

# THE DISTRIBUTED SCRUM PRIMER

Version 1.0

Pete Deemer  
*Author*

Narinder Kumar Vikas Hazrati  
Gabrielle Benefield Robert Benefield  
*Contributors*

## Important Note

A thorough understanding of the principles and practices of Scrum is recommended prior reading this guide. We recommend *The Scrum Primer*, available for free at [www.scrumprimer.com](http://www.scrumprimer.com), and *The Scrum Guide*, available for free at [www.scrum.org/scrumguides](http://www.scrum.org/scrumguides).

## About the Author

**Pete Deemer** trains and coaches companies doing distributed agile development in India and elsewhere in Asia. Pete is the founder of GoodAgile ([www.goodagile.com](http://www.goodagile.com)) and co-founder of The Scrum Training Institute ([www.scrumti.com](http://www.scrumti.com)). He is a Scrum Alliance Certified Scrum Trainer, and the co-author of *The Scrum Primer*, a widely read introduction to Scrum. Pete has spent the last 22 years leading teams building products and services at global companies, and as Yahoo!'s VP of Product Development he led the company's large-scale adoption of Scrum. In addition to his corporate work, Pete is a visiting lecturer at the National University of Singapore's Institute of Systems Science.

## About the Contributors

**Narinder Kumar** and **Vikas Hazrati** are the Delhi-based co-founders of Inphina Technologies ([www.inphina.com](http://www.inphina.com)), which provides agile software development services to clients in Europe and the US.

**Gabrielle Benefield** is a Scrum Alliance Certified Scrum Trainer, and she is the founder of London-based Agile Lean Training ([www.agileleantraining.com](http://www.agileleantraining.com)). She and **Robert Benefield** provide agile and Lean training and consulting in Western and Eastern Europe, the US, and the Middle East.



This work is licensed under a [Creative Commons Attribution-NonCommercial 3.0 Unported License](http://creativecommons.org/licenses/by-nc/3.0/).

# INTRODUCTION

Attracted by significantly lower costs in places like India, China, and Eastern Europe, many companies have embarked on globally distributed software development initiatives. Unfortunately, many have found that while per-hour development costs are indeed lower, overall project costs can be higher, after factoring in the significant challenges of communication and coordination, the cost of difficulties and delays, and higher project failure rates. Not surprisingly, many organizations have turned to Scrum in hopes that it will enable their distributed teams to achieve significantly better results. This primer outlines practices that can help distributed Scrum Teams excel, and highlights some of the common pitfalls that teams encounter, along with ways to respond to these challenges.

The most important point to start with is that the principles and practices of Scrum in a distributed project are no different from the principles and practices of Scrum in a single-location project: It's simply Scrum, but with added challenges brought on by the distances and differences between locations. The Scrum practices enable teams to deliver customer value early and often, add transparency, surface dysfunction, and drive continuous improvement through a simple framework of inspect and adapt – all of which are even more more acutely needed in a distributed project, but at the same time are logistically more difficult to conduct. The following pages outline practices which can help overcome these challenges, first in **enabling communication** and then in **building trust**. Then, useful tips for implementing the Scrum **roles, meetings, artifacts, and technical practices** are outlined, as well as common pitfalls to be avoided.

In the final analysis, there is no one “right” way to do distributed development using Scrum, other than for teams to start with the principles and standard practices of Scrum, and inspect and adapt to a solution that is well suited to their particular situation – but this Primer provides starting points and ideas that may help speed teams along the path of improvement.

---

## ENABLING COMMUNICATION

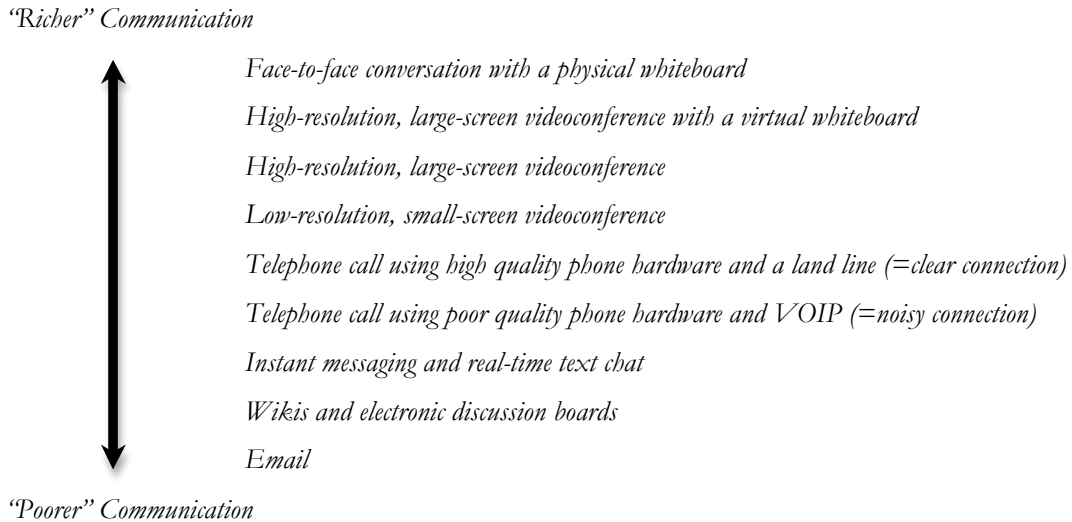
While there are technical issues that need to be considered when using Scrum in a distributed environment, the biggest challenges center around human issues, starting with communication.

At its most basic level, software development is both enabled by, and constrained by, the quality of the communication that takes place among the people involved. Customers form ideas about what they need, and communicate them to the development team; the team-members communicate with each other and with the customer to build functionality that satisfies those needs; the customer communicates feedback to the team about what's been built; and throughout this process, everyone communicates with each other about questions they have, obstacles they encounter, opportunities they see, and how they are feeling (satisfied, concerned, etc.)

