

Agile Development and Remote Teams: Learning to Love the Phone

Christian Sepulveda
at design time, inc.
csepulv@atdesigntime.com

Abstract

I currently work on a project where we adopted an agile process that integrates elements of extreme programming and agile modeling. Our approach is unconventional however; instead of the team being co-located, I work remotely as the lead developer.

The risk of increased communication costs can be mitigated rather easily. However, trust is the most complicated element of team dynamics to establish and maintain. A virtual team must address this issue in order to succeed.

Remote teams can work quite well. We have been delivering quality software in a timely manner, within the expectations of management for the last two years. We actually are more efficient and successful with a virtual team than when we were all co-located in the same room.

1 Introduction

“Hey Mike.”

“Yeah.”

“You got a minute to take a look at something with me.”

“Yeah, sure.”

[Two programmers get themselves set up to work as a pair.]

[After a few minutes of working together has passed.]

“You know, I think Peter wrote something similar to this a week ago. We could probably refactor it and use it here.”

“Hey Peter...”

This is a typical dialogue for me. It is probably a typical dialogue for any agile team that employs pair programming. There is an unconventional difference in my situation; certain details were omitted in the stage direction. When the two developers were setting up to work, they were in two different locations, communicating over telephone headsets while utilizing Placeware® online screen sharing software. We develop software with a remote team and agile methods.

2 Background

Three years ago I started working with my current client. At that time, they sold a single application, developed by one of the founders (their president). They wanted to develop new applications, though they had no experience assembling and managing a development team. I was commissioned to build the team, coach the developers and lead product development.

As the development team was assembled, there was a lot of debate over process election. The client wanted a large degree of formality. The president (original developer) would not be writing any code and therefore wanted various reporting and supervisory mechanisms to track progress and specify requirements.

I was in the middle of my own professional transition; I was a partner in a small consulting firm that primarily did implementations. We had always used ad hoc practices that, by today’s standards, would be classified as agile development: pair programming, “just-enough” documentation, close collaboration with users and domain experts, and automated testing. But I always had a little bit of concern that our approach was too informal. I was curious if something like the Rational Unified Process was effective.

In addition to supervisory mechanisms, my client wanted significant documentation, as they were worried about knowledge loss resulting from staff loss. We decided to use the Rational Unified Process because of its incorporation of iterations, documentation and structured workflows.

After a year, we were not much further along than when we started. We had degenerated into a waterfall approach. I spent most of my day in meetings: product positioning, architecture, requirements capture, requirements analysis, etc. This left little time for me to actually write code and left the other developers in a holding pattern, of sorts. We had only been writing production code (though we did a lot of prototyping) for three months.

To make matters worse, I was moving to a new city (for personal reasons). We were faced with a difficult

decision: replace me or I would remotely manage the team and write code. Replacement meant recruiting, training, and acclimating a new team leader; remote management was unknown.

We pursued both options initially. We talked to staffing agencies and described our requirements. I called colleagues and associates, looking for my replacement.

The staffing firms were of no help in finding candidates, but the process of communicating my client's needs made us appreciate how enmeshed I was in the project. The client was comfortable with me. I had learned much of the domain logic and the needs of marketing and sales. I had developed relationships with the team. We decided remote development was the better option.

3 Time to Reevaluate

We needed a plan for the transition to remote management. Remote development would have its costs and risks, so we would have to find ways to compensate. We began to reflect on the past year. We had little tangible evidence of progress other than diagrams, requirement documents, models and memos. We realized that all the paper wasn't worth much.

Mentoring developers and collaboration would be difficult. I already had very little time to work directly with the team; how would I ever be an effective mentor remotely if I wasn't doing a good job onsite?

Everyone was getting fed up with our situation. My moving and our choice of remote development was the excuse to start over. We asked one question and used it to guide all of our decisions: What is the bare minimum process we can get away with?

Management wanted to me to regularly conduct code reviews. Seeking to minimize overhead, I suggested pair programming. I presented it to the client as a way "to kill several birds with one stone"; I could mentor developers, ensure common design and code practices and "supervise" the team.

My client agreed that pair programming between myself and each of the developers would be a good idea. The client already had a site license for a product called Placeware®, an online meeting application. It works in a web browser and has a screen-sharing feature. We planned to use this feature to facilitate pair programming.

4 And so it Begins...

We started our remote development with two workflows: limited pair programming between an onsite developer and me and small batches of requirements that we would assemble every few days via informal requirement memos. (Basically, these are XP cards, but

they are typed.) These memos were blurbs we would record about the feature, as we discussed it over the phone in small groups of two or three people.

