia.org/wiki/Conway's_Law

Focus on Communication

Maximizing the available communication bandwidth is the key to successful distributed teamwork. There are many simple practices your team can implement to improve communication.

Minimize the overhead of setting up a meeting across locations by having a conference phone and projector easily accessible—ideally in the team room—for impromptu meetings. This allows the whole team to get together quickly regardless of their location. Teams can use Web conferencing software in conjunction with the projector to share applications across the whole team. Video conferencing, if available, is even better than voice conferencing, as participants are less likely to disengage.

Team members can use hands free headsets, Web cams, and application sharing software to allow them to work together remotely. They can also make use of instant messenger software for synchronous peer connections and e-mail for asynchronous broadcast communications. Again these tools try to minimize the differences between conversations in the room and those with remote parties. Getting people into the habit of actually using them is the key to reducing communication overhead.

These tools can also be used passively to connect to locations. For example, using a pair of Web cams to connect two geographically separated team rooms to give additional visibility as to who is actually around or having a constantly open conference line between locations.

The way teams communicate may have to change to support distributed development. In some cases, communication what would have been informal (verbal) for a colocated team needs to be replaced by more formal scheduled nonverbal communication for a distributed team. For example one team member may use e-mail or a work item tracking tool to update their status on a particular task, rather than waiting to be asked by another team member. If the team is distributed across time zones then this approach is particularly valuable (see Provide the Right Tools) because it means that the information is available even if the person is not due to differing work hours.

In other cases, communication needs to become an explicit part of someone's duties on the team. Examples of this include assigning a representative for a remote sub team or a team buddy for a single remote team member (see Team Distribution) to help them catch up on missed hallway conversations. Another practice to get the team focusing on communication is to be deliberate about involving remote team members. For example, having them lead meetings from the phone.

Meeting formats may also need to change to account for the lack of opportunity to communicate outside of those meetings. For example, your team might find that daily standup meetings become overly long as they are the only opportunity for issues to be raised for the whole team. Changing the format to include some time *after* the standup where the team can discuss other topics is beneficial.

At patterns & practices: Because communication takes longer for distributed teams, teams and coaches need to plan for this. Some tactics patterns & practices has found successful:

Each patterns & practices team room has a dedicated conference phone and projector. This is especially important at Microsoft were meeting rooms can be hard to find at short notice.

We also provide individual team members with hands free headsets and desk phones (in addition to the room's conference phone). They also use Windows Live Messenger and Office Communicator for quick voice and text communication.

Our team rooms do not have dedicated video conferencing facilities, largely due to cost constraints. Teams have experimented with using Web cams but the uptake hasn't been good and teams have not gotten into the habit of using them. This is, at least in part, due to network connectivity issues and bandwidth limitations of remote team members in South America and India.

Our developers adapted their pairing practices. They use hands free headsets, Live Meeting or SharedView[6] to pair remotely. Where pair programming isn't always possible then frequent code reviews supplement or replace pairing. No code is checked into the tree without having been reviewed by at least two people. Usually this is two developers but may involve anyone on the team who is able to review the code. Some teams also review key pieces of code collectively as part of their iteration review.

Teams at patterns & practices use Visual Studio Team System to track their work and share status across the whole team (see Provide the Right Tools). During meetings teams share a Visual Studio instance using the team room's projector combined with LiveMeeting for those who are remote.

Our teams use a parking lot system for issues that come up during the daily standup[7] and address them right after the meeting. This keeps the meeting short but allows those who need to continue to talk about related activities to do so afterwards without lengthening the standup. Team members who are not involved in these extended discussions can leave the meeting and get on with other work.

The one thing that patterns & practices teams haven't found a distributed equivalent for is the whiteboard. Taking pictures of whiteboard diagrams and adding them to a wiki allows you to share the *results* of design meetings, but currently we do not have a way to share diagrams in real time.

Plan to Travel

There's no substitute for face to face communication, particularly at pivotal points in the project. At the beginning of the project teams should insist on colocation for the first few iterations. This allows team members to get to know each other and build rapport and trust. Plan to have some social events during this period so that team members can get to know each other on a personal level. The start of a project is also when many key decisions are made, this happens more effectively if the whole team is involved in making decisions and committing to them. Plan and budget for travel as part of your project. If travel isn't possible then expect longer ramp up time for your team and plan to employ other tactics (see Team Distribution).

