Implementing Distributed Extreme Programming: A Case Study

by Monica Yap
Implementing Distributed Extreme Programming: A Case Study
by Monica Yap

Introduction
In early 2004, WDSGlobal brought together three independent development regions — the United States (Seattle), United Kingdom (Poole), and China (Singapore) — to form an around-the-clock Extreme Programming team. I was brought in as the Extreme Programming coach for the U.S. in early 2004. This paper describes the challenges the team faced in this environment, lessons learned, and how issues such as global continuous integration, cultural differences, and conflicting priorities were resolved across regions.

Background
WDSGlobal is the leading global provider of knowledge-based services to mobile operators, manufacturers, application providers, and sales channels. Delivered via the Integrated Service Framework and based on ten years of experience in the mobile date marketplace, WDSGlobal services assist in every stage of the mobile product lifecycle from pre-launch problem mitigation to specialist end-user customer support. Headquartered in Poole, Dorset, WDSGlobal has over 350 employees serving over eighty companies worldwide.

Much of the work undertaken in the development environment is devoted to the refinement of WDSGlobal’s market leading mobile configuration platform, which is used on the Nokia, Sony Ericsson, Siemens, and Vodafone Web sites. As the mobile industry advances and innovates frequently, the demand for WDSGlobal services has reached an unprecedented level, heavily tasking the development teams.

Motivation and Goal - One Global Team
To meet customers’ needs across the globe, the company created three regions: the Americas, Europe, and Asia. The business quickly discovered that the regions have some common and some differing needs. The company has much to gain by locating the developers to meet a market’s specific requirements in that region, and then sharing results and knowledge among regions as appropriate.

Until the end of 2003, the core of the Web-based tools were developed and deployed as Application Programming Interfaces (APIs) by the UK team using Extreme Programming and Java. In each region a non-Extreme Programming team would use these internal APIs to deliver to localized sets of requirements using non-Java technologies. This structure and method created a lot of unnecessary duplication in the Web sites, and required costly maintenance for the different technologies employed in each region.
At the beginning of 2004, the idea to combine all the development teams into one global team using the same process and technology, Extreme Programming and Java, was proposed. This global team was also to share the same code base to minimize duplication and reduce maintenance costs.

**Obstacles**

Having a development team in each of the three locations presented a number of challenges. One of these challenges was that the teams did not previously have frequent communication with each other. There were also cultural differences, time zones with little overlap during the workday, different technical backgrounds, and differing technical platforms, all of which had prevented the regions from operating as one team previously.

**Cultural Differences**

Though the members of all regions speak English, there is much more to a language than how its words are pronounced. Many misunderstandings arose because the groups did not really share a higher-level common language. Adding to these problems were a number of issues caused by unrecognized cultural differences. For example, one culture has more emphasis on self-sufficiency, so they tend not to ask for help when problems arise. Another culture would not offer their help unless they were asked, while the third felt that presenting the problem was an invitation for their team to jump in and help. The cultural differences often led to misunderstandings and conflicts between regions.

**Time Zone**

There was a minimum of overlapping time between the three regions’ working hours, so a fair amount of compromising took place to enable members in different regions to communicate with each other outside of their normal working hours. This impacted family commitments and personal schedules.

**Technical Background**

The UK members came from an object-oriented programming background, had done some Extreme Programming, and placed more emphasis on doing things in an academically correct fashion. The other team members had no knowledge of the Extreme Programming process or object-oriented programming, and instead focused on getting work done quickly, learning new tools and techniques as needed. In addition, the programming styles among the regions were widely different. For example, the U.S. and Singapore developers had no experience with unit testing or refactoring prior to the move to Extreme Programming.

**Technical Environment**

Before the virtual global team transition, all servers were hosted in the UK office. There was no shared source control system, and deployment was done independently for each Web site. Each region had its own independent build and test servers, source control system, development environment, and technical platforms.
Transition Phase

During the transition to a virtual global team, WDSGlobal faced a number of technical, ownership, and training-related challenges.

Challenges

The source control system (CVS) used by the original UK team was not feasible for remote use by other regions. It took forty minutes to synchronize changes from a remote region. A new source control system was chosen (Perforce), and local proxies had to be put in place to deal with the performance problems. The UK build machine was not sufficient for all regions, so two more build machines were created. In the U.S. and Singapore regions, there were no pairing machines available, so new pairing machines and desks were purchased and set up to the same specification as those in the UK. Both of these regions also had to reconfigure their office space to accommodate the pairing stations and cube-free structure.

