

Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams

Jeff Sutherland, Ph.D.
Scrum, Inc.
 Boston, MA, US
jeff.sutherland@scruminc.com

Guido Schoonheim
Xebia b.v.
 Hilversum, Netherlands
gschoonheim@xebia.com

Maurits Rijk
Xebia b.v.
 Hilversum, Netherlands
mrijk@xebia.com

Abstract

Scrum was designed for hyperproductive teams where productivity increases by 5-10 times over industry averages and many colocated teams have achieved this effect. The question for this paper is whether distributed, offshored teams can consistently achieve the same level of performance. In particular, can a team establish a localized velocity and quality and then maintain or increase that velocity and quality when distributing teams across continents. Since 2006, Xebia (Netherlands) started localized projects with half Dutch and half Indian team members. After establishing localized hyperproductivity, they move the Indian members of the team to India and show the same velocity with fully distributed teams. After running XP engineering practices inside many distributed Scrum projects, Xebia has systematically productized a model for high performance, distributed, offshore teams with one of the lowest defect rates in the industry.

1. Introduction

The advantages of colocated teams, doubling developer productivity compared to non-colocated teams, are commonly eliminated by distributed software development and offshoring [1]. This paper introduces a model for producing distributed and offshored team velocity that is equal to colocated velocity of a single team. The model is repeatable, proven across many projects, and is recommended for teams that can execute a high performance Scrum implementation [2] with XP engineering practices inside [3].

Agile project management with Scrum derives from best business practices in companies like Fuji-Xerox, Honda, Canon, and Toyota [4]. Combining Scrum with XP engineering practices has generated hyperproductive teams with 5-10 times industry average performance since 1993 [5] [6]. In 2005, two Agile companies, SirsiDynix (U.S.) and Exigen

Services (Russia), used distributed Scrum teams to deliver linearly scalable performance for a large project of over 1M lines of code [7]. A distributed team of over 50 people in the U.S. and Russia delivered velocity per developer equivalent to a 6 person hyperproductive, colocated Scrum team. This 50 person team produced the same number of features as a typical 350 person waterfall team [8].

During 2006-2008, Xebia implemented a distributed software development team model on multiple projects of variable types with teams half in the Netherlands and half in India. These distributed teams used the Scrum process with XP engineering practices inside. When replicated over multiple projects, the Xebia implementation shows distributed velocity equal to local velocity and has created a new recommended standard for deploying distributed teams across geographies.

Here we describe offshoring strategies for overcoming the typical geographic, language, and cultural barriers that impede distributed development. Traditional outsourcing failures can be avoided with the approach described here. Distribution of individual Scrum teams across geographies eliminates communication failures, XP practices solve integration problems, and daily team meetings maintain high focus on customer priorities. Earlier work in other companies showed that colocation doubled Agile team productivity [1]. The fully distributed model supports geographically transparent software development projects where performance consistently meets or exceeds productivity of colocated Agile teams.

2. Benefits and challenges in outsourcing offshore

U.S., European, or Japanese companies often outsource software development to Eastern Europe, Russia, or the Far East. The three key advantages that offshoring strives to achieve are (1) lower costs of labor, (2) capture talent not available locally, and (3) increase and decrease project size without layoffs.

Each of these advantages comes with its own challenges that have to be solved to make outsourcing successful.

2.1. Lower cost of labor

Typically, remote teams operate independently and communication problems lower productivity. Most offshoring organizations require detailed specifications before they begin a project and these traditional project planning methodologies show high failure rates.

The hidden costs of offshoring are significant, beginning with startup costs. Barthelemy [8] surveyed 50 companies and found that 14% of outsourcing to offshore operations were failures. In the remainder, costs of transitioning to a new vendor often canceled out anticipated savings from low labor costs. The average time from evaluating offshoring to beginning of vendor performance was 18 months for small projects. As a result, the MIT Sloan Management Review advises readers not to outsource critical IT functions offshore.

Secondly, high productivity is not easily achievable. IDX Systems (now GE Healthcare) averaged 240% productivity improvement with Scrum during 1996-2000 based on analysis by external function point experts. Outsourcing development to an offshored waterfall team typically saves 20% over internal waterfall costs. At IDX, it would cost twice as much to get a project done offshore compared to internal Scrum teams. At PatientKeeper (a MIT startup company in 2000) during 2004-2007, the break even point for outsourcing was achieved only when Indian developers cost less than 10% of American developers. Because Indian waterfall developers cost 30% of Boston Scrum developers, it cost three times as much to get software developed by an Indian waterfall team. The PatientKeeper Board permanently terminated outsourcing after reviewing these ROI data.