[6] http://www.microsoft.com/downloads/details.aspx?familyid=95AF94BA-755E-4039-9038-63005EE9D33A

[7] http://www.ademiller.com/blogs/tech/2008/07/daily-standup-meetings/

Having the whole team in the room during the early stages of the project isn't just about making key design decisions. It helps create personal bonds that will serve the team well later when they are no longer working in the same room and using toneless communication mediums—like e-mail—to coordinate their work. It mitigates miscommunication, misunderstandings and the "Us vs. Them" attitude that can all too easily infect groups working in different locations, especially in times of stress. Periodically bringing the whole team back together refreshes these personal bonds that otherwise erode over time.

Bringing the team back together for the last couple of iterations before the final release makes the process of shipping a final deliverable much smoother. It helps the team focus on "getting things out the door". Being in the same room means that they whole team is available when key decisions have to be made.

At patterns & practices: Being part of a distributed team at patterns & practices involves some travel. While it varies from team to team typical teams usually do the following:

Our teams have site visits from remote developers both at the start of the project and for one to two weeks out of every six while the project is in flight. Many teams try to incorporate some social activities for the team during these visits. Activities include both organized events—like a team dinner or game night—and opportunities for lots of informal time over lunch or coffee.

Teams also work together in Redmond at the end of the project for the last one or two iterations before final delivery. They also try to have a project retrospective and ship party before the team disperses.

Our testers either travel to Redmond for the duration of the project or work remotely from India. Remote test teams employ a team room representative approach to compensate for the large time zone difference (see Team Distribution). Our teams generally have at least two testers in the team room to compensate for the inability of the test team in India to visit Redmond. A senior Microsoft tester, a team room representative, and possibly one or more testers from India who are on site for the duration of the project.

Subject matter experts also visit the team in Redmond, for review meetings early in the project to give feedback on the teams' proposed direction. In some cases SMEs may work in the team room for extended periods. The goal here is *not* for the SMEs to do the team's work but to give the team easy access to their knowledge of the problem domain.

Team Distribution

Distribution across time zones has far more impact than geographical distribution alone. Significant time zone differences introduce communication blackouts into the team's day during which part of the team is simply not available. A time zone distribution of three or four hours is workable with the whole team sharing either morning or afternoon hours. Teams need to focus on making best use of this overlapping time, with meetings and pairing taking place during these core hours.

If your team is distributed offshore and the time zone distribution is so large that working days no longer overlap then new tactics are required to allow the different parts of the team to work together. Sub teams can each run standup meetings at the beginning of their respective days, with one sub team staying late or arriving early to accommodate this.

Another practice for dealing with large time zone differences is to use a team room representative for the offshore team. The representative should be someone who has worked with the remote team before and has their trust and a good working relationship with them. The representative is a core part of the team and attends daily standup meetings. Their role is to buffer (store and forward) communication across large time differences. For example, the representative may poll team members during a standup or at the end of their day for specific issues that need to be addressed to the offshore team. The offshore and onshore team members should still be communicating directly about specific issues, when they can.

The role of team representative can be very demanding. They need to be a part of the team, active in discussions and meetings in the team room during the day in one time zone, and also available to pass the results of these discussions on to the offshore team. The end result is a working day that might start early or end late in the evening. Over time this can lead to overwork, fatigue, and burnout.

Using a team representative has some disadvantages. Those offshore are not truly part of the team; they interact with the team through their representative (delegate). From the team's point of view the representative appears to do the work of three or four people over night. The most obvious problem with this model is that ramping up a new group of offshore people is particularly hard, as they are disconnected from the team and find it hard to learn the problem domain. In the worst case, the remote sub team may replace people without your knowledge.

Remote teams may also be *function focused*, for example a remote testing team. There are considerable risks associated with doing this. Agile tea*ms like* to use the whole team to get things done rather than focusing *on job functions* and titles within the team (cross-functional). The "developers write code and testers test" mentality leads to pushing work back and forth between silos rather than focusing on completing the user story. A functional sub team is in itself bad enough, but if that team is remote this adds to the likelihood of creating knowledge silos.

