

XP Expanded: Patterns for Distributed eXtreme Programming

Keith Braithwaite

`keith.braithwaite@wdsglobal.com`

Tim Joyce

`tim.joyce@wdsglobal.com`

WDS Global, Forelle House, Marshes End, Upton Road, Poole

June 13, 2005

Abstract

The ever-increasing globalisation of businesses that consume development effort leads to the desire to create development organisations that span the world. At the same time, XP and other Agile approaches to development emphasise the importance of close communication and collaboration. While these two forces on development teams seem to be in flat contradiction, in fact a body of techniques for successful distributed Agile development is beginning to emerge. These few patterns record those facets of one successful distributed XP team's practice that seem to be widely shared amongst distributed development efforts with an Agile bent.

The patterns are: Kickoff, Visits Build Trust, Virtual Shared Location, Remote Pair, Multiple Communication Modes

Introduction

We distinguish three styles of cross-site development team organisation using Agile ideas: Distributed XP¹, Agile Dispersed Development and Agile Outsourcing.

What this paper is *not* about

Agile Dispersed Development

Individual developers in dispersed locations work on a single project. There is a degree of overlap with DXP (one ADD project is cited here), but this paper is primarily about something else.

¹Different from the process of that name in [10]

Agile Outsourcing

At first sight, this seems to be an oxymoron. Some organisations do apply agile techniques to make outsourcing as successful as it can be (some are cited here), but this paper is primarily about something else.

And, what it is about**Global Business, Global Team**

There is more than one market in the world. These markets have some similar, and some differing needs. Business has much to gain by locating the developers meeting a market's specific needs in that market, then sharing results between markets as appropriate.

Distributed XP

This paper began as an exploration of how to maintain the best qualities of XP in a development organisation distributed across regions, and how to use XP to support a business so distributed. We have found that some of the practices we use to achieve this have been reported elsewhere, and may form the basis for a future pattern language for distributed XP.

Summary of the Patterns

Kickoff *A co-located meeting where a project can begin with a shared understanding amongst, and trust between, team members*

Visits Build Trust *Periodic visits by team members based in remote locations preserve the trust relationships required for a distributed team to maintain healthy relations*

Ambassador *Have a local expert in remote conditions to resolve misunderstandings in either direction*

Multiple Communication Modes *Compensate for the shortfall in communication between team members caused by distance by affording as many different, simultaneous communications channels as possible*

Virtual Shared Location *When team members cannot share a physical location simple but effective collaboration software creates a virtual shared location*

Remote Pair *Team members in different locations to work on problems together, sharing knowledge, sharing experience, remaining close colleagues when far apart*

These patterns flow together.

When beginning a distributed agile project, have a Kickoff. Maintain the good working relations established there using Visits Build Trust with some long-term Ambassador visits. Lower the barriers to effective collaboration between site by providing Multiple Communication Modes (of which a particularly valuable one is a Virtual Shared Location), and use those channels to build a sense of one team of peers through Remote Pair.

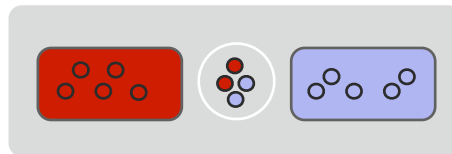
There are other practices [4] aimed explicitly at remaining “eXtreme” while working across locations, but which are not yet widely enough practised (or described) to constitute patterns. Doubtless, as distributed agile development becomes more widespread the catalogue of patterns will expand. We look forward to being part of that exercise.

Acknowledgements

We would like to thank our boss, Benjamin Marais, for his support, our colleagues in the world-wide WDS Global development team for inspiring the collection of these patterns, and our shepherd Kevlin Henney for his many helpful comments and suggestions.

Kickoff

A co-located meeting where a project can begin with a shared understanding amongst, and trust between, team members.



A new project is to start and will be executed by a single distributed team. Co-located XP teams share energy and goals by practicing **Sit Together** all the time, but a distributed team is dispersed most of the time.

Team members must always begin work in a spirit of mutual trust so that energies can be focused on solving and innovating. They must synchronise their ideas about the project in order to establish common goals that everyone can measure their progress against, but in a distributed team they will execute the project apart from one another.

Humans find it hard to trust others that they have not physically met. When challenges appear in the work of a distributed team it is too easy to blame and criticise the remote “others”, rather than to collaborate with them to overcome the challenge.

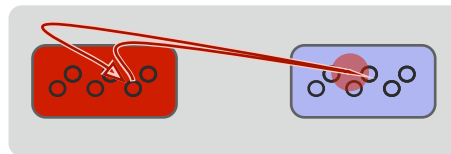
Therefore: Bring everyone involved in the project together in one place at the same time for a kick-off meeting. Lay the foundations of the technical work on the project here, but also allow time and space for team members to get to know one another.

Future distributed working is informed by a cohesive view of the project and secure interpersonal relationships formed while the advantages of **Sit Together** (a core practice of XP [1]) were available. However, when humans cannot meet one another frequently, trust between them decays, and so the relationships built during the Kickoff must be actively maintained once the team members disperse. To do this, use **Visits Build Trust** and follow on by sending an **Ambassador**.