2.2. Capture talent

Capturing external talent may also be a problem. Jack Blount, CEO of Dynix and former COO of Borland decided not to outsource to India and China after he verified that annual turnover rates were 30-50% [9]. Rathi describes how employee turnover is largely determined by two variables: person-culture and person-job fit [10]. Lack of these fits results in lower job satisfaction and consequently in employee turnover.

2.3. Scale project size without layoffs

Scaling with remote capacity gives you a local team that remains stable when you scale down. However if core development is moved offshore, knowledge gets lost when you downsize, causing severe problems and sometimes vendor lock-in.

3. Distributed Scrum team models

Achieving promised benefits of outsourcing requires real cost savings, stable offshore teams, and a strategy for retaining core knowledge onshore. This can be achieved with stable offshore Agile teams that can maintain the same velocity as onshore teams and with onshore teams that maintain the same knowledge level as offshore teams.

Here we consider three distributed Scrum models commonly observed in practice.

Isolated Scrums - Teams are isolated across geographies.

Distributed Scrum of Scrums – Scrum teams are isolated across geographies and integrated by a Scrum of Scrums that meets regularly across geographies.

Fully distributed Scrums – Scrum teams are cross-functional with members distributed across geographies. This means that a single team will have members in multiple locations. A single team might have a ScrumMaster and two developers in the Netherlands while a tester and four developers reside in India. These team members share a single sprint backlog and share code ownership. In the SirsiDynix case, the Scrum of Scrums was localized with all ScrumMasters in Utah. At Xebia, ScrumMasters may be in the Netherlands or India depending on project needs.

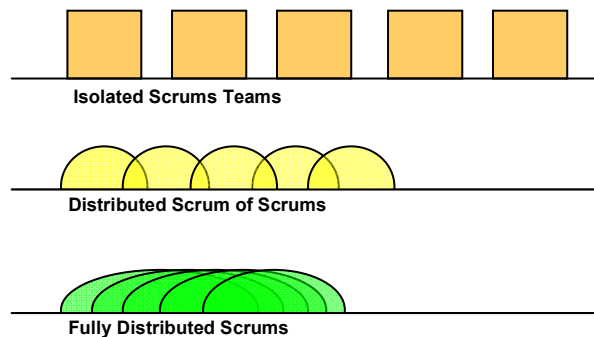


Figure 1: Distributed Scrum Team Strategies

Most offshore development efforts use a degenerative form of the Isolated Scrums model where outsourced teams are not cross-functional and not Agile. Requirements may be created in the U.S. and

developed in Dubai, or development may occur in Germany and quality assurance in India. Typically, cross-cultural communication problems are compounded by differences in work style in the primary organization vs. the offshored group. In the worst case, teams outsourced this way are not using Scrum and their productivity is typical of waterfall projects further delayed by cross-continent communications lag time. Implementations of Scrum in a data rich CMMI Level 5 company simultaneously running waterfall, incremental, and iterative projects, showed productivity of Scrum teams was at least double that of waterfall teams, even with CMMI Level 5 reporting overhead [11]. Outsourced teams not using Scrum will in the best case achieve less than half the velocity of a onshore site using Scrum assuming equal talent across teams.

The latest thinking in the Project Management Institute Guide to the Project Management Body of Knowledge (PMBOK) models is a degenerative case of isolated non-Scrum teams. This is a spiral waterfall methodology which layers the Unified Modeling Language (UML) and the Rational Unified Process (RUP) onto teams which are not cross-functional [12]. It partitions work across teams, creates teams with silos of expertise, and incorporates a phased approach laden with artifacts that violate the principles of lean development [13].

Best practice recommended by the Scrum Alliance is a Distributed Scrum of Scrums model. This model partitions work across cross-functional, isolated Scrum teams while eliminating most dependencies between teams. Scrum teams are linked by a Scrum-of-Scrums where ScrumMasters (team leaders/project managers) meet regularly across locations. This encourages communication, cooperation, and cross-fertilization and may be appropriate for newcomers to Agile development or those who have offshore limitations that cripple the productivity of the fully distributed model.