Shared Ownership

The UK team had developed the original core backend portion of the system and found it difficult to share ownership with other regions. While some of the U.S. and Singapore developers welcomed the new process and technologies, others were overwhelmed by the large shared code base and the new Extreme Programming practices (such as pair programming). A few who could not adapt left the company.

What Worked

Regional Coaches

Since the non-Extreme Programming regions did not have any prior experience in Agile methodologies or object-oriented programming (OOP), an Extreme Programming coach was hired in each region, reporting directly to the CTO. The coaches helped the local team learn Extreme Programming practices, and trained them on object-oriented programming with Java. Additionally, the coaches also helped local business units to adjust to the new development process, where a different kind of resistance arose (See the Outside Forces section). Coaches evolved to an important role. Over time, the coaches recruited more team members with Agile and OOP experience in the region, and weekly coach meetings yielded consistency and helped bring the teams together.

Boot Camp

Team members must always work in a spirit of mutual trust so their energies can be focused on solving problems and producing innovative solutions. Many people find it hard to trust those they have not met in person. To overcome this challenge, the non-UK region developers spent several weeks with the original Extreme Programming team in the UK office. The developers thus had a chance to work side by side and build initial trust relationships and common work practices.

Rotating Guru

Due to the complexity of the code base and unfamiliarity of the infrastructure in the non-UK regions, a senior team member from the original Extreme Programming team
was sent to the non-UK regions to help with setting up the infrastructure and providing initial training and mentoring. This also helped the UK developers to understand the difficulties the other teams faced in developing remotely from the UK servers—such as dealing with long delays during check-outs and check-ins.

Lessons Learned

Preparing the Business

Although WDSGlobal trained the developers in Extreme Programming methodologies, the company didn’t initially educate the business in each region adequately to prepare for this dramatic process change. The educator skills needed to train developers and business units are very different, and WDSGlobal did not have the necessary resources (see the Outside Forces section.)

Big Switchover

In April 2004, shortly after the boot camp, all development teams became one virtual team, using Extreme Programming as the development process and Java as the technical platform. The non-UK regions had no prior exposure to Extreme Programming and OOP, and had the most trouble during the switchover. Some were not able to make the transition. More training in Extreme Programming and OOP was needed prior to the switchover. Also, there should have been a more gradual transition. A better approach would have been to give the non-UK regional developers formal training on Java and OOP, then pair them with Java developers for three months, and then teach them Extreme Programming practices.

Immature Infrastructure

The company’s global infrastructure was not adequate to handle the new distributed development environment. The network bandwidth between the regions was too limited to allow large amounts of data and traffic throughout the network. The developers in the U.S. and Singapore regions constantly struggled with long delays during file copies and continuous integration builds across servers located in different regions. This has been gradually resolved with increased network bandwidth and with work-arounds, such as moving to Perforce and its caching proxies as the source control system.

Continuous Distributed Development

The success of an Extreme Programming team relies on a shared set of values and principles, such as coding standards, test-driven development, and refactoring. However, the specifics of these principles and values can be interpreted differently by each individual. In a non-distributed Extreme Programming team, common ground is achieved through individual adjustments and continuous communication among team members. In a distributed team environment, team members are physically located in different countries and time zones, come from different cultural backgrounds, and do not enjoy constant communication. When challenges appear, it is often easy to blame and criticize the remote groups and the team breaks down into competing tribes [2]. The global team struggled to maintain a common process vision and values. To share the same code base and system, everyone had to trust each other as equal team members. Without proximity, the trust relationship decayed over time.
Also, as business needs evolve, the current technologies and architectures may no longer satisfy these needs. In a non-distributed Extreme Programming team, when new ideas and technologies are introduced, the team comes to an agreement before the team proceeds. With the distributed, yet virtually combined team, how does one introduce new technologies or architectural changes and get buy-in from the whole group?