It was irritating to hold a phone next to your ear all day. Thankfully, one of the developers realized he could "borrow" a telephone headset from one of the telemarketers and this was much more comfortable. It wasn't long before we all had them.

No one, with the exception of myself, had ever done any form of pair programming before and they had mixed feelings. Some of the developers felt it was a form of micromanagement and supervision.

The first few sessions helped alleviate some of the apprehensions as the developers learned a few "tricks" from me. More importantly, I learned a few "tricks" from each of them. This dynamic exchange made the experience enjoyable and each of us felt the productivity and quality gains.

Management and domain experts had to adjust to the lack of paper documents. These were replaced by a series of phone calls, which were eventually augmented by online (via screen sharing) modeling sessions. These sessions were brief; I would continue until I had enough information that I could start coding.

5 Signs of Progress

We reached two significant milestones after the first and third month of remote development. After the first month, we had an actual executable that the client could tinker with. After three months, we had achieved significant mass and the application had taken its true shape; the primary features had been implemented. We had accomplished more in three months than in the previous year.

6 Tweaks to the Process

6.1 Pair Programming spreads....

The entire team started to work in pairs in ad hoc fashion. As they grew accustomed to the benefits of pair programming with me, they frequently would "grab another pair of eyes" to take a look at some code. They eventually started to work together in pairs, switching partners and working without the need for management supervision.

6.2 Domain Experts Work In Pairs

Our domain experts started to debug code with developers. This made communication much faster, as a defect needn't be logged. Its reproduction and resolution could be done more efficiently. Domain experts will, at

times, work directly with a developer while coding, further lowering communication costs, while increasing collaboration, the quality of team dynamics and trust between users and developers.

7 Here Comes the Pain Again

There were a variety of growing pains. Remote development changes the mechanism of communication from a tap on the shoulder and face-to-face conversation to a telephone call or “instant message”. Group meetings with a speakerphone in a conference room are a little awkward, regardless of the quality of the conference telephone hardware.

Some people have a hard time adjusting and can’t separate the method of communication (transfer mechanism) from the content and quality of interaction. Tools such as the headsets make the switch more comfortable, but it does require people to change their habits; some have an easier time than others.

7.1 Trust

I find that trust is the most difficult component to establish in remote development.

Trust relationships are somewhat strained, at least for me. I am “out of the loop” at times; I do not participate in morning chitchat. I have to make extra efforts to maintain the quality of interactions and be aware of how I say things, as my body language is not communicated via a telephone call.

I am onsite about twice a month. We try to schedule team lunches and dinners, particularly in conjunction with milestones and project accomplishments. These interactions make the remote communication easier. The person on the other end of the telephone becomes more human; his body language and expressions can be envisioned. He is more than just a voice.

One particular incident taught me how truly complicated building trust is in a remote setting. I became aware of animosity that existed between a few team members and me. The worst part was that I was completely ignorant of the tension for some time.

The problem originated from a confrontation between one individual and me. I thought the situation was resolved, but the individual felt the relationship as strained. He would make casual, derogatory comments about me with some of the other developers; it was office gossip that started to fester. His negative perception started to spread somewhat. A small clique developed among a subset of the teams, whose original bond was dissatisfaction with me.

These developers were able to separate the professional from the personal and we were able to work

fine together, or at least so it seemed. Since we were delivering timely and reliable software, I assumed the team was working well. I wasn’t sensitive enough to the situation to realize there was a problem.

One day a comment a developer made struck me as inappropriate. Upon some prodding, he told me some of the developers did not get along with me. We had a group meeting to expose grievances. The group disclosed their negative experiences with me and that they had discussed them with each other. Some admitted that in hindsight, their shared complaining fueled their animosity and they should have confronted me about the problem. Others felt justified in their feelings and insisted that I was unapproachable; they had “written me off”. This was the reason they could work with me, as they had no expectations or emotional investment in me.

There has been much discussion amongst the team about the whole incident and those incidents leading to our “pow-wow”. I have since made extra efforts to work closer with all team members. For those who felt they should have addressed the problem earlier, our relationships have greatly improved. I sense it hasn’t improved much with the others who had dismissed me. However, they continue to work in pairs with myself and interact with the whole team. Productivity has not changed in any perceptible way throughout the whole incident.

Upon reflection, I think some of the complications are directly related to the fact I am a remote member. For example, one of the individuals was hired after I had moved to a remote setting. With the exception of onsite visits, our entire relationship developed through email and telephone calls. Neither one of us knew the “shorthand” and idiosyncrasies of our communication and collaboration styles.

