Patterns for Managing Distributed Product Development Teams

V. Bricout, D. Heliot, A. Cretoiu, Y. Yang, T. Simien, L. Hvatum

Sugar Land Technology Center, Schlumberger Oilfield Services

In recent years, an increasing number of companies and organizations have put in place geographically distributed teams to carry out development of various products – in particular software products. This trend has economic drivers (globalization, outsourcing) and has been facilitated by the IT revolution (better, cheaper communications). However, it has brought new challenges for the organizations to manage those distributed teams effectively [11].

Schlumberger is an international oilfield service company, present in 127 countries, with 19 R&D centers in Europe, North America and Asia. Over the past several years, it has run a variety of geographically distributed development teams. Experience from managing these teams was collected and discussed at several internal workshops [1,2,3,4,5,6,7,8]; specific resources were also allocated to evaluate and recommend anything from tools and licenses (VC systems, mobile phones, VPN, instant messaging, etc.) to techniques and best practices [4].

The collection of Patterns we present here is intended to capture that experience for two purposes: first, to make it available within our company to new teams and new team members involved in distributed team projects; secondly, to enable a communication with other companies that are working in a similar fashion.

It is worth mentioning that the project teams from which we draw our experience were not just geographically distributed, but culturally diverse and often in very different time zones. To be more specific:

- Much of the gathered experience is from working with teams in two engineering centers: one in Houston, Texas and the other in Beijing, China.
- The team member background varied from newly hired, young Chinese developers or Russian interns to American, British, Pakistani or French seasoned developers or domain experts with up to 25 years of experience.
- The time difference between Houston and Beijing is 13 or 14 hours depending on the season, meaning that no regular work hours could be shared between the centers.

Although these teams were primarily consisting of company employees, the ideas are probably applicable in outsourcing situations as well. Several of the authors have a software background that may come through in the writing. The practices documented are still thought to be of general validity for product development.

Thumbnails (29)

The thumbnails below include patterns based on our experience with distributed teams, as well as proto-patterns based on recommendations made to better manage those distribution teams – and currently being tried throughout our organization. The latter are listed in gray.

1. **Commitment from All** – any engineering center involved in the project must have a clear commitment to the project
2. **Defined Organization** – all roles and responsibilities along with the reporting structure is clearly set and well communicated
3. **One Project** – the project team must truly be run as a single team with a single person in charge, and the same explicit objectives and project plan w/ budget, despite not being co-located
4. **Explicit Communication Strategy** – all means of communication and how/when they will be used is clearly defined
5. **Early Bonding** – make sure you do what it takes to really create a team, like spending time together face to face and in social situations
6. **Face-to-face Every 2 Months** – make sure the team is meeting each other face to face at least every 2 months
7. **Iterate as you Meet** – align the length of the iteration with when the team is meeting next
8. **Common Terminology** – make sure you do not get local "dialects" and that everyone understand the lingo of the problem domain
9. **Continually Aligned Process** – spell out the fundamental values, methodology and techniques used and keep it up-to-date for all team members
10. **Local Guru** – a person in each location who is on top of all issues on plans and status, requested changes, whom to contact for domain expertise etc.
11. **Responsibility Model** – code ownership and requirements ownership clearly defined
12. **Common Baseline Management** – set up for distributed development from the beginning
13. **Prepared Troops** – make sure all are trained on distributed development from the start of the project
14. **Global Focus on Risk** – globally understood and tracked
15. **Common Storage** – make sure there is only one project web page, only one documentation storage, a master storage of the code and issues (bugs and enhancements) etc.
16. **Common Development Environment** – use the same tools and technologies
17. **Distributed Credit** – make sure all team members get credited for success