As seen in: [5], [9], [12]

Visits Build Trust

Periodic visits by team members based in remote locations preserve the trust relationships required for a distributed team to maintain healthy relations.



Without proximity, trust decays over time. No matter how successful the Kickoff meeting has been, team members find it hard to retain faith in the good intentions of remote colleagues whom they have not met recently. Under stress, blameworthy can replace collaboration and fingerpointing can replace problem solving. This would lead to a breakdown of team operations across sites.

When the team can no longer collaborate within itself across sites, local customers lose the benefit of the remote team members. No matter how effective the **Multiple Communication Modes** deployed to aid communication between team members nothing is as good as face-to-face communication. Small misunderstandings can become severe breakdowns when they occur over poor communications media, especially in the face of language differences.

Therefore: Have team members rotate through locations continually. Always have at least one team member working away from their home location. Eventually, every team member should have had experience of working in at least two locations. This can be an expensive operation, and the damage done to team relations during a period when there are no visitors may not be immediately apparent, making the cost hard to justify.

Trust in a team member currently remote is more likely to be maintained with the memory of having worked with them locally in the past. Contrariwise, trust in remote colleagues is more likely to survive with the memory of having worked with them.

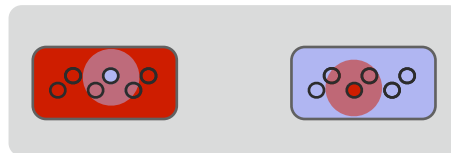
Having experienced a colleague's communication style directly, in their local environment, it's much easier to work with them subsequently over a (possibly quite low-bandwidth) medium.

Only so much can be achieved with short visits. Deep trust relationships can only be built over time, so use **Ambassador** to maintain longer-term relationships at the team level.

As seen in: [5], [12], [9]

Ambassador

Have a local expert in remote conditions to resolve misunderstandings in either direction.



Members of a team in one location find it hard to understand the point of view of members in another location. This can be due to local technical constraints, market conditions or cultural and language differences.

Changes to tools or practice developed for the particular conditions at one site can seem threatening to other sites when the spread through the codebase. Inevitable differences in infrastructure can dramatically change the implications of technical decisions across sites. Decisions made in one site can simply seem upsettingly strange and arbitrary in the context of another site. Attempts to explain and explore such issues fail due to a missing shared understanding of decision-making context over shared technical resources (the codebase of the system).

When misunderstandings caused by lack of knowledge of local conditions escalate, the trust vital for cooperation between sites breaks down.

Therefore: Send an ambassador from one location to another, for an extended period. While Visits Build Trust, longer visits maintain more trust for longer.

Such a local representative can interpret the communications and actions of the remote group, using their experience of the remote site's peculiarities, and so ensure good relations between sites over the longer term. They also can reflect back to their home site what they have learned about local conditions, to help foster understanding between sites.

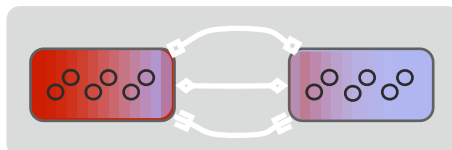
An **Ambassador** also serves as a continual demonstration that “they” are just like “us”, and can influence locally on behalf of the remote group when required. Ambassadors can also carry business domain knowledge between sites. The role requires an individual who is a committed multiculturalist and internationalist, is free to work away from home for long periods and has great social skills, as well as being able to make a contribution to the technical work of the site hosting them.

Fewer, longer-term visits may be easier to justify financially than many, shorter ones, allowing **Ambassador** to be balanced against **Visits Build Trust**.

As seen in: [9], [12], [8]

Multiple Communication Modes

Compensate for the shortfall in communication between team members caused by distance by affording as many different, simultaneous communications channels as possible.



The members of a team cannot be colocated, but must collaborate freely and closely. All development relies on tacit knowledge, but Agile development places a “deliberate reliance” upon it. But in a distributed team [6] face-to-face communication (explicit and “overhearing”) cannot be used to maintain tacit knowledge, trust and shared understanding. During cross-site collaboration episodes many different kinds of knowledge must be shared, often during sharply time-limited handover slots.

Therefore: Provide team members with as many communication media as possible. At least these: individual and conference telephone, teleconference, video conference, email, IM, **Virtual Shared Location**, VNC.

Communication is fostered greatly, and many different modes of communication can be applied in parallel. An **Ambassador** can guide the choice of communication media so that it most appropriate for the target audience.

A good conversation to hear at a videoconference standup meeting would be: *Site 1: We had an idea for that problem, I've just jabbered you the URL for the wiki page that discusses our example code, see what you think. Site 2: Great! Let's Remote Pair on this tomorrow.*

As seen in: [9], [8], [5]

Virtual Shared Location

When team members cannot share a physical location simple but effective collaboration software creates a virtual shared location.



Team members can meet neither at the same time nor at the same place, but must share informal and formal ideas and information.