Asymmetric distributions are the hardest to deal with. In this case the majority of the team are in the team room but a single person is located off site. It is very easy for the remote team member to become entirely forgotten by everyone else. The one person joining a conference by phone will miss out on whiteboard drawing, gestures, and side conversations, sometimes without even being aware of what they are missing.

The obvious solution is to arrange conference calls to maintain parity. If one person is dialing from their (remote) office in then *everyone* should dial in from separate phones and not sit together. This enforces parity between all participants by preventing side conversations and other artifacts that are missed when only some participants are on the phone. Alternatively have people who would normally be in the room dial in for the occasional meeting to remind themselves how challenging being on the end of a phone can be.

Another option is to assign remote individuals a team room buddy. The buddy's role is to represent the remote team member to the rest of the team and make sure that they don't miss out on any key information during meetings. Team room buddies or the coach need to remind teams to focus on maintaining parity in asymmetric meetings, otherwise remote participants will feel less involved.

At patterns & practices: Teams usually use developers from South America which keeps the time zone difference down to four hours. Teams often use offshore testing teams located in Chennai, India—a twelve and a half hour difference—but make heavy use of testers in the team room and a team representative to minimize the separation of the offshore team.

Each patterns & practices team uses a team room representative for their Chennai test team. One or two other testers and the test lead are also located in Redmond. At the end of each day in Redmond the team room representative updates the Chennai team and makes sure that they are clear on what needs to be done during their day. The following morning at the daily standup meeting they bring the Redmond team up to date with the Chennai team's progress. Other teams at Microsoft have also used this approach and found it critical to their success[x].

At patterns & practices transferring domain knowledge to the remote test team is primarily the responsibility of the team room representative. This may be augmented by joint whole team meetings especially at the start of the project. SharePoint wikis have also proven very useful for sharing information and design decisions made by the team (see Provide the Right Tools).

Several patterns & practices teams have also focused on having developers and testers work together pairing on creating acceptance tests for user stories. This has been a very successful practice for sharing knowledge around the team.

We try and avoid highly asymmetric distributions as single remote team members inevitably get left out. Therefore we don't tend to have need for the team buddy approach.

Focus on Coaching the Team

Distributed teams encounter more challenges and need more help to enable them to stick with the core practices employed by agile teams. Many distributed teams abandon key practices because they seem too hard. Having one person on the team who is committed to the coach role is vital in keeping a distributed team on the right path.

Many of the core Extreme Programming development practices serve to reinforce each other so it is important not to simply discard a practice, rather it should be modified or replaced with something equivalent. For example; pair programming might be augmented or replaced by code reviews and story cards may need fuller descriptions to convey details usually gained from conversations.

It's important to realize that the role of the coach on *any* agile team is not as the practice cop—policing a set of practices come what may—rather they should be helping the team deliver software in an agile manner. The coach's role is to remind the team of the underlying principles and help guide them to adapt their practices when the temptation might be to abandon them. A coach should help the team remember the value of intensive communication even when distribution makes this harder.

This means the team can still be agile; *delivering working software in a rapidly changing environment and continuously delivering value to customers in a sustainable way*. This is even more significant in a distributed team where some practices may have to be extensively modified. The coach can help the team to keep the values and principles of agile in mind while it figures out the practices that will best allow them to keep their promise to themselves and the customer. Over time you can still build an effective distributed team.

At patterns & practices: We've recently refocused on the coaching responsibility within teams. This was largely motivated by feedback from project retrospectives and reviewing success factors on individual projects.

We provide sufficient documentation so that new team members have a reasonable idea how a typical project runs and what is likely to be expected from them. This consists of a wiki which builds on the following books:

Scrum and XP from the Trenches, Henrik Kinberg[xi]

Agile Project Management with Scrum, Ken Schwaber[vi]

Extreme Programming Explained, Kent Beck and Cynthia Andres[vii]

This documentation describes a baseline approach that teams are encouraged to adapt to their needs.

Our teams and coaches focus on what works for the team, not checking off a series of practices on a list.

Our teams routinely experiment with different practices and tools to see what works best for them. A recent example of this was the adoption of the Conchango Scrum for Team System template for Visual Studio Team System (see Provide the Right Tools).