18. **Culture Awareness** – some specific training may be necessary to learn to "read" team members from a different culture, and to ensure respect for others
19. **Assigned Mentor** – give a new team a mentor with experience in distributed development as mentor
20. **Prepared Company** – the overall structure and processes in the company may need to be adjusted to accommodate and support distributed teams
21. **Prepared Workspace** – design a work environment that supports the way the distributed team is working
22. **Pilot Team Space** – to gain experience with specially designed work areas, try out the solution on one team first
23. **Flexible Hours** – make sure it is culturally acceptable within the team and within the engineering community that the team members can be flexible in work hours to enable them to work from the office or from home at hours that align with the other center
24. **Frequent Short Meetings** – combine frequent planned and spontaneous meetings to keep the team focused
25. **Temporary Engagement** – short-term assignments at the other location will enable team members to get to know other team members better
26. **Carefully Selected** – team members should be screened for personality and the team carefully built over time for maximum cohesion
27. **Conflict Management** – a pre-defined way of managing conflicts, possibly involving an independent party outside the core team
28. **Social Funds** – make sure some funding is allocated to improving the personal relations between the team members (team building, celebration of achieved milestones etc.)
29. **Join for Completion** – bring the development team together with testers on a single site for completion

The above patterns and proto-patterns can be grouped as follows, where the same pattern can belong to several different groups:

Organization

Patterns that deal with how to set up your organization to support the distributed development:

Commitment from all (1)	Local Guru (10)
Defined Organization (2)	Responsibility Model (11)
One Project (3)	Prepared Company (20)

Communication

Patterns that deal with communication within the team and between the team and other parts of the organization:

One Project (3)
Explicit Communication Strategy (4)
Early Bonding (5)
Face-to-face Every 2 Months (6)
Iterate as you Meet (7)
Common Terminology (8)
Continually Aligned Process (9)
Responsibility Model (11)
Culture Awareness (18)
Flexible Hours (23)
Temporary Engagement (25)
Social Funds (28)
Join for Completion (29)

Training and Mentoring

Patterns that apply both the team and to the overall organization to prepare individuals and entities that are to take part in the distributed development:

Common Terminology (8)
Local Guru (10)
Prepared Troops (13)
Culture Awareness (18)
Assigned Mentor (19)
Prepared Company (20)

Process

Patterns that apply to the (iterative) development process as applied by the team:

Explicit Communication Strategy (4)
Early Bonding (5)
Face-to-face Every 2 Months (6)
Iterate as you Meet (7)
Continually Aligned Process (9)
Responsibility Model (11)
Common Baseline Management (12)
Global Focus on Risk (14)
Prepared Company (20)
Flexible Hours (23)
Frequent Short Meetings (24)
Join for Completion (29)

Team

Patterns that deal with the creation and evolution of the project team:

One Project (3)
Early Bonding (5)
Common Terminology (8)
Continually Aligned Process (9)
Local Guru (10)
Prepared Troops (13)
Distributed Credit (17)
Culture Awareness (18)
Prepared Workspace (21)
Temporary Engagement (25)
Carefully Selected (26)
Conflict Management (27)
Social Funds (28)
Join for Completion (29)

Work Environment

Patterns that apply to the physical environment of the team, as well as to tools and procedures:

Common Terminology (8)
Common Baseline Management (12)
Common Storage (15)
Common Development Environment (16)
Prepared Workspace (21)
Pilot Team Space (22)
Flexible Hours (23)

Acknowledgements

This was our first submission of the work, and our first serious attempt at writing patterns. Our paper for EuroPLoP 2004 was shepherded by Jens Coldewey who did a fantastic job. He convinced us to use the Alexandrian form, and guided us strongly on the structure of the paper, not only drastically improving the EuroPLoP submission but more important creating the foundation for our future work.

The workshop participants at EuroPLoP 2004 gave us very thorough and useful feedback. A special thanks goes to moderator Klaus Marquardt and workshop leader Andy Longshaw. We have decided to incorporate the advice that indicate more fundamental changes in a future rework, and keep this version closer to what was submitted to EuroPLoP 2004.

Warm thanks are going to Linda Rising who is helping us develop the internal patterns culture by training and mentoring us, and who has also given us valuable feedback on the work on several occasions.

Final note! The paper is work-in-progress. These first 3 patterns are submitted to EuroPLoP 2004. We have chosen the Alexandrian form as we find that it lends itself best to this type of patterns. As the work grows more complete, we plan to expand the introduction with suggested sequences of patterns for the different roles in the organization (developer, project manager etc.) as well as the project phase (new, in the middle of development, close to complete, general). We may also include other organizational patterns in these suggested sequences.