Consider a Product Owner in one location and a development team in another location. The quality of the communication between them will directly determine how much business value (in the form of useful, high-quality software) is delivered. Every misunderstanding between the Product Owner and team means a little less value will be delivered; when the team implements a piece of functionality incorrectly, and has to go back and redo it, there is other work that in the end will not be completed. Also, the more effort the communication requires, the less business value will be produced; if the team has to leave 3 voicemails for the Product Owner to get a response to their question, the Product Owner will inevitably get a little less software in the end; the team was spending their time dialing and waiting, not coding! Great software is typically produced only when there is great communication between the people involved, and poor communication will limit the quantity, quality, and correctness of the end result.

So how do we ensure that communication between the Product Owner and team is as effective as possible?

First, there are practical considerations. The various modes of communication – email, telephone, face-to-face conversation – can be placed on a “richness” scale, which looks something like this:



By and large, the higher up this scale you are, the richer and easier the communication, the more natural the interaction and the more immediate and faithful the understanding between people.

Email is, unfortunately, the go-to mode of communication between most distributed Product Owners and teams, and this is a mixed blessing. Its great strength is that it is not dependent on both parties being present simultaneously, and it preserves a record of the discussion that can be referenced later. The big disadvantage of email is that it is often much more time and effort-intensive. A discussion that might otherwise require a single, five-minute telephone chat could easily turn into 10 back-and-forth emails, each cc:ed to other people (thus consuming their time and attention, if even just to hit the delete key). Email conversations also breed misunderstanding, and as a result, unnecessary or unintended emotionality; without the subtle cues of voice intonation and facial expression, one can easily misunderstand the mood, tone, and intent of the writer.

ScrumMasters working with teams and Product Owners that are distributed need to help everyone shift away from email as the primary means of communication. This starts with making live communication as effortless as possible.

First, the group itself (including the Product Owner) needs to agree that wherever possible, conversations should take place live rather than via email. (If the conversation needs to be documented, either party is always free to send a brief email summary after the call.)

It is important for the Product Owner to clearly communicate to the team that it is acceptable to phone with quick, urgent questions without any “pre-scheduling” required – otherwise, many teams will assume that it is *not* ok to call, and will default to email.

Next, everyone’s (and especially the Product Owner’s) desk and mobile phone numbers and IM usernames need to be placed on a wiki or other shared location, along with acceptable outside-of-offices hours to phone with urgent questions, as well as a photo of the person (to remind us that it is in fact a person!). For example:

*Tom (Product Owner)*  
*desk: +1-123-456-5678      mobile: +1-123-456-6789*  
*office hours: Mon-Fri, 8am-6pm PST = 8:30pm-6:30am India time*  
*urgent questions: Call mobile Mon-Fri 6:30am-9:30pm PST = 7pm-10am India time*

In the team work area, there should be a high-quality speakerphone (for example, a Polycom SoundStation) with the speed-dial buttons programmed to the Product Owner's desk and mobile phone numbers (preceded by any long-distance "unlocking" codes), plus a sticker attached to the phone with the acceptable local hours to call (or clocks with the different location times).

In addition, each team-member's desk phone or VOIP application (and if possible, mobile phone as well) should also have the Product Owner's telephone numbers programmed on speed dial.

Enabling easier telephone communication is an important step, but it is not enough. All of the key Scrum meetings – Sprint Planning, Product Backlog Grooming, Sprint Review, and Sprint Retrospective – should be conducted *visually*. The problem with audio-only meetings are many. One misses out on facial expressions and body language entirely. It can be unclear which voice belongs to which person. The natural "flow" and cadence of a conversation is often missing; there are either unintentional interruptions, or people are afraid to speak up for fear of interrupting. If participants have unfamiliar accents, it is harder to understand them without a view of their face as they speak. However, the most significant dysfunctions of voice-only calls is people "multitasking" during the call; without a visual on what they are doing, people will often find checking email or surfing the Internet irresistible, and only pay partial attention to what is being discussed. Participants are effectively only "half-there."

Some companies have invested in sophisticated videoconference equipment, but teams may find it complex and cumbersome to operate, or the conference room where it is located is often booked. It may be more effective to provide the team with an improvised solution as follows:

Video: Skype with a wide-angle high-resolution webcam. (It is important to use a wide-angle webcam – this gives a wider field of view, enabling more people to be seen on-camera)

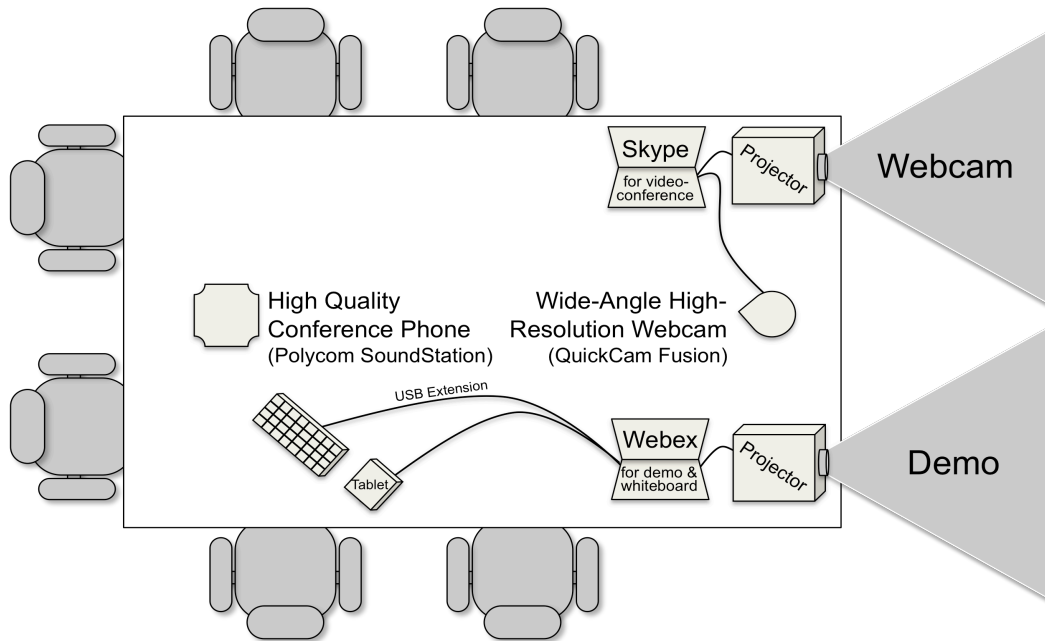
Audio: High-quality conference phone connected via a land-line, with multiple extension microphones for the table. (In some cases doing the audio via Skype is sufficient, but generally a high-quality conference phone on a land-line will produce much better fidelity.)