We hold iteration retrospectives and end of project retrospectives. If possible these are planned to coincide with visits by offsite team members. When this is not possible, teams run retrospectives using a conference phone and Live Meeting. Live Meeting's shared text page feature is used as a substitute for sticky notes during brainstorming exercises. Anyone can type their feedback or suggestions into the shared text page. The retrospective facilitator is responsible for making sure that *all* suggestions receive equal consideration.

Distribution of Work

One of the challenges when the team distributes work is *not* to do so according to location. Doing this will have negative effects over time. The architecture will begin to reflect the team's geographical distribution (according to Conway's Law). Different locations will become over specialized in particular components. Ultimately both these effects will make user stories harder to complete within an iteration as different parts of the team have to complete specific pieces of work on a critical path to complete a given cross-component user story. This component based approach and its shortcomings are examined more fully by Larman and Voode[xii] and have even more significant downsides on large scale agile projects.

Distributed teams should continue to think about their work in the context of completing user stories *not* adding features to components. They need to consciously distribute tasks relating to a single story across the whole team, regardless of geography, and think in terms of user stories *not* system components. This is one of the most challenging areas for a distributed team as it requires the team to work closely across geographical boundaries but will to lead to consistent user experiences and reduce gaps in functionality that fall between components.

Many of the other practices discussed in this paper will help improve communication and mitigate the problem. Over specialization by geographical location and/or component is a sign that your team is succumbing to some of the communication challenges caused by distribution by changing the product to suit their development, *not the customer's requirements*.

At patterns & practices: At patterns & practices we are very conscious of the above pitfalls since almost all of our deliverables include our source code. We want the source to reflect the best guidance on the technology, not the locations of individuals on the team. Achieving this is difficult to manage as teams have a natural tendency to break down stories into features and then create and assign tasks based around components for that feature..

Our teams use on site visits by remote developers to transfer knowledge using pairing and design sessions. This practice breaks down knowledge silos that start to build up over time when working is distributed.

We avoid breaking user stories into tasks and then assigning tasks according to geography and/or skill sets. Over time this would build up knowledge silos leaving you with new work that can only be done by one or two people.

We consciously work to pair and review code regardless of geographical location.

Build the Team over Time

Good teams take time to form, and work best if they stay together on multiple projects over long periods. One of the challenges of distributed teams is that this formation process is artificially extended by reduced face to face communication. New team members take longer to ramp up on the team's practices as well as the customer problem domain. It also takes longer for team members to get to know each other on the personal level that is required for really effective work. Therefore, reorganizing distributed teams is even more disruptive than with a colocated team.

Agile teams often do not produce a lot of formal artifacts, like specifications and design documents, which new team members could use to learn the relevant information from. Special attention must be given to new team members who are working remotely when they join the project. Consider assigning them a buddy or mentor to help them get up to speed. Distributed teams may produce additional artifacts like wiki pages (see Provide the Right Tools) to document designs and processes solely for the purposes of helping new team members get up to speed.

Building teams from a set of core members who will be on a project for several releases will help. Adding other team members who are there for shorter periods will then be less disruptive as they can learn from the team's core members. Over time the makeup of the core team may also slowly change. Avoid starting teams from scratch for each project with no continuity guaranteed between projects.

At patterns & practices: We realized the significant impact of churn on agile teams, and how much better long lived teams tended to perform, so we have tried to keep teams together over multiple projects.

Our projects tend to be aligned around programs of work with several projects delivering to the same program over a period of one or two years.

Our teams have core members may stay on the same project for two or more releases to ensure team continuity. We also try to get the same contract staff and consultants to return for subsequent projects to further maintain team continuity.

We try and keep at least half of the team from one release to the next. This gives the team a core on which to build.

Our teams use SharePoint wikis to store key project information like designs and build processes and metrics which are helpful to new team members joining the team and when starting up a new release.

Provide the Right Tools

While tools are not the primary focus of agile teams, or this paper, they can make agile teams more effective provided they are adaptable and do not disrupt the natural flow of work.