What Worked

Daily Handovers
One of the most important Extreme Programming values is communication among all team members, and this is even more critical for a distributed team. With the three separate regions, daily handovers occur when one workday ends and another begins, similar to relay runners passing the baton. At 9 a.m. Seattle time, the UK region hands over the system to the U.S. The U.S. region hands over to Singapore at 5 p.m. Seattle time, and the Singapore region then hands over to UK at the end of their workday. At first the daily handovers were primarily daily status updates. The objectives of the handovers gradually shifted toward teaching each other what each region had learned in their last working day, collaborating on design ideas, and raising cross-regional issues.

Face-to-face Communication
Face-to-face communication is much more effective than e-mail or phone conversations. Video conferencing tools were used as much as possible, including during each of the daily handovers and for customer requirement gathering sessions.

Shared Common Environment
In the distributed team, as in a traditional Extreme Programming team, all pairing machines are set up the same to help minimize configuration and adjustment time when development pairs rotate to different machines. The configurations are checked into the shared source control system, so all developers receive configuration changes automatically when they synchronize with the system at the start of each workday.

Remote Pairing
The tight interaction within a non-distributed Extreme Programming team — not just through pairing session experience, but also through informal discussions at coffee breaks or lunches — often allows the team to form a group mind. When a complex idea or design is being discussed, each team member can understand the others much better from within this shared context, based on prior knowledge and experience. This is much more difficult to achieve in a distributed team. To help create a group mind across regions, frequent remote pairing sessions were introduced, where two developers from different regions would pair using virtual network computing (VNC), teleconferencing, and the integrated development environment (IDE). This often helped improve collaboration and sharing, avoid misunderstandings, and foster shared memory between the pairs and, through them, the teams.

Round-the-World Program
The markets in each region share some common and some different circumstances, and decisions made by a team within a region are often based on the specific local market conditions. These decisions may not be understood by the other regions. If misunderstandings are not addressed, the trust relationship across regions eventually breaks down, and collaboration is no longer possible. To combat this problem the round-the-world program was introduced, which rotates members to another region for several weeks. Longer visits of four to six weeks are most effective, though due to personal constraints (such as family or visa issues) or funding limitations, this may not always be feasible.

**Shared Stories**
For stories that applied to all markets across regions, the regions were encouraged to work together in an around-the-clock manner. The conversation and collaboration that ensued helped build common values and mutual trust, strengthening the relationships between the regions.

**Putting Out Fires**
Similar to sharing related stories, if an emergency production issue could not be addressed during one region’s working hours, the work was handed over to the next region until the fix was complete. This approach forced the regions to communicate detailed designs and strengthened the trust level.

**Coach-to-Coach-Level Communication**
The team found it was essential for the coaches to maintain a constant level of communication and synchronization. Coaches talked to each other one-on-one after the daily handovers, and all coaches met weekly via teleconference to ensure any general or cross-regional issues were addressed and resolved. Every quarter the coaches met in person for two weeks to address more complex issues and conduct long-term planning for the quarter.

**Extreme Programming Principles and Practices**
The Extreme Programming principles and practices help the distributed team to maintain a certain level of common values and knowledge. Extreme Programming encourages open communication and collaboration, and thus increases communication across all regions. The Extreme Programming core principles, such as refactoring and test-driven development (TDD), allow us to share a common language. However, the subtle meanings of these principles and practices may not always be the same across regions, so regular discussion is needed to maintain a common understanding.

**Lessons Learned**

**Balancing the Teams**
In the distributed team environment, the region with the most developers always lead the architecture and design in a particular direction, and the other regions were forced to follow, hindering cooperation and the one-team goal. It was important for each region to have the same skill level and number of developers.
Introducing Process Changes
To allow each team member to focus on their daily tasks, all team members must know exactly what to expect from each other. Having a set of rules and predefined processes is the only way to set these expectations [3]. The team learned that when one region introduced high-level process changes without talking to the other regions, it created confusion and frustration. If this occurred often and was left unresolved, the collaboration and trust relationship between regions suffered a breakdown. The team decided no major process change could be introduced without prior agreement from all the coaches.

Introducing Innovation
The most difficult part of employing distributed teams was introducing architecture changes or new technologies needed to satisfy customer requirements. In most cases this would be done by one region and then brought into the code base, where it had to be supported by all regions. Without a team consensus, the regions polarized, making integration of the change difficult. But to gain buy-in from the whole team required a time-consuming feedback and discussion period. Unfortunately, to this day the team still has no perfect solution for this problem.