4. OneTeam Model

Xebia's Fully Distributed Scrum model has all teams fully distributed and each team has members at multiple locations. While this "OneTeam" model might seem to create communication and coordination burdens, most communication is handled by following the Scrum cycle. The daily Scrum meetings actually help to break down cultural barriers and disparities in work styles while simultaneously enhancing customer focus and offshore understanding of customer needs. On enterprise implementations, it can organize the project into a single whole with an integrated global

code base. Proper implementation of OneTeam provides location transparency and performance characteristics similar to hyperproductive co-located teams.

Maximum business value is delivered in Scrum by implementing the Product Backlog in order of business value of features. Xebia team product features are represented in user stories and size of a story is represented in story points [5].

Xebia teams consistently validate that distributed velocity equals colocated velocity by measuring cost per story point. The value of the feature divided by actual cost is the prime indicator of business value and this is directly proportional to the velocity of the team in story points per iteration. The Fully Distributed Scrums model is recommended for experienced Agile teams in multiple locations because cost per story point is the same as localized teams and the Xebia distributed teams have improved focus on executing stories that better fit customer needs in the right order than localized teams.

Cost per story point cannot be standardized across the industry. The best standard metric to compare productivity across projects is Function Points. Capers Jones demonstrated many years ago that the number of features delivered in Function Points can be estimated by "back-firing" using lines of code delivered [8]. While this is an indirect measure of business value delivered, it is the best measure available to compare teams industry wide.

While one might argue that delivering lots of code may not produce business value, this is controlled by Scrum teams running XP engineering practices in two ways:

- Scrum orders Product Backlog by business value and assures lines of code delivered directly increase business value.
- The XP practice of refactoring eliminates many thousands of lines of code that would remain static in the code base of a waterfall team.

The net result is that comparisons of business value delivered by Scrum/XP teams is conservative compared to waterfall teams when measured by any indicator affected by lines of code.

Thus the message of this paper is that Xebia Scrum/XP teams deliver Function Points over seven times faster than industry average waterfall teams and the Function Points they deliver have higher business value than the waterfall teams by over an order of magnitude. Since this value is delivered at the same cost per story point, and this cost is a direct indicator of business value, either locally or distributed, the OneTeam model is recommended for distributed

development by those Agile teams capable of executing it.

5. Xebia ProRail PUB Case Study

The model for Fully Distributed Scrums is best illustrated by a real life example of a Xebia OneTeam project; the ProRail PUB project.

ProRail, the logistical and infrastructural part of the Dutch railways, has been developing a new information system for travelers. Information about train departure times is stored centrally and updated with information from the rail network. When a train is delayed or arrives early this information is captured by sensors in the infrastructure as well as by manual actions to update train information.

The publishing of this information to travelers on all the railway stations throughout the Netherlands is the scope of Xebia’s development assignment. Development included the aggregation and distribution system (combining real time information about multiple trains into messages relevant for stations), the client in the displays, the audio system and the controlling and monitoring interfaces. As this is a mission critical, high-availability enterprise system with large visibility, the non-functional requirements are extensive.

Xebia took over this project from a failed waterfall implementation and meeting deadlines was now a key criteria. The transparency and empirical project control that Scrum delivers were key incentives for the client to engage Xebia. The choice to make it an offshore project was driven by cost and scalability.

While Scrum is simple to understand, it is not easy to implement and distributed development adds another layer of complexity. The PUB project encountered a number of challenges in these areas.

5.1. Lower cost of labor

The OneTeam approach for Fully Distributed Scrum teams delivers the same results as a well running colocated Scrum team in an offshoring situation. Different aspects of the PUB project can illustrate this.

5.1.1. Productivity.

Velocity of a Scrum team is determined by the number of story points that the team can complete using a standardized definition of “done” in a single iteration. As story points are not translatable between projects the PUB project size has also been measured in function points. This measure has been done for both the old (failed) implementation and the new

implementation by Xebia and these figures correspond. As a measurement this does not give a completely accurate picture, as it does not capture the amount of business value delivered very well. It is however the best means available to make comparisons over projects. Below is a table taken from a colocated 6 person Scrum, the SirsiDynix fully distributed Scrum project, and extended with PUB data.

	Cohn Colocated Scrum	Cohn Waterfall	SirsiDynix Distributed Scrum	Xebia Distributed Scrum
Person Months	54	540	827	125
Lines of Java	51000	58000	671688	100000+
Function Points	959	900	12673	1887
FP per dev. per month	17.8	1.7	15.3	15.1

Table 1: Productivity of Colocated Scrum vs. Waterfall Team [5], SirsiDynix Distributed Scrum [9], and Xebia OneTeam.