Ideally, the above equipment should be set up and ready to use at any time in the team room, and this should be replicated at the Product Owner's side. While the quality may not compare with a more sophisticated system, it more than compensates with its simplicity, low cost, and convenience, and it provides the most important visual information: Who is speaking, their expression and body language, and whether people are paying attention. And perhaps most importantly, you are reminded that your colleague is not just a disembodied voice on the end of the line, but a real, live human being!

If the team itself is split between multiple locations, it is strongly recommended to equip each team-member with a webcam and a comfortable, high-quality headset with microphone. This allows for quick, one-to-one audio-video communications at any time, without people even leaving their seats. In addition, there should ideally also be an "always-on" videoconference between the team's locations: a high-resolution wide-angle webcam and large-screen plasma display in each of the team work areas, with continuous Skype video streaming between the two. This serves as a "window" between the two rooms, and because it is always on, it enables instantaneous multi-person conversation and collaboration.

In addition to videoconferencing capability, it is important to also have some type of desktop-sharing software with virtual whiteboard capabilities. Many teams also find it useful to use a low-cost digital tablet for diagramming on this virtual whiteboard.

Finally, for the key Scrum Meetings – Sprint Planning, Product Backlog Grooming, Sprint Review, and Sprint Retrospective – it's helpful to have simultaneous videoconferencing and whiteboarding capability. The following diagram shows a conference room with an ad-hoc setup for doing this, with two projectors side-by-side (one projector displaying the Skype video feed from the other location, and the second projector displaying the shared desktop or virtual whiteboard), plus a high-quality conference phone on a land-line.



Offshore teams may feel uncomfortable asking for these investments in quality communication for fear of being perceived as burdensome or demanding, but when one considers the “big picture,” it is hard to justify *not* making this investment:

*The Pioneer team in Bangalore had been feeling frustrated for some time about the difficulty of long-distance meetings with their Product Owner, Steve. Everything was done by conference call and email, and communication was really quite difficult. The conference phone was not very good – it was just a cheap desk phone with a “conference” button – and the sound quality was very unclear. Everyone was constantly interrupting each other by accident, and sometimes there were long pauses from Steve that made the team wonder whether perhaps he perhaps had them on mute and was typing emails – either that, or he was unhappy with them, and did not feel comfortable saying so – they just were not sure. The communication was always a struggle, and the team felt like it was always difficult to express themselves, there were frequent misunderstandings, and this eventually resulted in the team building functionality that was not quite what Steve wanted. The team’s ScrumMaster, Sanjay, resolved to do something about the situation.*

*The first step was upgrading the technology they were working with. The team felt confident that if they could make the four key meetings of each Sprint visual, it would really improve the quality of their communication with Steve. Sanjay gathered the team and led a brainstorming session to come up with a “shopping list.”*

Wide-Angle Webcam × 2	\$150	[for the team plus Steve]
High-Quality Conference Phone	\$300	[for the team]
Digital Tablet × 2	\$150	[for the team plus Steve]
GoToMeeting subscription (1 year)	\$150	[shared]

TOTAL	\$750
-------	-------

*Sanjay took this list to his department manager, Vikram.*

*“Vikram, we’re having some serious communications issues with our customer Steve, and the team and I feel that we need to upgrade our communication tools. I need your approval to spend \$750 on the following items.”*

*Vikram studied the list.*

*“Unfortunately, that’s rather a lot of money for us to spend on things that aren’t really necessities. I don’t think I can approve this.”*

*Sanjay thought for a moment, then took out a sheet of paper and a pen.*

*“Vikram, think of it this way. How much does the team cost the company? Let’s include salary, benefits, rent, electricity, equipment, everything. It’s about \$33,000 per person per year, and we have 6 people on the team. So the total is...”*

*[writing]       $6 \times \$33,000 = \sim\$200,000$*

*“So for this \$750 purchase to make sense financially, it has to improve our effectiveness by \$750 / \$200,000. This comes to 0.4%. This means that it will pay for itself with even a tiny improvement in our effectiveness. And if better communication with the customer improves our effectiveness even more – let’s say by just 10% or 20% -- then this could be the single best investment we make all year!”*

---

## **BUILDING TRUST**

The other key enabler – or constraint – for distributed projects is how much trust there is between the Product Owner and the team. Inevitably, in the course of day-to-day cooperation, there will be bumps in the road. Miscommunication will happen, misunderstandings will occur, mistakes will be made, and myriad other problems will come up. If there is a strong human relationship between the Product Owner and team, these issues can simply be taken at face value; they will remain routine misunderstandings or mistakes which can be overcome. However, if there is not a strong relationship, over time these issues tend to pile up and become “evidence” in a dark narrative about the other party: that they are incompetent, dishonest, or even crazy – or even all three! Unfortunately, it is extremely difficult to “unthink” these thoughts about others once they have taken hold; at that point, the relationship has reached rock bottom, and every interaction will be difficult and minimally productive, and significant time will be spent documenting interactions rather than building software. It is not uncommon to find distributed projects where the Product Owner is utterly convinced that the team is incompetent, and the team is utterly convinced that the Product Owner is incompetent. With limited information about the other person, we often tend to fill the gaps with fears rather than facts; when someone does the wrong thing, we are apt to take it as evidence that they do not know what they are doing – which is what we fear most – rather than other possible explanations (such as: they did not fully understand what was expected of them and were afraid to ask for clarification).

The only way to reduce the risk of these misapprehensions taking hold is by building a foundation of trust between the Product Owner and the team.

This begins with a human relationship between the two. One of the most critical steps for the success of a distributed Scrum project is for the Product Owner and team to come together in person at the beginning and spend quality time sharing key project information and building a relationship with each other.