Technical and domain information is often non-volatile and detailed. This knowledge is critical to the team forming a cohesive view of the problem space, and so must be shared. However, the information arrives ad-hoc, and its value may not be immediately obvious. Remote team members may not be available for immediate communication. Instead, team members need to ‘pull’ this information from the collective memory whenever they need to use it.

Team members need to share personal information and interests in order build trust, but it is often intrusive to ‘push’ informal information to other team members who are not well known to each other .

Therefore: Use a wiki.

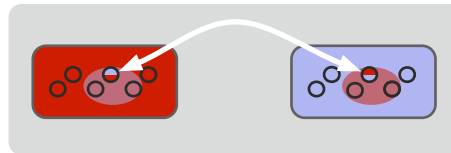
A shared virtual place is created where notices may be posted, asynchronous conversations take place in a persistent form, and a feeling of community fostered. Wikis are preferred over more complicated Computer Supported Collaborative Work tools (such as that described in [13]) because of their ease of deployment, ease of use and “just structured enough” nature.

A team wiki will not only be a virtual meeting place for team members but will also act as the memory of the team, both as a store of information but also as a record of the team’s learning processes.

As seen in: Many sources mention team wiki’s (although the notion of wiki as shared virtual location is not often made explicit as they don’t realise that this is what they are doing). The walls of public lavatories.

Remote Pair

Team members in different locations to work on problems together, sharing knowledge, sharing experience, remaining close colleagues when far apart.



Informal conversations with remote team mates tend to be unsustainable because common cultural references do not exist.

The codebase might have as much as 16 hours worth of changes applied before team members in that location see it again. In a team with collective code ownership (such as an XP team) the code you (and your pair) wrote today might have 16 hours of made changes to it before you see it again.

Team members in one location need to be confident that changes made elsewhere will be of the same quality as those made locally (and made with aligned intent).

Therefore: Adapt Pair Programming to the distributed environment². Establish an easy-to-start environment with rich communication (video, text and sound) and a shared development tool (use VNC or similar to share an IDE). Have two developers in each of two locations work together on some problem for an hour or two as timezones allow.

By providing the code as a common artifact to interact around, team members will discover other shared contexts that stimulate informal discussions, and therefore build trust. Even if remote pairing were possible all day, it would be an inefficient way to code—however the goal is not so much to write code as to build trust and share understanding. Code-level decisions and communication will occur between remote team members in a way as uniform as possible with that used for local pairing, fostering the same sort of trust between team members as is possible through pairing with **Sit Together**.

Remote pairing needs to happen frequently, in order to maintain equity of technical expertise and knowledge between sites and as another mechanism to foster trust between sites. With a tool set ready to go, the default question should be “why not do this in a remote pairing session?”, rather than the opposite

As seen in: [13], many informal mentions on newsgroups, etc. The ADD team in [7] provide new joiners to the organisation with a pre-configured common hardware stack to facilitate this and other practices

²Colocated Pair Programming is discussed at length in [1]

References

- [1] Beck, K. with Andres, C.: *Extreme Programming Explained: Embrace Change* (2nd Edition) Addison–Wesley (2004)
- [2] Beck, K. *et al*: *The Agile Manifesto*
<http://www.agilemanifesto.org/>
- [3] Beedle, M., Schwaber, K.: *Agile Software Development with Scrum* Prentice Hall (2002)
- [4] Braithwaite, K., Joyce, T. *Truly Agile Development in a Distributed Environment: Knobs to 11* Submitted for XP2005
- [5] Carmel, E.: *Global Software Teams* Prentice Hall (1999)
- [6] Cockburn, A. *Core Elements of would-be Agile* The Cutter Edge Cutter Consortium (2002)
- [7] Daniels, J., Dyson, P.: *CS2 Dispersed Development* OT2004
<http://www.spa2005.org/ot2004/programme.shtml> (2004)
- [8] Jensen, B., Zilmer, A.: *Cross-Continent Development Using Scrum and XP* <http://www.informatik.uni-trier.de/ley/db/conf/xpu/xp2003.html> (2003)
- [9] Fowler, M.: *Using an Agile Software Process with Offshore Development* <http://martinfowler.com/articles/agileOffshore.html> (as at April 2004)
- [10] Kircher, M., Jain, P., Corsaro, A., Levine, D.: *Distributed eXtreme Programming*
- [11] Martin, A., Biddle, R., Noble, J.: *When XP Met Outsourcing* XP 2004, LNCS 3092 (2004)
- [12] Poole, C. J.: *Distributed Product Development using Extreme Programming* XP 2004, LNCS 3092 (2004)
- [13] Reeves, M., Zhu, J.: *Moomba—A Collaborative Environment for Supporting Distributed Extreme Programming in Global Software Development*. XP 2004, LNCS 3092 (2004)
- [14] Rogers, R. O.: *Scaling Continuous Integration* XP 2004, LNCS 3092 (2004)
- [15] Simons, M.: *Internationally Agile* Informit.com (2002)
<http://www.informit.com/articles/article.asp?p=25929>