As we can see the Scrum projects easily outperform the Waterfall project. Xebia Distributed Scrum comes close to the small colocated Scrum team. The performance of the SirsiDynix and Xebia project is very similar. This shows that the high performance fully distributed Scrum approach is reproducible and not typical for only the SirsiDynix environment.

To investigate the effect of distributing teams on the productivity we can look at the cost per story point delivered throughout the project.

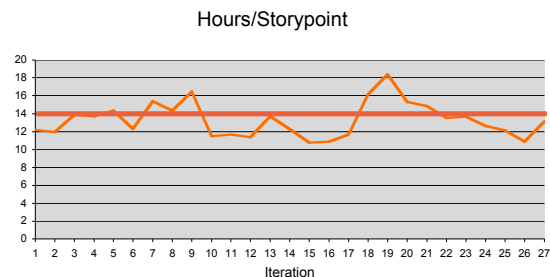


Figure 3: Hours per story point during the project

It is important to note the gradual increase in story point cost during the life of most projects due to growing complexity and growing codebase. This constant has been compensated for to focus the above diagram on any outliers. The transition from a local team to a distributed team took place at iteration 6. As can be seen from the resulting graph, the number of

hours needed to implement a story point was not affected by this distribution. Storypoint estimates were determined at the beginning of the project for the whole product backlog and were determined for new requirements as they surfaced. Iteration 18 and 19 show a significant increase in hours needed per story point. Technical debt had been built up during the previous iterations. Starting with iteration 20 this technical debt was consistently removed, resulting in a gradual increase in productivity.

5.1.2. High quality and consistency.

Throughout the course of the PUB project a lot of attention has been paid to quality. The Scrum definition of done for this project includes unit test coverage of at least 80%, fully automated functional testing, full regression testing, performance and load testing for all implemented stories as well as updating the necessary documentation.

For every piece of functionality the whole team discusses proper design and necessary refactoring takes place. In addition to this shared ownership over design every team employs a ‘quality watchdog’. This is a team member accountable for quality and consistency. Any problems that he / she signals are to be picked up and discussed by the team. All teams share the same team room and team members participate in design discussions of other teams in order to maintain architectural consistency across teams. Pair programming and rotation of people between teams is used to avoid code ownership and spread knowledge.

All issues that are found outside the iteration are measured and solved as shown in the following graph.

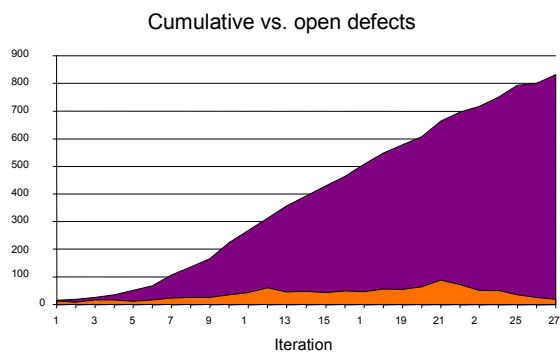


Figure 4: Open defects during the project

The above figure shows that the number of open defects remains constant (around 50). This means that the project is not building up technical debt during development. The number of open bugs per KLOC is actually decreasing because the code base is continuously growing. Other data also shows that more than 90 % of the defects found are solved in the same iteration in which they were introduced.

Based on these numbers we can conclude that the verification and validation process has isolated 6 defects per KLOC. During acceptance tests less than 1 defect per KLOC was found. A fair estimate is that 50% of the defects are still left in the product after the acceptance test, leaving us with 1 defect per KLOC. This is far less than industry average, which is around 5 defects per KLOC [14]. Fully Distributed OneTeam Scrums applying XP practices produce extremely high quality.

5.2. Capture talent not available locally

As described in 2.2 this benefit depends on the employee turnover. To cope with high turnover rates the project concentrated on clear communication, and special attention to the people-culture and people-job fit. In addition the Agile way of working provides talented people with responsible work, thus guaranteeing job satisfaction. This resulted in a turnover of less than 5% in one year.