Internal references are not available on any public site. Authors can be contacted through e-mail to hvatum1@slb.com, or by mail to the following address:

Lise B. Hvatum
110 Schlumberger Drive, MD-11
Sugar Land, TX 77478, USA

Internal References

1. “Cross-Center Development Rules”, Schlumberger Software Métier publication, September 2003
2. “Recommendations/Action items on Collaboration Technology at SPC”, memo from Sugar Land Product Center, 15 November 2002
3. “Process Tailoring for Distributed Team Projects”, by Denis Heliot, presentation at SLB Software Process & Testing Workshop, September 2003
4. “Global Team Interim Report”, by Yiming Yang, SLB internal presentation, 2003
5. BGC-SPC Collaboration Project Recommendations, SLB internal presentation, April 2003
6. “BGC/SPC Cooperation – Feedback from the WellEye Project”, by Simon Fleury, SLB internal document, March 2003
7. “DecisionXpress – a short note about a joint project between SPC and BGC”, by Pascal Rothnemer, SLB internal document, March 2003
8. “Cross-Center Project Management Procedure”, SLB Beijing GeoScience Center internal document, November 2003

External References

9. “A Generative Development-process Pattern Language” by James O. Coplien, PloPD-1 Book p. 183-237, editors Coplien and Schmidt”, Addison Wesley 1995
10. “Using an Agile Software Process with Offshore Development” by Martin Fowler, http://www.martinfowler.com/articles/agileOffshore.html
11. “Can absence make a team grow stronger?” by A. Majchrzak, A Malhutra, J. Stamps and J. Lipnack, Harvard Business Review, May 2004

Frequent Short Meetings

We have come to realize that focus on communication is of key importance when working with remote team members. Not just to discuss system development issues, but also to keep relations between team members healthy over time.

Combine frequent planned and spontaneous meetings to keep the team focused.

***

Your team members are distributed on engineering centers that are geographically far apart. The time difference is significant. The team meets *Face-to-face Every 2 Months*. They spend about 10 days together, aligning design issues and development priorities.

How can you make sure that the global team keeps focused and does not drift apart on priorities and design during the 6-8 week periods between physical meetings?

The on-site team members at each site are co-located in open team space and communicate frequently in an informal way. Continuous team communication is vital for the project to ensure that all are working in the same direction. We want no team member to get stuck on a task. All the small decisions made at each location must be made known globally. There has to be a constant focus on the shared system design. And risk management and project planning must be up-to-date on a global level. But poor communication between the sites is causing problems. Local decisions and assumptions cause a drift in focus between the centers over time. This leads to quality problems in the system. We see both architectural discrepancies, and uncoordinated implementation of functionality that affects the usability.

Daily Scrum meetings are normally focused on development status and priorities. A distributed team also needs to keep a constant focus on the system design. A local team can easily have short, daily meetings (Scrum meetings) and get together spontaneously when needed. Preplanned meetings cause fewer problems for teams that do not share common work hours. This will minimize the negative effect on team member’s private life (families) of having to be available late evenings or early mornings.

In a local team, the members have access to the project architect and the domain experts all day. These people can walk around and help the developers during the workday. In a distributed team, it is necessary to organize more formally to ensure that all the team members have good access to the knowledge resources.

Frequent meetings are important in a global team setting. But meetings take time. Developers are usually negative to time spent in meetings.

Trust and respect between the team members is important to keep a functional team. As a result of the communication problems and the resulting problems on the system, the frustration level is increasing between the developers. It breaks down the team feeling and contributes to the “they” and “us” attitude.

Therefore:

Schedule a number of short focused meetings with the global team. Typically, this could be three per week. The team members can plan for availability up front. Make sure architects and domain experts are present in the meetings. Focus both on progress and on architecture.

By getting the whole team together in very focused events, you have a better chance to limit the effect of sub-teams drifting apart on system design and project planning. With only 2 days between the meetings, reversion/changing of local decisions should not cause too much extra work. The purpose of the meetings is to ensure that the global team keeps up-to-date on tasks, progress, priorities, risks, and design decisions. Less drift inside the team means less potential for later problems. This is very important on the architecture and design.

Focus on system design is kept by the weekly design session.

By having short, preplanned meetings on a regular basis you balance the need for spontaneous communication with the need to plan your work hours. The meetings need to be frequent enough that issues are not forgotten or being solved by workarounds. The meetings also give all developers access to architects and other experts.