> *There is reason and sense behind valuing individuals and interactions over processes and tools. Processes and tools distill common experience. Along comes an uncommon experience and processes and tools can't adapt. Only humans can. That's where the individuals and interactions come in—figuring out how to make progress in unusual situations. A blind adherence to processes and tools in the face of novelty is as ineffective as persisting in variety, debate, negotiation, and creativity in the face of routine and repeating circumstances.*
> Kent Beck— Tools for Agility[xiii]

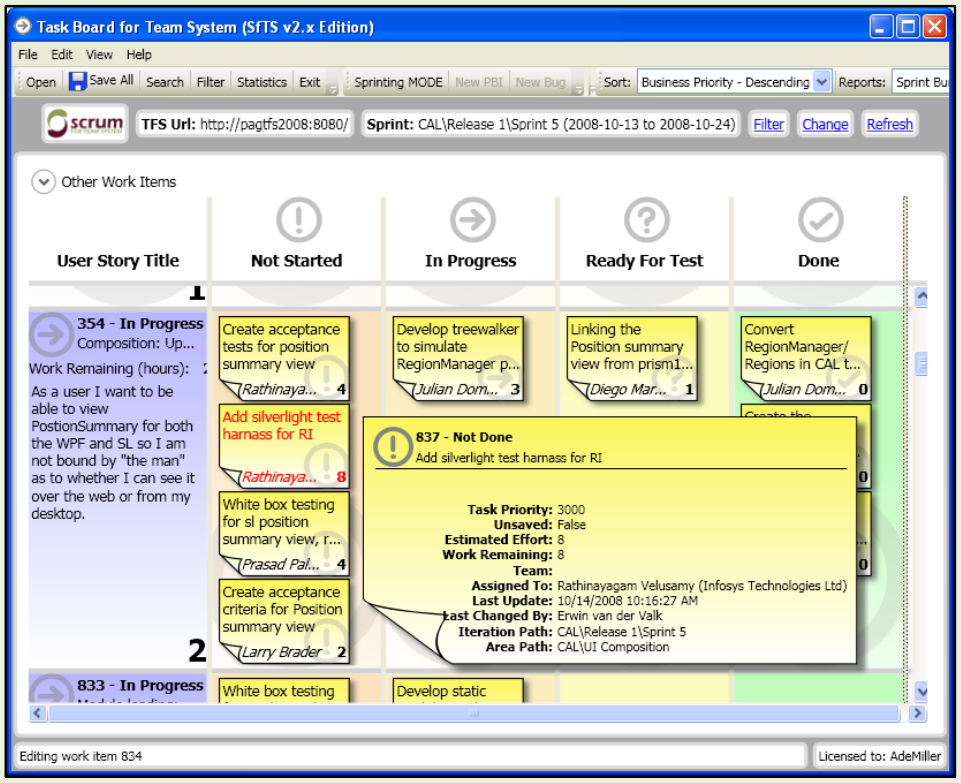
Agile teams cannot rely on sticky notes on a task board or a burndown chart on the wall for project tracking if they are not in the same room. Similarly designs and diagrams that might have lived in the team room need to be shared across multiple locations. In addition to the communications tools already discussed (see Focus on Communication) some sort of tooling is required in order to share information amongst the team.

Depending on your development environment there are numerous products which allow distributed teams to publish status and coordinate their work. Teams will probably need to experiment over several iterations to find out what works for them. The decision to distribute a team needs to be coupled with the commitment to provide the team with the tools it needs to maximize communication and the expectation that it will take them some time to optimize around them.

At patterns & practices: There are a number of specific tools available that patterns & practices teams have found particularly useful for tracking progress and working in a distributed environment.

Visual Studio Team System. Sticky notes stuck to a wall lose their effectiveness in a distributed team. Team System[8] helps the whole team see the current state of the iteration backlog and update individual tasks. It is also possible to use a combination of Excel and Word's mail merge feature to create task cards for use in a team room directly from Team System work items[9].

Scrum for Team System. Conchango's Scrum for Team System[10] is a free add-in for Team System that implements a Scrum task board and burndown charts to help with iteration planning and tracking. Tools like Scrum for Team System and its task board try to replicate the experience in the team room for distributed members of the team.


Task Board for Team System (SfTS v2.x Edition)
File Edit View Help
Open Save All Search Filter Statistics Exit Sprinting MODE New PBI New Bug Sort: Business Priority - Descending Reports: Sprint Bu
Filter Change Refresh
Other Work Items
User Story Title
Not Started
In Progress
Ready For Test
Done

TFS Url: http://pagtfs2008:8080/ Sprint: CAL\Release 1\Sprint 5 (2008-10-13 to 2008-10-24)

354 - In Progress
Composition: Up...
Work Remaining (hours):
As a user I want to be able to view PostionSummary for both the WPF and SL so I am not bound by "the man" as to whether I can see it over the web or from my desktop.