5.2.1. Cultural differences.

Indian and Dutch team members have a different background and culture. This shows most clearly in communication. For example, where Dutch team members can be loud and direct, Indian team members can be careful and cautious in their expression. Also India is more hierarchically oriented than the Netherlands. The first and most important thing to counter these differences is good personal relationships. By traveling at the beginning and throughout the project, by seeing each other daily in video conference stand-ups, and by being part of the same team personal relationships formed. Secondly, a team culture aimed at openness and direct communication was actively developed by the ScrumMasters. This helped bring out issues during retrospectives and lowered communication barriers. Thirdly, a company culture of openness with an equal value system on both sites supported the team culture and made identifying with each other easier.

5.2.2. Sharing context and priorities

In an offshoring situation it is difficult to fully communicate all client nuances, context and priorities to offsite team members. To actively distribute this knowledge Xebia scheduled regular traveling, always-open Skype connections, a project news gazette after every iteration and informal updates by the product owner.

5.2.3. Clear communication through Scrum

The Scrum meetings facilitate practically all necessary communication. This is only possible because the team is fully distributed and shares the same sprint goals. All

Scrum meetings were done in a distributed way using video conferencing via a simple Skype video call with the exception of the Demo. Separate meeting rooms are set up with conference equipment and a Scrum planning tool with a digital burn down chart is used to share the status of the sprint across locations. A microphone is passed around as ‘talking stick’ to facilitate clear audibility. Xebia found that face to face visuals greatly increases the effectiveness of communication and enhances personal relationships.

The Sprint planning meeting is done with the whole team using planning poker [15] so that members on both shores contribute to the estimation process. Planning a distributed sprint took 4 hours on average using two week sprints.

The daily standup meetings are done when the developers in the Netherlands come to work. A distributed standup lasts no longer than 15 minutes. A Scrum of Scrums meeting was held by ScrumMasters after the stand-ups to synchronize any dependant issues or impediments as well as technological issues.

The Scrum demo was not distributed in this case to provide maximum focus and responsiveness to the customer. The Dutch members briefed the Indian members after every Demo. The Scrum retrospective goes in the same fashion as the Sprint planning meeting. The distributed retrospective is completed in 2 hours.

Together these meetings provide the full official meeting cycle. One on one meetings are being held as necessary, as well as design discussions. This is no different from a colocated Scrum with the possible exception of tooling.

5.2.4. Some work is local

While all development work can be distributed there is project work that is not easily done in a distributed way. A fourth Scrum team, consisting only of local team members, was dedicated to specific customer facing compliancy activities and removing certain impediments. Examples of local deliverables are writing Dutch documentation, aligning with customer architectural stakeholders, discussing requirements with technical stakeholders and researching technical dependencies between the infrastructure and other systems. This resulted in clearing of a lot of roadblocks and a high velocity for the distributed teams.

5.2.5. Tooling for communication and process

In this project ScrumWorks [16] was used to manage the product backlog and sprint backlog electronically. Burndown graphs were printed everyday and posted on the wall in the team rooms.

For global sharing of information and documentation a wiki was used intensively. To discuss architecture a smartboard (computerized whiteboard) was used, along with other solutions for digital whiteboarding. A single code repository, single continuous build system, test servers accessible from both locations and a shared mailing list are some of the tools used to facilitate the development process.

5.3. Project structure and scaling

Xebia initiated the PUB project with a short initiation phase where the product backlog was developed, basic architecture constraints were established and processes such as QA, Acceptance and Requirements management were set up with the customer to match the client organization in an Agile way.

After three weeks of project initiation, a colocated development team started the first iteration of the project. Iteration length was set at two weeks throughout the project.

The first two iterations were done with Dutch developers. Indian team members were included onsite as soon as immigration and logistical constraints allowed, starting with the third iteration. Both Dutch and Indian team members worked as a single colocated Scrum team with a single sprint backlog, following all XP engineering practices. In the shared onsite iterations the team members forged personal relationships to last throughout the project. By being onsite with the customer, the Indian team members acquired a good sense of customer context. It also got everyone aligned concerning practices, standards, tooling, and natural roles in the team formed. After three iterations the onsite Indian team members returned to India. During these 5 initial iterations (10 weeks) the team established colocated hyperproductivity.

The project scaled up after Indian team members returned home. Engineers were added and two new full teams were formed. Engineers in the Netherlands were split over both teams as were engineers in India, creating two teams that both have members in multiple locations. Careful attention is paid to spreading the experience among the new teams and practices like pair programming are used to get new members up to speed. This cell division like process is repeated until the project is at the desired scale.