Keep the meetings short and to-the-point. If the team is negative to meetings, you can use other terminology. What you are really doing is working! So you can have an “architecture & design session” once per week, led by the architect. And two weekly “workouts” led by the project leader. You could even use Scrum techniques like:

Stand up during the “workouts” to emphasize short duration.

Be very strict on the time/level of detail for each team member’s contribution.

Defer special discussions and follow up to smaller settings outside the global meeting.

By minimizing the frustration caused by drift between the centers, the personal relations between the team members will be less strained.

Preplanned meetings cannot always substitute for the need to get together then and there, and you still may have to do that occasionally. Virtual meetings are not as good as face-to-face for communication, and the team does need to meet physically during the development.

Our US teams that work with the center in Beijing have several meetings per week. This has been an absolute must, as all the heavy expertise was located in the US at the start of this collaboration. The young team members in China were depending completely on frequent contact with the senior developers. We could immediately see the effect of dropping communication for even just short periods. Preplanning of meeting time is not to be avoided. We share no normal work hours between the two centers. Unless people were willing to be flexible on work hours, e-mail was the only means of communicating.

Originator: Lise B. Hvatum

Sheperded for EuroPLoP 2004

Prepared Workspace

When in need of more office space, we decided to take an area of our building with large individual offices and modify it into team rooms. We would not only use the space more efficiently, but had the opportunity to create space that was especially designed for the software teams that were to move in. Two of the teams were working with team members abroad, and the design of the rooms reflected their special needs.

Design a work environment that supports the way the distributed team is working.

***

Your team members are distributed on engineering centers that are geographically far apart. The time difference is significant. The global team solves the need for physical proximity and face-to-face communication by a combination of meeting *Face-to-face Every 2 Months* and *Frequent Short Meetings*.

The project team wants to communicate as a complete team several times per week, and meet at team locations on a regular basis. Can you create a work environment that supports the communication needs and work methods of your team?

Distributed teams that apply agile development methods recognize that good and frequent communication is key to success. They want to minimize the time and effort lost in contacting team members and setting up get-togethers. Communication with the remote team members must be possible at short notice**.** Meeting rooms and communication equipment are frequently not available when needed. They are often too scarce, and require reservations well in advance.

The global team wants an environment (space and tools) that facilitates informal communication between both local and remote team members. The traditional office structure with individual offices does not support an easy flow of communication and flexible solutions for team interaction. Communication tools are generally available in the meeting rooms, not in the office areas.

Visiting team members need to spend as much time with the local team as possible to focus on knowledge of system design and functionality, and to improve team relations. Because the office structure normally does not accommodate visitors, they are placed at whatever available office space we can find. This can be far away from the local team, even in another building.

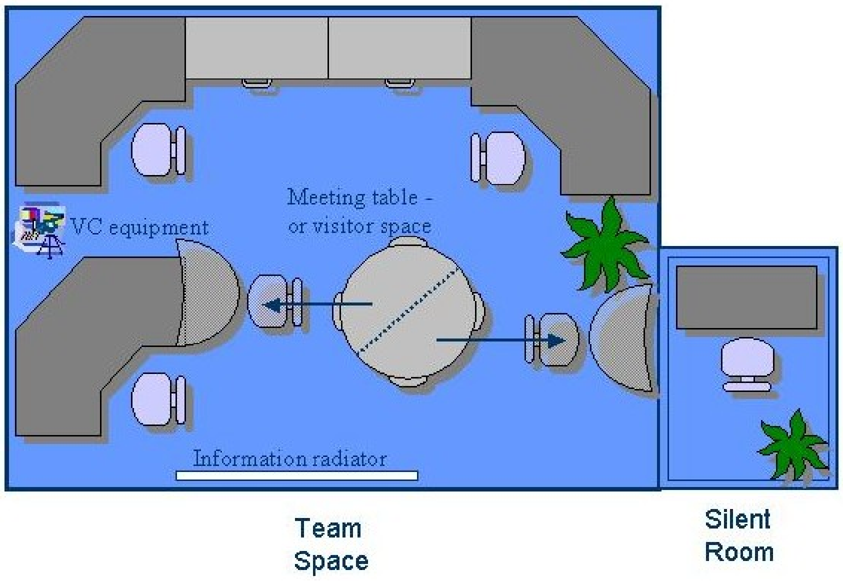
Single offices are usually permanently occupied. Often by key staff (managers, architects and domain experts) whom the other team members need to interact with. Some may need private space because of personnel responsibilities. Some may be negative about losing their private office. The team members all need access to space for private time. Visiting team members may need it even more because their window to call home may be during work hours.