Create acceptance tests for position summary view — Rathinaya... 4
Add silverlight test harnass for RI — Rathinaya... 8
White box testing for sl position summary view, r... — Prasad Pal... 4
Create acceptance criteria for Position summary view — Larry Brader 2
Develop treewalker to simulate RegionManager p... — Julian Dom... 3
Linking the Position summary view from prism1... — Diego Mar... 1
Convert RegionManager/ Regions in CAL t... — Julian Dom... 0

837 - Not Done
Add silverlight test harnass for RI

Task Priority: 3000
Unsaved: False
Estimated Effort: 8
Work Remaining: 8
Team:
Assigned To: Rathinayagam Velusamy (Infosys Technologies Ltd)
Last Update: 10/14/2008 10:16:27 AM
Last Changed By: Erwin van der Valk
Iteration Path: CAL\Release 1\Sprint 5
Area Path: CAL\UI Composition

833 - In Progress
White box testing
Develop static

Editing work item 834
Licensed to: AdeMiller

Figure 2

Scrum of Team System's task board allows distributed teams to work together.

[8] http://msdn.microsoft.com/vsts

[9] http://www.ademiller.com/blogs/tech/2008/09/team-system-and-scrum

[10] http://www.scrumforteamsystem.com/

SharePoint. Collaboration tools like SharePoint are also valuable for sharing information and recording decisions made by the team. SharePoint wikis are especially good for quickly creating a historical record of design decisions made during the project. The important information is often *why* a particular approach was used rather than the final design. Our teams use cameras to capture and upload pictures of whiteboards into the wiki.

Live Meeting. Conferencing tools that support real time information sharing are also really valuable. Teams use Live Meeting to show code, notes, or a document to the entire team for discussion. Team code reviews, paired programming, design discussions, and even shared prototyping can all be conducted over Live Meeting. Teams at patterns & practices have also used Live Meeting's shared text page feature (which allows multiple people to type in the same text document at the same time) to conduct brainstorming sessions, story estimation sessions, and team retrospectives.

A typical iteration planning meeting for a patterns & practices team would make use of many of the tools described here. The Redmond team gets together in the team room and everyone dials in to a conference call. In the team room they use the projector to display their task board so they can collectively review it. They also share the board using Visual Studio so remote team members can see updates to it in real time.

Conclusions

With distributed agile development it is possible to tap into new global markets and make best use of globally available talent, while potentially reducing costs. Teams at patterns & practices have been successfully using this approach for a number of years but its success should not be taken as a given. The decision to distribute your project should be a conscious one and the decision maker(s) must understand that in doing so they reduce the project's likelihood of success, increase the delivery time, and reduce the team's performance and increase its dysfunction. The risk/reward tradeoff needs to be clearly understood before deciding to distribute your team(s).

Distributed agile development requires significant effort on the part of the team and support from management in order to be truly successful. One of the key success factors in an agile project is the high level of communication made possible by having the team sit together. If your team is distributed they need to make deliberate efforts to replace as much of this lost communication bandwidth as possible and adapt and augment their practices to account for this loss.

Bring the whole team together at the start, end, and other pivotal points during the project. Use these periods of working in the same place to not only build shared understanding of the problem domain but also working relationships within the team. Expect to have people travel a couple of weeks every other month if you want to maintain the team's cohesion.

Agile teams work on user stories not component features or discipline based tasks. Organizing your team's distribution by component or discipline will make it much harder to focus on what matters to users, their stories. Where these distributions are unavoidable make a conscious effort to mitigate them in the short term and ultimately change them in the longer term.

Agile development is hard and requires a great deal of discipline. Distributed development is harder still and requires yet more resolve to stay on track. Make sure that your team has someone with a clear mandate to coach them. Establishing a baseline for how your teams should function will help guide them and the coach, but they should be encouraged to treat it as guidance and adapt it to suit their needs.

Having made all this investment to build a team don't throw it away at the end of each project and start from scratch. Even teams working in the same room take a long time to form. When distributed this formation takes even longer. Plan to keep teams together or change them slowly over time to maximize your investment in them.