This is particularly important at the beginning of a major project; in addition to starting the relationship, there is also a large amount of information that needs to be communicated. First, the Product Owner needs to provide the team with a clear and comprehensive understanding of the overall vision, purpose and goals of the project; this context gives the team a strong foundation for their day-to-day work, and also helps build their motivation and drive. Next, the Product Owner and team will have an opportunity to go through the items on the Release Backlog (the subset of the Product Backlog targeted for the nearest release), and discuss in detail the features and functionality required. This gives the team a vastly deeper and more nuanced understanding of the project requirements than they could ever derive from a written specification alone. This conversation will also provide more “subtle” information and understanding; for example, about the values, attitudes, and mindset of the Product Owner and team members.

The major objection to this is the cost of the trip in time and money. But let us take a moment to analyze this objection further.

Let's consider the example of a Product Owner in San Francisco flying to visit her team in Sofia, Bulgaria:

<i>Flight, US to Bulgaria</i>	US\$3,000
<i>Comfortable hotel accommodations and meals for 4 days</i>	US\$1,500
<i><u>Ground transportation, visas, other incidentals</u></i>	<u>US\$500</u>
<i>Total cost of trip</i>	US\$5,000

This hypothetical Product Owner is kicking off a 1-year project:

<i>Cost of project</i>	US\$500,000
------------------------	-------------

The cost of the trip is just 1% of the total project cost. Will an in-person project kickoff, knowledge transfer, and relationship-build improve the results of the project by more than 1%? Guaranteed. In fact, anecdotally, the overall project ROI improvement this produces is probably more on the order of 30-50% or more.

Still not convinced? Let us go further with our analysis. The cost of the project is US\$500K, but that is not the *value* of the project. What is the business value this project is going to be producing? What is the cost to the business if this project fails? Likely an order of magnitude greater than what it will cost to complete. Let us assume this hypothetical project, if successful, will enable \$4 million in new revenue and \$1 million in cost savings; its total potential value is \$5 million. Compare that to the cost of the Product Owner making the journey to India for the project kickoff:

<i>Value of project</i>	US\$5,000,000	100%
<i>Travel cost for project kickoff in person</i>	US\$5,000	0.1%

The bottom line is that there is just no excuse for the Product Owner not to join the team in person for the project kickoff. If the project matters, that is – and it is worth noting, the willingness of the Product Owner to come in person for the kickoff sends a very clear message to the team that “this project matters”.

To be really useful, this project kickoff must include more than just workplace interaction; the Product Owner and team should plan informal outings away from the office, to give them the opportunity to interact not just as co-workers but as people. The ideal itinerary for “human meshing” could include group outings to tourist sites, a bowling outing, dinners together, possibly even a visit to one of the team members’ homes. (It is important to note that these excursions are for the Product Owner and the team to bond – not for senior management to dazzle their client.) The bonding experience can be particularly important when the Product Owner and team are from very different cultures. Sometimes, accentuating some of the cultural differences can have benefits. For example, a team in Bangalore took their Product Owner Phil, who was visiting from the US, to a local temple for a *Puja* (religious ceremony) to bless the success of the project; it turned out to be both a very memorable experience for Phil, and also a strong bonding event for the entire group. Management needs to understand that while this looks like “non-work time,” it is actually a critically important investment in the vitality and success of the project.

In addition to the human bonding, the in-person visit by the Product Owner can achieve other goals as well. First, it lay the groundwork for bridging some of the cultural differences between the two locations. For example, in the business culture of many Asian countries, there is a taboo against sharing bad news bluntly, or appearing over-emotional. In other countries – the US, for example – frankness is more the convention. For example, when a team in Delhi says to their Product Owner, “we have a bit of a concern about X,” what they may really be expressing is “we see a very serious issue with X that needs immediate attention”; unfortunately, what their Product Owner in Silicon Valley hears is “this is a minor worry, let’s not waste time on it”. Similarly, when the Product Owner in Silicon Valley says “The situation with Y is a complete disaster!” what he may be trying to convey is “There’s a problem here, let’s really focus hard on solving it,” but unfortunately the team in Delhi interprets the statement to mean “Go pack up

your desks, you're all out of a job!"). None of these people are intentionally misleading – rather, they're expressing their thoughts using the norms of their particular locale.

One final benefit of the Product Owner traveling to the team's location is for him or her to experience first-hand some of the challenges the team experiences working in their location. A Product Owner from the US who is used to a 30-minute commute to the office, uninterrupted power, continuous air-conditioning in the summer, and fast, reliable broadband likely assumes his counterparts on the other side of the world experience the same. After an in-person visit, though, he might realize that his team faces a 2-hour commute in each direction, power and broadband connectivity that comes and goes, government-mandated bureaucracy for purchasing hardware, and a host of other daily challenges.

Apart from this initial visit by the Product Owner, it is also important that the entire group co-locates again every 3-4 months, or after logical milestones have been achieved. If the Product Owner has traveled to the team's location for the project kick-off, then it may be helpful for the team to travel to the Product Owner's location after the first release, and spend an entire Sprint (or at least a week) together. This time together is used to re-emphasize the "big picture" vision and goals, kick off the forthcoming release, discuss any major issues or upcoming decisions with their colleagues, and generally re-sync the two locations with each other. It will also be very helpful if the development team members from the offshore location get to interact with business stakeholders and end-users of the software they are building; this provides them with a better understanding of the real context of their work, and goes a long way towards reinforcing a common goal and "one-team" mindset.

In addition, team-members who join the project at a later stage should also travel to the onshore location at the first opportunity; it is enormously helpful for them to hear the context, business-drivers and vision of the project from the Product Owner and other important stakeholders directly.

The second part of building a relationship of trust is developing openness and honesty between the players. Teams are often fearful about being open with the Product Owner, especially if that person is the customer; the team worries that if they raise a difficulty or concern, the customer will be upset or disappointed, and may even complain to management. As a result, many teams invest a significant amount of effort in trying to create the *appearance* that everything is going well. Indeed, the less well things are going, the more effort has to be invested in maintaining this appearance – effort which, ironically, would be much better spent trying to solve the problem. Teams are often afraid to even ask questions, concerned that their uncertainty will be perceived as incompetence – and as a result, questions go unasked and the team makes assumptions, produces the wrong thing, and ultimately creates the very perception they were trying all along to avoid!