From a management perspective, you want to do what is needed to support the team and increase their possibility to deliver as planned. Still, you may not be convinced about the need for a specially designed team space. The reconstruction period will disrupt the team progress, and you have to make sure the investment will pay off. You need to make sure that the new space is suitable for other teams when the current project is done.

Therefore:

Create a team collaboration space at each location with enough flexibility to accommodate visiting team members and virtual meetings. Design the space to facilitate communication. Make it general enough to accommodate other teams later.

The team workplace could be one room or a combination of private rooms, meeting areas and shared team space. Use flexible furniture that is easy to move and reconfigure. To convert a meeting area to work space for the visitors, use tables that can be taken apart and moved with little effort.



Equip their meeting area with the necessary communication tools (VC equipment, broadband network connection etc.) so that the global team can get together virtually for status and planning meetings as well as for design events. Use Information Radiators[1] so that visiting team members quickly can get the picture of the local teams progress and individual assignments.

In the new work environment, short meetings are easy to do without up front planning and room reservation. Lack of meeting space or VC equipment does not delay the development progress. Even if more tools like VC equipment are needed, this could ambulate between a few teams to reduce cost.

The environment and tools support the work methods of the distributed team both when they meet virtually and when team members are physically at the site.

Visiting team members are co-located with the local team in the same space, and will pick up useful information from conversations. They will feel more as part of

[1] "Agile Software Development" by Alistair Cockburn, p. 84, Pearson Education, 2002

the same team, and get more opportunity to socialize with their teammates, building trust and respect that will help building a well-performing team.

Common team space combined with a Silent Room[2] is a good alternative to allow for some private time for team members. It can be used by several individuals according to need, but is not the permanent workspace for anyone.

— oOo —

Adding a Silent Room may solve the need for privacy. You may still choose to add single offices for a few individuals. You will have to balance the adjustment for individual personalities with the communication need. Some special offices may be placed so that they form an integral part of the workspace. Let the doors face the work area and use glass walls to support the feeling of one room.

If it is hard to convince management to do the investment, a *Pilot Team Space* solution for one team may be a way forward. With good metrics it may possible to show better productivity as a result of the modifications. With focus on risk management, one can show that the risk in running the team in a distributed way can be made smaller by being willing to invest in their environment.

With careful planning, the cost of the building modifications can be kept minimal: Remove a few (non-bearing) walls to create bigger space, and keep one small office as Silent Room, with a door in to the team. Glass walls are nice but not necessary.

Finally, the decision to do the investment in tools and a reconstruction of the building needs to be based on the possible gain in productivity of the current project and of other project groups that could gain from the new environment later. The pattern should be applied early in the lifetime of the project.

The RTGS team had 4 employees in Houston and 3 in Beijing. Some way into the project, it was decided to move the Houston team from single offices into a common area. The new team room had a private silent room, and a meeting area in the middle of the room. The space was big enough to accommodate the visiting team members. Flexible furniture was used for the meetings and by the visitors. VC equipment was not permanently installed. Since the team room was initially made for this team, they were free to design the configuration of the workspace and the actual pieces of furniture. This gave them ownership, and they could decide what they needed to support their own work methods. It was also a positive factor to get a newly decorated room with new furniture. The solution worked well for the team.

Originator: Lise B. Hvatum

Sheperded for EuroPLoP 2004

[2] Private Space in the collection “Interaction Patterns of Agile Development” by Jens Coldewey, Proceedings from EuroPLoP 2002

Join for Completion

We have on several occasions brought developers and testers together for the final tuning of the system before commercialization and delivery. Possibly the most extreme was to have the developers actually sailing on a seismic exploration vessel just before the start-up of the first commercial job with a new system. But it is efficient. Now working with geographically distributed teams, we have continued the practice.

Bring the development team together with testers on a single site for completion.