Provide distributed teams with the right tools to work as effectively as possible and remove as many of the barriers created by distribution as possible. Expect to experiment with how you use these tools, be it conference phones, collaboration software, or work item tracking tools.

The patterns & practices group has been successful with our distributed approach largely because our teams have done many, but not all, of the things described here. Understanding what makes your agile teams tick and how that might change when the team is distributed is the key to knowing how to get the best results from them.

Dos and Don'ts: What are the most important aspects of distributed development that patterns & practices teams focus on when running distributed teams? Key things that have, or have not, worked for patterns & practices teams:

Do work to maximize the available communication bandwidth available to your team. Provide communications tools—like conference phone, Web cams, and hands free headsets—for your team and help them adapt their existing practices to distribution.

Don't continually reorganize your teams for each new project. Building teams takes time, building distributed teams takes even longer. Maximize your investments in team building by minimizing churn on teams.

Do plan to travel, especially at the project's pivotal points. Bring everyone together for the first couple of iterations, periodically during the project, and right before final release.

Don't distribute the work by system components, focus on user stories. Avoid organizing distributed teams by function—for example, the offshore test team. Both these approaches create knowledge silos within the team.

Do provide tools to augment or replace those that only work within a team room—like a work item tracking system to replace sticky notes on whiteboards.

Don't let remote team members be forgotten in team meetings. Pair them up with a buddy and try putting everyone on the same footing by having all members call into conference calls at least occasionally.

Do evolve the team's practices as they identify better ways to deal with the challenges of geographic dispersion. Frequent retrospectives are the key to getting a team to consider how to improve.

Don't forget to include everyone in frequent team retrospectives to identify what does and does not work for the team.

Do focus on coaching. Make sure everyone understands why agile practices need to be adapted for distributed development.

Acknowledgements

The author would like to thank the following for their feedback on drafts of this paper; Harry Chen, Brooke Hamilton, and Jezz Santos (Microsoft Consulting Services); Larry Brader, RoAnn Corbisier, Steve Elston, Alex Homer, Ajoy Krishnamoorthy, Michael Puleio, Don Smith, and Erwin van der Valk (Microsoft patterns & practices); and Alan Wills (Visual Studio Team System).

About the Author

Ade Miller is the Development Manager with the patterns & practices group where he manages several agile teams executing on a variety of projects. He spends much of his time thinking about what being "agile" really means and writes about this frequently on his blog http://www.ademiller.com/tech/.

---

[i] 3rd Annual Survey: 2008 "The State of Agile Development" http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf
[ii] Ken Schwaber, The Enterprise and Scrum, Microsoft Press, ISBN 073561993X
[iii] Teasley, Covi, Krishnan &Olson, How Does Radical Collocation Help a Team Succeed? http://possibility.com/Misc/p339-teasley.pdf
[iv] David Norton, Making the Most of 'Agile' in a Distributed Development Environment, Gartner Research G00159246, August 2008
[v] Scott Ambler, Has Agile Peaked?, Dr. Dobbs Journal, June 2008
[vi] Ken Schwaber, Agile Project Management with Scrum, Microsoft Press, ISBN 073561993X
[vii] Kent Beck & Cynthia Andres, Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley Professional, 0321278658
[viii] Begel & Nagappan, Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study http://research.microsoft.com/projects/hip/papers/AgileDevAtMS-ESEM07.pdf
[ix] Kent Beck, Test Driven Development by Example, Addison-Wesley Professional, ISBN 0321146530
[x] Joy Shafer, Distributed Agile: An Experience Report, Proceedings of the Pacific Northwest Software Quality Conference, October 13-15th 2008, Portland OR, in press
[xi] Henrik Kniberg. Scrum and XP from the Trenches: How we do Scrum, C4Media Inc., 2007.
[xii] Craig Larman & Bas Vodde, Scaling Lean & Agile Development: Successful Large, Multisite & Offshore Products with Large-Scale Scrum, Addison-Wesley Professional, ISBN 0321480961
[xiii] Kent Beck, Tools for Agility, http://www.microsoft.com/downloads/details.aspx?familyid=AE7E07E8-0872-47C4-B1E7-2C1DE7FACF96