So how does one avoid this syndrome? The Product Owner must communicate to the team in no uncertain terms that he or she wants to hear good news as well as bad, that nobody will be punished for honesty, and that the only "dumb" question is the question that goes unasked. Then after "talking the talk," the Product Owner must "walk the walk" – he or she needs to constantly press the team to ask questions and raise concerns, and when the team does bring up problems (which will happen tentatively at first), respond in as positive and solutions-oriented a way as possible.

*Tom was becoming increasingly concerned about his team in Shanghai. The project they were kicking off was going to be extremely challenging technically, and they would be working in a new domain and using tools that were new to them. But what concerned Tom the most was the fact that the team was expressing no worry or doubt about this at all. On every call they seemed to have the stance of "we don't see any problem at all; we're fully confident that we will be able to master these new areas, and we don't have anything to worry about." Tom felt this attitude was unrealistic, almost to the point of being irresponsible; if the team truly had no worries, then they were living in a fantasy land!*

*In reality, of course, the team was very concerned. But the last thing they were going to do was communicate this to their customer. If Tom found out that they were concerned, who knows what he might do! Complain to management? Try to cancel the contract? They certainly did not want to find out.*

*Tom tried giving subtle hints and suggestions, but in the end, he decided to really share what he was thinking during a call with the team.*

*“Guys, there’s something I’m really concerned about, that I really need to talk to you about.”*

*Hearing these words, the team stiffened.*

*“On every call with you guys, all I hear is “things will be fine”. But let me tell you, this is a big project with a lot of risk. I’m losing sleep over it, and we’re not even a month in. And what worries me most is that I’m not hearing any of the same concern from you guys. That’s making me wonder if maybe I’m the only one who sees how hard this is going to be. Do you guys get it? Are you guys at all concerned about this?”*

*There was silence. Then Lee Wei, the most experienced of the developers, responded. He sounded a little tentative.*

*“Of course Tom, we have some concerns…”*

*Tom felt a little relieved that the team was acknowledging what he was saying.*

*“Okay, so tell me about your concerns.”*

*There was more silence, and then Lee Wei continued. There was a major issue the team had been worrying about for some weeks now.*

*“One concern we have is that the database may not give the performance we need under heavy loads.”*

*This caught Tom off-guard. At no point in the discussions so far had there been any mention of performance worries. Tom’s first instinct was to respond with “Why the heck didn’t you bring this up sooner!,” but he caught himself. He took a deep breath.*

*“Okay. Guys, I have to admit, I’m pretty surprised to hear this. I didn’t realize that there was a concern here, and this is something potentially very serious.”*

*There was silence, and the the team braced themselves for what came next.*

*“But I have to say, I am really happy you guys shared this with me. Now that I know about it, we can do something about it. Great performance is make-or-break for this project, so we need to get to grips with this issue. So what could we do to answer the question now? Let’s create an item to go at the top of the Product Backlog…”*

In this scenario, how will Tom’s reaction affect the team’s behavior going forward? What would have happened if Tom had reacted badly to the team’s revelation?

To establish a foundation of trust at the beginning of a long-distance working relationship, it can be very helpful to have an open and direct conversation about what everyone is committing to, and what each expects the other to do. This could simply take the form of a conversation, or it could come in the form of a “working agreement” between the team and the Product Owner.

Some examples of such commitments among real-world Scrum Teams are:

*We commit to be honest with each other. If we have a concern, a doubt, a worry, or if we see a problem, we commit to surface it to each other immediately.*

*If we are unhappy about something that has happened, or something that the other has done, we commit to surface this immediately to each other.*

*We commit not to escalate a problem to upper management without first trying to work it out with each other. If an escalation does become necessary, we commit to letting each other know in advance, so it doesn't catch anyone by surprise.*

*We recognize that people make mistakes and have misunderstandings, and that the important thing is to find and fix the mistake or misunderstandings as quickly as we can. For this reason, we commit to each other that there will be no retribution for surfacing a problem or a concern, a mistake or misunderstanding, or for speaking honestly.*

---

## DISTRIBUTED SCRUM PRACTICES

In a distributed environment, all the standard practices of Scrum – the roles, meetings, and artifacts – are present. However, it may be necessary to adjust how those practices are implemented, to overcome differences in timezone and geographic location.

### Sprints

There is no “best” Sprint length to use, either in a co-located or a distributed environment. Longer Sprints (3 or 4 weeks) enable teams to produce larger increments of functionality each Sprint, and Sprint Planning and Review / Retrospective (which typically involve early morning or evening meetings for everyone involved) occur less frequently. Unfortunately, both of these benefits can create other drawbacks. Because of the communication problems that flow from having the participants in different locations, it is far more common to discover misunderstood requirements when we reach the Sprint Review. In a 4-week Sprint, it is possible that twice as much of the “wrong” functionality will have been built than would have been built in a 2-week Sprint. Additionally, a 4-week Sprint offers half the frequency of inspect-and-adapt cycles for the team’s practices, so many teams find they have fewer opportunities to surface and address dysfunctions.

One solution is to start with 2-week Sprints, and focus initially on mastering the ability to deliver increments of potentially shippable product (possibly very small ones) by the end of a Sprint. A number of Sprints’ worth of inspect-and-adapt may be required for the team to achieve this, but once they have succeeded, they can shift to a longer Sprint length, and be able to deliver larger, more satisfying increments of functionality.

### The Product Owner

One common distributed Scrum configuration is to have a Product Owner in one geographic location, and the team in one or more other locations. It becomes even more important to have an actively involved and committed Product Owner in this situation. Often, organizations will select an individual in the same location as the team and put them in the role of “Proxy Product Owner,” to provide guidance to the team when the actual Product Owner is not available. This can work, but it often brings with it a new set of challenges; the proxy will rarely have the depth of domain expertise or the decision-making authority to give definitive answers to the team, and the risk is that they give answers that are later reversed by the “true” Product Owner (resulting in wasted effort and discouragement for the team), or they serve purely as an intermediary between the team and the “true” Product Owner, which dramatically slows response times, and introduces errors and misunderstandings in both directions. If there is a proxy working with the team, it is probably important to still maintain a daily flow of questions and answers between the team and the “true” Product Owner – for example, by agreeing on a 30-minute overlap between the Product Owner’s and the team’s working hours, during which time either can phone the other with questions, or having the ScrumMaster compile and email the Product Owner at the end of each day a list of open questions, which ideally the Product Owner will have guidance waiting when the team returns to the office the next day.