Other factors are simply personality differences. I am very close to some developers, even though our entire relationship has developed remotely. These developers are much closer to me than they are to their co-located colleagues. In any situation, some people will become closer than others. Unfortunately in the remote setting, natural differences are magnified because of the lack of face-to-face interaction.

I have made some adjustments. Scrum-style morning status meetings, conducted via a conference call, allow me to engage in small talk with everyone as people join the call. This prevents me from calling each member individually, which even if I did only for trust building, would appear contrived and as a form of micromanagement. I also use the telephone instead of email wherever possible. The telephone allows tone and character to be conveyed where email is cold and lifeless. Finally, I use “spies”. I tend to engage in small talk with those closest to me and they will volunteer a variety of

information. This “gossip” contains valuable clues regarding current office climate and events that affect the teams mood and environment.

8 State of the Union

Development is doing quite well, particularly from the point of view of management. We reliably produce on time. It empowers the decision making of other business units. It also gives development a degree of latitude; management doesn’t care about the name of the process we are using, or why we are working as pairs. Development just works.

There has been our share of complications. Some have already been discussed, such as the tensions between some team members and myself. Others were the transitional complications of adjusting to remote development.

One of the most significant achievements is that no one worries about the remote situation; it is not an impediment for us. We all keep the same hours and are available to each other as if all co-located. At times someone must repeat herself because the speakerphone wasn’t too clear, for example, but these elements don’t impede development. Overall team dynamics, development schedules, and requirement negotiations are unaffected by me being remote.

I am the only one who works remote exclusively, but other team members will work remote on occasion. For example, one of our developers had to go overseas for some time to resolve a visa issue and was able to work remotely without incident. There are other instances where others work remotely without any problems. We have become adept at remote communication and collaboration.

9 Guidelines for Remote Teams

9.1 Identify the Reason for a Remote Team

In my current project, personal reasons required me to move. I was a critical resource for the project and we had to choose from two alternatives: the overhead of replacing me or making remote management and development work. We decided that a remote team was less costly and less risky, but we considered both options carefully. Electing to use virtual teams should be more of a necessity than simply an idea.

9.2 Mitigate the Communication Cost

Outfit each team member with a telephone headset; phone conversations will be easier. Instant messaging

software can be useful, particularly for developers as it is accessible right from their workstations.

Invest in remote screen sharing software. Microsoft NetMeeting®, Placeware® (www.placeware.com) and WebEx® (www.webex.com) are such products. Placeware® and WebEx® work in a browser, so there aren’t firewall issues, but Microsoft NetMeeting® is free. A drawing tablet (Wacom and Aiptek) is useful for diagrams. Video projectors combined with screen sharing facilitate group meetings.

9.3 Building Trust

Pay extra attention to phone or computer-based interaction. This requires reflection and assessment of interactions, always considering how they can be improved. Strive to build a relationship as if co-located. If team members like to engage in small talk, then engage in small talk. In general, the same principals of building trust that apply to any context apply to the remote context. The significant difference between remote settings and co-located settings is that the monitoring and feedback mechanisms (body language and observation of office climate, for example) are not available. Alternate monitoring and feedback mechanisms need to be devised.

Arrange group activities and onsite time. This is logistically difficult if the team has never worked together and the majority of members are remote.

9.4 Assess the Cultural Context

Each organization has a certain culture, which is influenced by both management and the team members. Certain contexts are more accommodating of remote development; others are not. For example, some people have a hard time adjusting to alternate primary communication mechanisms, such as the telephone instead of face-to-face conversation.

Besides the communication mechanism, the culture needs to have trusting management. Micromanagement and excessive monitoring are very difficult in a remote setting. Though these are never good ingredients for any development, certain organizations will not tolerate the lack of “control” in remote development.

There are two different scenarios to consider: an agile team that will pursue remote development and a non-agile development that will pursue remote agile development.

For an existing agile team, there are a few factors that are significant for remote development. If the culture of the organization already accommodates agile development and it is truly being practiced, remote development will require an adjustment, but the practical nature of agile teams should be sufficient. If the culture has little support for agile development, the remote setting

can be too much to bear. For example, if pair programming occurs infrequently and is mildly productive, the adjustment to online screen sharing, telephones, and arranging a session will make remote pair programming unproductive.

For a non-agile team, culture has a different significance. Though it may be easier to “sell” agile development to management (please see the section “10.3 Agile Development is Easier Sell to Management” for this discussion), the cultural mores of the organization need careful consideration. Why is agile development not being used currently? Is there management opposition? Are the developers not comfortable with pair programming? If agile development is inappropriate for the context, it may exacerbate the complications in a remote configuration.