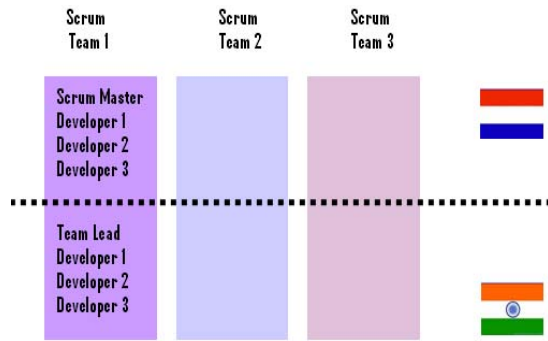


Figure 2: Fully Distributed Scrum team division

The project scaled up to three fully distributed Scrum teams and a fourth local Scrum team, with a total of 25 people. The different teams shared the same product backlog but used their own sprint backlogs.

At the end of the project the teams were scaled down and merged. As the client preferred to work with Dutch engineers for maintenance the Indian side was scaled down further. This was no problem since the use of distributed teams also ensures distributed knowledge.

The total size of the Xebia realization on this project is about 20 man-years, 100.000+ lines of code over a period of 11 months.

6. Conclusions

In summary, it is possible to create a distributed/outsourced Scrum with the same velocity and quality as a colocated team and this capability has been reproducible over many projects. The OneTeam strategy lower costs, captures offshore talent and allows increasing and decreasing team size without knowledge loss. We highly recommend this strategy for experienced Agile teams.

7. Future Research

Several other Xebia projects achieved the same velocity and quality as the PUB project confirming the OneTeam capability of distributing localized velocity and quality across continents. However, during the PUB project data collection was standardized and refined to a high level. Future projects will use the same data collection standards as the PUB project allowing a larger prospective study of the OneTeam effect with comparability across many projects.

While Xebia has now repeatedly demonstrated the effectiveness of the fully distributed team model, it requires Agile teams to fully implement the practices of Scrum and XP. Less than 10% of Agile teams worldwide are capable of doing this in 2008.

Additional studies of the fully distributed model across multiple companies would be useful and may help some companies move beyond partial implementation of Agile practices in order to achieve the fully distributed model benefits.

8. References

- [1] S. Teasley, L. Covi, M. S. Krishnan, and J. S. Olson, "How Does Radical Collocation Help a Team Succeed?," in *CSCW'00 Philadelphia*, PA: ACM, 2000, pp. 339-346.
- [2] J. Sutherland, "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects," in *AGILE 2005 Conference Denver*, CO: IEEE, 2005.
- [3] K. Beck, *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley, 1999.
- [4] H. Takeuchi and I. Nonaka, "The New New Product Development Game," *Harvard Business Review*, 1986.
- [5] M. Cohn, *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.
- [6] J. Sutherland and K. Schwaber, *The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method*. Boston: Scrum, Inc., 2007.
- [7] J. Sutherland, A. Viktorov, and J. Blount, "Adaptive Engineering of Large Software Projects with Distributed/Outsourced Teams," in *International Conference on Complex Systems* Boston, MA, USA, 2006.
- [8] C. Jones, *Software assessments, benchmarks, and best practices*. Boston, Mass.: Addison Wesley, 2000.
- [9] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in *HICSS'40, Hawaii International Conference on Software Systems* Big Island, Hawaii: IEEE, 2007.
- [10] N. Rathi, "Human resource challenges in Indian software industry: An empirical study of employee turnover," Mercer, 2004.
- [11] J. Sutherland, C. Jacobson, and K. Johnson, "Scrum and CMMI Level 5: A Magic Potion for Code Warriors!," in *Agile 2007*, Washington, D.C., 2007.
- [12] K. E. Nidiffer and D. Dolan, "Evolving Distributed Project Management," *IEEE Software*, vol. 22, pp. 63-72, Sep/Oct 2005.
- [13] M. Poppendieck, "A History of Lean: From Manufacturing to Software Development," in

- JAOO Conference*, Aarhus, Denmark, 2005.
- [14] W. S. Humphrey, *Introduction to the Personal Software Process*: Addison Wesley, 1996.
- [15] M. Cohn, *Agile Estimation and Planning*: Addison-Wesley, 2005.
- [16] Danube, "ScrumWorks Pro." vol. 2008
Seattle: Danube, 2008.