## The ScrumMaster

The role of the ScrumMaster becomes even more critical in a distributed project, because the “dysfunctions of distance” – for example, the difficulty of communication -- require an even more active and tenacious commitment to openness and inspect-and-adapt. Day-to-day, distributed projects also tend to have a greater-than-usual load of impediments, obstacles, and disruptions that will require the ScrumMaster’s attention and effort.

If the Product Owner is in one location and the team is in the other, the ScrumMaster should be located where the team is. While an onshore ScrumMaster may be able to help an offshore team with some types of issues, he or she will unfortunately be entirely absent from the realities of the team’s day-to-day worklife, and thus they will be far less useful to the team. The critical ScrumMaster duties of coaching the team, helping remove impediments, and protecting the team from disruption will essentially be absent if the ScrumMaster is located far from the team.

If the team itself is divided between multiple locations, there should be a primary ScrumMaster designated for the team overall, but it will probably be helpful for each location to have a team-member playing the role of “local” ScrumMaster during that location’s working hours.

## The Team

When the team itself is split between multiple locations – for example, several team members are located in China, and several team members are located in the US – the challenges of development are often multiplied further. The level of coordination, cooperation, and team-work that is necessary to deliver working software every 4 weeks or less is even more demanding. While there are case studies that describe exceptional results with this model, it takes real commitment and a significant investment in the working relationships, skills, and tools used by the various team members to deliver that level of success.

To begin with, team members will typically need to spend periods of time working side-by-side with each other, especially at the beginning of the project. An excellent practice is for the team to be colocated for the entire first Sprint of the project, and ideally the first two or three Sprints. This allows the individual developers to build working relationships with each other, as well as trust and visibility into each others’ skills, strengths and weaknesses. In addition, the team will develop a set of working agreements and set standards for their “definition of done,” quality, coding conventions and other development practices, tools, escalation, overlapping work hours, and other necessities.

In addition to colocating the team for the first Sprint or more, the distributed teams that succeed with Scrum typically have some sort of “ambassadorship” practice, where team-members are constantly traveling to the other location for periods of time working side-by-side with their distant colleagues. When these “ambassadors” return home, they bring with them knowledge and values that will inform the work of their local colleagues, and they will also be able to function as points of contact for their remote colleagues when issues arise. This ambassadorship is ideally a continuous practice, with one team member placed at the other location at all times, in rotation.

The immediate objection to this constant travel is “it will cost money and time!” The simple response: Yes, but it is cheaper than the alternative, which is getting much less business value from the project. Without face-to-face contact and high-quality working relationships, the team will produce either less software, lower quality software, or functionality that’s less right for the customer needs – or all three. A team of 6 developers with a generous travel budget will probably produce much more business value than a team of 7 developers with no travel budget.

A common dysfunction when the team itself is split between two locations and are not properly bonded is that “one team” actually operates as two teams. Team-members may form “cliques” by location, and miscommunication or miscoordination between the locations may give rise to mistrust and conflict. It is also possible that if a portion of the team is located onshore, they will

be in closer communication with the Product Owner, and as a result they will have an information advantage over the offshore developers. While one would *hope* that this could benefit the entire team, it can sometimes do the opposite, driving a wedge between the two groups of developers, with the offshore group being seen by onshore as clueless and always a step behind, and the onshore group being seen by offshore as hoarding knowledge and looking out only for themselves.

Rather than trying to work as a single team, it may be more effective to form into separate Scrum teams, one per location, and as loosely coupled with each other as possible. Each team should be fully cross-functional, and should be responsible for producing entire pieces of functionality, not simply doing a particular activity (coding, testing, etc.).

## Technical Practices

In addition to strong working relationships and effective communication, there are a number of other practices which are helpful for the success of teams doing Scrum in a distributed environment.

For teams that are split between multiple locations, it is very important that all team members have the same development environment (same IDE's, plug-ins, code quality checking configurations, etc.), and work on shared DTAP (Development Test Acceptance and Production) servers; this removes ambiguities and reduces problems caused by inconsistencies between the locations.

As is true for colocated Scrum teams, the practice of Continuous Integration is extremely helpful. In Continuous Integration, new or changed code is integrated early and often; commits trigger an automated build-and-test cycle, allowing integration problems to be detected and corrected immediately. By doing this frequently and with small increments of change, problems can be found when they are smaller and more manageable, so less time overall is spent in the finding-and-fixing activities. It is important of course to have the discipline to immediately resolve the issues, and many teams agree on conventions, such as “you can't leave the office with a bad build unfixed”; the last thing the developers on the other side of the globe want is to start their day with this problem. Continuous Integration also enables a less rigid and more emergent approach to defining and building interfaces; with more confidence in their ability to find and fix problems quickly, teams can be more dynamic in the way they work, and work together more smoothly.

It is important that the required knowledge, capabilities, and skill levels are evenly distributed across both locations; an imbalance will possibly result in less value produced each Sprint. For example, having senior architects and designers onshore and junior developers offshore will likely result in a lot less software being produced, and a much more dysfunctional relationship between the two locations, than if a more experienced team was recruited offshore. In the event there are skills or capabilities which cannot be shared across both locations, this should be made clear and visible to all stakeholders as a possible impediment.

When deciding who works on what during a Sprint, it's important that team members from one side not take all the work in a particular area on a regular basis; for example, onshore team members always taking UI tasks, while offshore team members always working on back-end services. While this may sound like a simpler approach initially, it generally results in silos that undermine the “shared responsibility and ownership” mindset of the team.