9.5 Simulate the Co-Located Environment

Identify the conditions of the co-located environment and structure the remote arrangement to simulate it.

For example, if the developers would have worked together in the same room, remote members should work the same hours and be available as if in same room; phone tag should not happen.

Morning status meetings are common in methods such as Scrum. These meetings should occur normally, with remote participants using speakerphones, conference calls or video conferencing. Be careful not to change the normal workflows and preferably make small adjustments to accommodate remote members. In my experience, there has not been any workflows or arrangements that could not be accommodated remotely.

10 Lessons Learned

10.1 Remote Needs a Strong Justification

There must be a need. It shouldn't be an experiment or something attempted as you tinker with a process.

Generally, the reason will be based on member retention or recruiting. For example, those with necessary expertise may not be willing to re-locate. Some may have to re-locate, as was my situation. Others may want to work from home for personal reasons, to raise children for example. In all these situations, the risk of not having a certain team available is worth the complications using a remote team.

10.2 Mechanism of Communication and Collaboration

Location affects the mechanism of communication and collaboration. When onsite, the mechanisms are

direct conversation and working side by side; a remote situation requires telephones, headsets and screen-sharing software. Though these alternative methods may seem awkward at first, you can quickly adjust. The quality of interaction does not need to suffer, assuming there is adequate trust and open-mindedness.

Alistair Cockburn writes of the cost of communication in *Agile Software Development*. He provides an example of the varying costs of getting a question answered. He maintains that greater latency for getting a question answered results in greater cost to the project. This cost compounds with the number of questions that arise per team member, per day, over a project's duration. [1]

Remote development does not have to increase the latency of responses; it simply changes the mechanism of the exchange. As discussed in section 9.5, if in the co-located environment developers and domain experts sit next to each other, the remote configuration should provide for the same immediate feedback. Instant messaging software and speed dial quickly put people in contact, assuming all parties make themselves available.

Several of the domain experts I work with have multiple responsibilities besides contributing to development. At times, they are unavailable and hinder development. This is a detractor from our efficiency, but it is independent of the any remote configuration as these domain experts are located with the developers on the same site.

Time zone differences can complicate matters. In my situation, we have dealt with bi-coastal differences and overseas differences. The three-hour time difference is not hard to accommodate; some start earlier, others work a little later. In an overseas situation, it is not so simple. The single individual who was overseas did shift his work time to provide some overlap with the remainder of the team. For the incompatible time, communication latency was increased as asynchronous communication resulted. I do not know the feasibility of maintaining a protracted effort with distinct time zones and work periods.

10.3 Agile Development is Easier Sell to Management

It is easier to “sell” agile development to management when positioned as a risk mitigation strategy and not some technology trick or fad the developers may wish to try. Assuming the need for a virtual team has already been established, the potential complications of remote development will be a concern for the stakeholders.

Agile development is partially motivated by pragmatism and it this perspective that appeal to management and stakeholders. Agile methods provide

for immediate feedback with techniques such as pair programming and short iterations. The efficiency offered by agile development can be presented as a mitigation strategy to offset the potential communication and collaboration costs of remote development. In my current situation, I presented agile techniques without disclosing the “agile” moniker. I simply addressed the obstacles of remote development and the solutions provided by agile development.

11 Overall

Software development depends on mitigating risk and lowering cost. If there are not pressures to consider a remote team, then I advise against “going virtual.” Why introduce risk?

Remote teams can have some non-obvious benefits. During the time I was onsite in my current project, we spent too much time in meetings, modeling and gathering requirements. I was repeatedly called upon to put out “fires”. I had little time to write code, one of my primary responsibilities. Being remote, I am not in a position to get distracted by such tangential issues.

We had concerns about the costs and troubles of a remote team. We evaluated our workflows and knew we

needed to be more efficient. For example, we eliminated most meetings, as it is harder to organize a meeting with remote members; you can’t simply walk around the office and gather people.

The transition to a remote team forced us to prioritize and focus on developing software. We employed agile methods, as we couldn’t afford much ceremony or formality. We basically cut the “fat” from our process.

We have been successful delivering quality software in timely and reliable manner for about a year. We are actually more successful in the remote team arrangement than the co-located one.

This doesn’t in anyway imply that remote teams make it easier to succeed, but they don’t have to threaten success either. A project’s success is dictated by the quality of its team dynamics, experience of its members and a commitment to producing quality software, not its address.

12 References

[1] Cockburn, Alistair. *Agile Software Development*, Addison-Wesley, Boston, 2002. p. 77.