***

Your team members are distributed on engineering centers that are geographically far apart. The time difference is significant. The product you are working on is close to completion. Users are on-site with engineering to perform the final testing. There has been a close collaboration with users during the development, so you do not expect any big surprises. But you know that you will find a number of smaller problems.

How can you make the final tuning and qualification of your product efficient and focused?

Intense testing by users to qualify the system for commercial use will certainly result in a number of issues that must be corrected before you can deliver. To deliver on time is important, still it is not an option to deliver before the qualification team is ready to approve the release. The fact that your developers and testers are not co-located does inevitably slow down your rate of correction.

Some issues must be dealt with very quickly as they prohibit testing of functionality further down the workflow (hiding other problems). If the problem has to be corrected by one of the remote team members, communication problems and time zone differences will delay the correction.

When working individually with the testers, there is always a risk that we make changes that are based on the opinion of a single user. There is also a risk that developers are not focusing on the right issues to fix based on a global priority. When we are working directly with individual testers, we will often accept their issues as our highest priority. It is a heavy process to coordinate all the input to decide on the right order for corrections, and difficult for each developer to keep to a global "official list" rather than fixing what the tester tells them directly.

Although issues are recorded in a tracking system, it is too inefficient to write detailed explanations so that a remote developer can do the right modification. Still, a developer at a remote site will need these details to be able to fix the problem. The most effective way for a tester to explain the problem and the necessary corrections to the developer is usually to run the system with the developer watching. This is difficult or impossible to do when they are on different physical locations.

The remote team members will not feel the same pressure or be motivated by working side by side with the users. Still, you need the full commitment from all developers, and you need to maintain the good relations within the team.

Therefore:

Gather the whole team (developers, architects, domain experts, and internal and external testers) together in one location for the final tuning of the product before delivery.

Twice per day, have a status meeting where the corrected and new issues are presented. Use this opportunity to make sure that requested changes are urgent and agreed, and make the users consolidate the priorities of what is most important to correct.

You have increased the chance of delivering a high quality product on time. The team can work much more efficiently when being together. Communication is face-to-face and there are no time zones or other obstacles to deal with. During the final tuning of the product, this can be the best way to optimize the resources involved.

If possible, place all together in one room. This can possibly be a *Prepared Work-space* that is created for the occasion. It can even be off-site given that the testing can be done in a proper way, that there is good connectivity to the engineering site etc.

By bringing both testers and developers together frequently, it is easier to coordinate priorities and make all aware of the ongoing modifications, and of the current problems. The risk of tuning the system to individuals is lower, as the testers are able to discuss with each other on tuning of the functionality.

With the testers and the developers at on site, you can reduce the formality and level of detail in tracking defects. Make use of the possibility to team up a user and a developer so that they can do the smaller corrections on the fly as they are detected (“pair bug fixing”). This can be very efficient.

The tuning session could potentially be negative for the team feeling if done at one of the locations. By bringing the team together, it turns into a new opportunity for jelling the team. This time it even brings a possibility to bond with the user community.

The team created from combining users and developers can give engineering very useful contacts in the user community for later. The users will feel ownership of the product, and may be a good help in introducing the new system in a positive way to their peers. Hopefully they also left the activity with a better understanding of engineering and product development challenges.

The financial side will always be an issue for engineering. The best approach is usually to start small, and use the experience from pilot activities to help in the decision making with management. After very positive experience with using this pattern for one or two projects, it is easier to establish the practice.

The result for the users that participate in the testing activity is that they get trained on the new system, and if done with care it can be possible to define the event as a training or beta user training and get financial help from your training organization.

As for most of our practices, there is a certain cost and effort involved. You will need dedicated space and possibly have to bring in extra equipment/systems to do the testing at one location. There are expenses associated with getting the team members and the testers on-site. This cost is low compared with what the users most probably will gain in improved quality, but it is hard to measure and prove that this is the case.

In Schlumberger, this pattern was applied on two different teams in 2003. Testers came in from several field locations, and we had an empty lab that was converted into a testing room. It had space for several work areas where a tester and a developer could sit together, plus a big meeting room table where status meetings were done twice daily. Systems were borrowed from the on-site training center. They also helped out in providing field personnel to take part in the testing. For both projects, the final tuning was very successful, and the systems were delivered on time.

Originator: Lise B. Hvatum

Sheperded for EuroPLoP 2004