When the team is physically split, real-time communication tools becomes critically important. At a minimum, teams require the following:

- Instant Messaging client (not only for communication and easy transfer of text, but also for indicating their presence online to other team members)
- Comfortable, high-quality headset and VOIP client, to make conversations with remote team-mates quick, easy, and free (with “always on” as an option)
- Webcam and Skype, for instant videoconferencing
- Shared digital whiteboard, for design and architecture discussions

- Desktop sharing solution (for example, a VNC client)
- Team wiki (with not only project details but also personal info about each team member)
- Shared bug tracker
- Team calendar, showing release dates, Sprint dates, local holidays, and vacation plans
- Team mailing list, to which all key emails are cc:ed.
- Build status alerting device (such as Nabaztag) at all distributed locations

(The upcoming **Scrum Technical Practices Primer** will provide more in-depth explanation and guidance around the tools, practices, and techniques referenced in this section.)

## Sprint Planning Meeting

One of the practical conflicts in distributed Scrum is the fact that (a) more time is typically needed to properly complete the Sprint Planning, Review, and Retrospective meetings than in a colocated environment, and (b) we often have less time available for these meetings (due to lack of timezone overlap). This is less of an issue for projects between Europe and Asia, for example, but for the US and Asia, it can become a real impediment to success.

One approach that can help is to split the longest of the meetings – Sprint Planning – into 3 shorter sessions over the span of two days, as follows:

Sprint Planning Part 1 (1 hour timeboxed)      Weds 8am New York / 6:30pm India  
Product Owner walks the team through the items at the top of the Product Backlog, team asks questions, clarifies their understanding, and make suggestions.

Sprint Planning Part 2a (2-3 hours)      Thurs India workday hours  
Team starts doing an initial analysis, task breakdown, and estimation of the items at the top of the Product Backlog. They come up with a list of questions for the Product Owner.

Sprint Planning Part 2b (1 hour timeboxed)      Thurs 8am New York / 6:30pm India  
Team and Product Owner discuss the team's open questions, and the team decides their commitment for the Sprint

Work begins      Friday India workday hours

On the days when the team will be staying late for the evening meetings, it is important that they maintain a reasonable workday by coming into the office later in the day than they would normally; a recurring schedule of 12-hour days will very quickly start to exhaust the team, and cause productivity and morale to drop, and mistake rates to go up.

Sprint Planning part 2 is typically also be used for design discussions, to review of major component changes, and other issues. This allows everyone to express their thoughts, and also helps in establishing a clearer vision for the implementation, which should result in more consistent quality from everyone.

Lastly, many teams find it very helpful to budget a generous amount of time during the Sprint for doing Product Backlog Refinement (aka Grooming) with the Product Owner. Together, they look ahead to upcoming Product Backlog items, gain a clearer understanding of them, split larger Product Backlog items in smaller pieces, and prepare them to be considered in the next Sprint Planning meeting; the more time spent on these activities, the more easily and quickly the next Sprint Planning will go.

## Daily Scrum Meeting

If the team is colocated together, and the Product Owner is in a different location, the first question is whether the Product Owner should be invited to join the team's Daily Scrum Meeting. There are pro's and con's to this. Some teams find it helpful to have the Product Owner join the meeting, so he or she is aware of their impediments on a day-to-day basis, and to

have a window of time after the meeting each day for live discussion with the Product Owner. However, there can also be downsides to having the Product Owner join. It often costs precious minutes each day, as either for the team or the Product Owner waits for the other to join the meeting. Having the Product Owner joining the Daily Scrum Meeting can also make the team feel like they're being monitored and overseen, and this adds pressure and stress, invites micromanagement, and can reduce the team's sense of responsibility and self-organization. Third, if due to the presence of the Product Owner the call has to take place in the evening hours for the team, it will hurt morale and significantly accelerate burnout, with the end result of much less business value being produced. If the Product Owner is anxious to know how the Sprint is progressing, it may be much less disruptive to have the ScrumMaster simply email a camera-phone photo of the team's Sprint Burndown Chart.

If the team itself is in different locations, the ideal is to hold the Daily Scrum Meeting live via webcam each day, at a working hour that overlaps for both teams; this is feasible between Europe and India, or West Coast USA and China. If there is no working hour that overlaps, here are several options to try:

- Hold the Daily Scrum meeting live via webcam or conference call each day at an hour that is inconvenient for one side or the other. It is very important to rotate the burden of the inconvenience from one side to the other every week or two. The main downside of this approach is the daily outside-regular-work-hours for one part of the team; this will likely add stress, hurt morale, and reduce productivity over time. If this option is chosen, be mindful of different cultures' meal-schedules; for example, in the US, it is common to eat dinner around 7 or 8pm, while in India, it is more typical for dinner to take place at 9 or 10pm.)
- Hold the Daily Scrum meeting live in each location at a time that's convenient for that location, but have one team member write a summary of the updates, and email them to a team-member in the other location, to read aloud at the start of their Daily Scrum meeting.
- Hold the Daily Scrum meeting using recorded reports. The team members in each location will do their Daily Scrum meeting at a time that is convenient for them. At the start of the meeting, they will use a cameraphone to record video of their updates, and the video will be emailed to the teammates in the other time zone, and played at their next Daily Scrum meeting. Repeat in reverse.

## **Sprint Review**

Sprint Reviews serve the same purpose for a distributed Scrum project as they do for a colocated one: To enable the Product Owner and team to inspect and adapt what has been produced in the current Sprint, and collaborate about what could be done next. In a distributed Scrum, features may be less "right" the first time they're shown, because clear and complete communication between the Product Owner and the team is made more difficult by the distance. This is one of the realities of distributed development, and the Product Owner should build into the release plan buffer to account for the additional rework that will be required, as items are placed back onto the Product Backlog for improvement.

It is very important that the Sprint Review be planned for a time when the entire team – including all onshore and offshore members – can participate together. All members of the team should feel like full participants in the Sprint Review, and should be able to hear first-hand the comments about has been produced, share their own opinions, and join in the discussion. Ideally, the demoing of the new functionality should be done by team members from both the onshore and offshore locations. Having the onshore team demo everything can often create negative feelings for the offshore team members, who may feel like they're not receiving recognition for their work or are being treated like "second-class" team-members.