Allowing Process Flexibility
Due to the different cultures and markets, each region needs a limited amount of process flexibility. There should be a balance between the need for a global process and local adaptation.

Outside Forces
Until the teams combined into one global team, they were managed by their local offices and were treated as local resources. Most of them did not actually have a formal development process and were developing software in an ad-hoc fashion.

After combining the teams, regional offices were forced to engage in the planning game and global story prioritization. This initially created a chaotic environment for the local business, since there needed to be an uninterrupted weekly plan. With the development team employing small incremental steps, the company gradually developed a process for building a consolidated monthly product backlog, known as the Company Program Plan.

Including the Business
Each week, the team asked the business managers to come to the planning game. Once stories were prioritized, business managers were asked to leave the development team alone for the rest of the week. Not changing the team’s goals for a whole week was the most difficult step for the business and took the longest for them to accept. The business was used to making changes on a daily basis, and pushing them to think a week ahead fostered better strategic planning. During the iteration, the coaches worked with the business managers to plan and prioritize new stories based on upcoming business needs.
Calculating Value

Where previously the loudest voice would receive the most resource allocation and get the most work done, suddenly the business needs with the highest global value received the most resources. To prioritize effectively, the business needed to understand true business value and cost for each story—in other words, what is the story worth to the company as a whole? The team introduced a common unit of measurement across all regions, called the value unit (VU). For each story, the value unit is calculated with the following formula:

\[ VU = \frac{\text{perceived revenue} \times \text{long-term customer potential}}{\text{story estimation}} \]

*Perceived revenue* is the immediate revenue WDSGlobal stands to gain from this project. *Long-term customer potential* is how likely the company is to receive more projects from this customer on a regular basis.

With this unit of measurement, the company can make priority decisions across projects globally. These stories are then added to the global product backlog in order of priority. Combining prioritized stories into one product backlog allowed the regional development teams to provide the highest global business value. This was a vast improvement over regionally optimized and managed development teams who had no insight into what was best for the company as a whole.

Incorporating QA

Previously there were no automated user tests, and in some cases the new software was released without customer signoff. Quality assurance is an essential part of the Extreme Programming process, and without it quality suffers. So the team adopted a user acceptance testing framework (Selenium by ThoughtWorks) and began to train a non-technical business team to work with customers to automate user acceptance tests. This effort is still in the startup phase, but the team is already seeing positive results.

Conclusion

After a year with the globally distributed Extreme Programming team, the company considers the transition to be successful. The team has demonstrated that Extreme Programming works for a globally dispersed group performing around-the-clock development with a shared codebase. Distributed Extreme Programming provides the company with the key benefits of an Agile process: the developers have face-to-face contact with regional customers, and the business is able to quickly adapt to market changes while producing quality products. While there are many opportunities for improvement in the current process, the team has established a firm foundation.
About the Author
Monica Yap is an Engagement Manager and Agile Coach for SolutionsIQ, focusing on building and leading successful Agile teams for outsource projects. Monica has over sixteen years of development experience in e-commerce, wireless, and various other industries, as well as over eight years of experience leading agile teams, providing quality products through the use of continuous refactoring, unit testing, pair programming, small releases, and an evolving architecture. She has worked with distributed Agile teams over four years (U.S., UK, and Singapore). She and her teams have demonstrated tremendous successes, with 90 percent of the iterations being delivered on time, meeting or exceeding customer expectations.

Monica is a regular presenter at seminars (including Agile2005, Agile2006, and Agile2007) and volunteer for technical and process groups, including the Seattle Agile user group, Seattle Java user group, and Seattle APLN.

References


About SolutionsIQ
SolutionsIQ offers a full spectrum of services to develop software and fulfill technical talent needs, while improving your Agile knowledge and capabilities. Clients include AT&T (Cingular), Amazon, Corbis, Expedia, Federal Home Loan Bank, InfoSpace, Key Bank, Microsoft, Nordstrom, Safeco, and US Bank. A Microsoft Gold Certified Partner, SolutionsIQ is also a member of Scrum Alliance, Agile Alliance, and Washington Technology Industry Association. Learn more at www.SolutionsIQ.com.