## Sprint Retrospective

The purpose of the Sprint Retrospective is for the team, the Product Owner, and the ScrumMaster to discuss their experiences and observations from the current Sprint, identify issues and areas for improvement, and agree on changes to make to their way of working to produce better results in the next Sprint. The more distributed the team, the more issues there will be – and thus, the more thorough and effective the Sprint Retrospective needs to be. The most successful Scrum Teams focus on the “learning” or “experimental” mindset that Scrum enables: identifying problems as quickly as possible, and then “testing” a practical solution in the very next Sprint. Rather than agonizing over what is the best possible way to do something, think of the simplest thing that could work and try it for a Sprint. Shorter-length Sprints may accelerate these improvements, by enabling more rapid cycles of inspect and adapt.

As in colocated Scrum projects, the Retrospective should include the Product Owner; many dysfunctions will play out between the Product Owner and the team, and excluding the Product Owner from the Retrospective will significantly hinder improvement. In addition, it’s very important for the Retrospective to be visual (via videoconference); the subtle cues of facial expression and body language become even more important in difficult conversations.

## Artifacts

In a distributed Scrum project, more written artifacts will typically be used, but just how much and what format should be left to the Product Owner and team to determine. This should not be taken to mean that the written detail is all that is required. Indeed, the presence of more written detail will often mean that more conversation – not less – will be required between the Product Owner and team to achieve an effective shared understanding. The additional written detail simply gives the team a reference tool for answering questions when the Product Owner is not immediately available.

With distributed Scrum teams, aim to share a common vision and break the work into small packages that are easier to inspect and adapt, thus reducing confusion and finding misunderstandings sooner. Product Backlog Items should be short and easy to understand, with clear conditions of satisfaction attached. Pictures in the form of sketches, diagrams and simple mockups can convey a lot of information quickly. While User Stories are a popular and effective format for articulating Product Backlog items, lightweight use cases can also work well. Some teams find that having a demo server where the Product Owner can review functionality on a daily basis can help keep everyone in sync and aligned, and surface misunderstandings sooner.

There are different approaches to managing the various Scrum artifacts. If the team is colocated, and the Product Owner is in a different location, paper may still be the best choice for the Sprint artifacts (for example, the Sprint Backlog and Sprint Burndown Chart used by the team to manage their work during the Sprint), but some type of electronic tool may be necessary for storing the Product Backlog; for example, using a Wiki or a Google docs spreadsheet may be a simple, very low-cost solution. If the team itself is distributed, or if multiple teams in different locations are working together, a more elaborate electronic Scrum information tool will likely be necessary. In every case, though, the thinking should be driven by the dictum “try the simplest thing that could possibly work, and inspect and adapt”. For example, many teams find it helpful to document their agreements with the Product Owner (for example, from a conference call) with some type of written confirmation. A fast, simple way to do this would be to send the Product Owner an email with the subject line “agreements – conf call – aug 11,” with a very brief, bullet-pointed list in the body; the email would be cc:ed to a team blog, so everyone could refer back to it.

## Product Backlog Refinement

Distributed teams find it particularly important to devote time during the Sprint (typically 5-10% of their availability) to work on Product Backlog Refinement with the Product Owner. New items and items that have changed significantly will be estimated by the team, large items rising in

priority will be split into smaller items, and the team will have time to start thinking about how they will approach upcoming Sprints' work. Items that are unclear or too big can be given back to the Product Owner to refine further, and when there is a difficult technical issue or uncertainty, the team can plan a "spike" within an upcoming Sprint to do technical exploration, and gain enough of an understanding to estimate and later implement the item.

## **Scrum of Scrums**

When multiple Scrum teams in different locations are working together on a project, there are three commonly used techniques for coordinating their efforts, all of which can be used together.

The first is enabling and encouraging informal, lateral communication between members of different teams. This is often overlooked, but it is one of the most powerful tools for day-to-day effectiveness. When a team or team-member is blocked by another team, their first step should be to reach out to someone on that team, and this should be made as easy as possible. There should be a project-wide Wiki set up, that everyone on the project has access to. Under each team is listed team-member contact info (including email but also IM, VOIP and mobile phone numbers, plus typical working hours and urgent question contact hours translated into the various teams' timezones) as well their particular domains or areas of expertise.

The second technique is Scrum of Scrums. This is a practice where a representative of each team (selected by their team-mates) meets with the other teams' representatives on a regular schedule (typically 2-3 times per week, but it could be daily if necessary) to update each other on progress, surface and resolve inter-team blocks and dependencies, make cross-team technical decisions, and otherwise provide a forum for cross-project visibility and impediment resolution. When teams are in significantly different timezones, it is important that the Scrum of Scrums meeting time be rotated, so that the pain of late-night calls is shared among all the participants; having a subset of the group perpetually more inconvenienced than others can breed resentment that will often start to manifest in other kinds of dysfunction.

The third technique is to establish cross-geographic "communities of practice," to enable team members with particular specialties (for example, architecture) to work across team boundaries and together guide the overall direction and evolution of the project. These groups inspect and adapt to find the right composition and meeting frequency.

# RECOMMENDED READINGS

## Articles and White Papers

*Distributed Scrum: Agile Project Management with Outsourced Development Teams* (Jeff Sutherland, Anton Viktorov, Jack Blount, Nikolai Puntikov, 40th Annual Hawaii International Conference on System Sciences (HICSS'07), 2007)

*Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams* (Sutherland, J. Schoonheim, G. Rustenburg, E. Rijk, M.)

*Fully Distributed Scrum: Linear Scalability of Production between San Francisco and India* (Sutherland, J. Schoonheim, G. Kumar, N. Pandey, V. Vishal, S.)

*Using an Agile Software Process with Offshore Development* (Martin Fowler, at [martinfowler.com](http://martinfowler.com))

*Scaling Agile* (Jim Highsmith, Cutter Consortium Agile Product and Project Management Advisory Service E-Mail Advisor Series)

## Books

*Succeeding with Agile: Software Development Using Scrum*, by Mike Cohn (Addison-Wesley Professional, 2009)

*Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*, by Craig Larman and Bas Vodde (Addison-Wesley Professional, 2010)

*A Practical Guide to Distributed Scrum*, by Elizabeth Woodward, Steffan Surdek, Matthew Ganis (IBM Press, 2010)