

Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams

Jeff Sutherland, Ph.D. Scrum, Inc. Boston, MA, US jeff.sutherland@computer.org	Guido Schoonheim Xebia b.v. Hilversum, Netherlands gschoonheim@xebia.com	Eelco Rustenburg Xebia b.v. Hilversum, Netherlands erustenburg@xebia.com	Maurits Rijk Xebia b.v. Hilversum, Netherlands mrijk@xebia.com
---	---	---	---

Abstract

Scrum was designed to achieve a hyperproductive state where productivity increases 5-10 times over industry averages and many collocated teams have achieved this effect. The question for this paper is whether distributed, offshore teams can consistently achieve the hyperproductive state. In particular, can a team establish a localized velocity and then maintain or increase that velocity when distributing teams across continents. Since 2006, Xebia started 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 increasing velocity with fully distributed teams. After running XP engineering practices inside many distributed Scrum projects, Xebia has systematically productized a model very similar to the SirsiDynix model [1] for high performance, distributed, offshore teams with outstanding quality.

1. Introduction

This paper introduces a highly successful model for producing distributed, offshore team productivity that is linearly scalable across continents and equal to collocated 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. A distributed team of over 50 people in the U.S. and Russia delivered velocity per developer equivalent to a collocated Scrum team and produced the same number of features as a typical 350 person waterfall team [7].

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 to be the same as SirsiDynix.

Here we discuss offshoring strategies for overcoming the geographic, language, and cultural barriers that impede distributed development and describe the secret sauce needed to avoid traditional outsourcing failures. 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 collocation doubled Agile team productivity [8]. Here, the fully distributed model supports geographically transparent software development projects where performance consistently meets or exceeds productivity of collocated Agile teams.

2. Challenges in outsourcing offshore

U.S., European, or Japanese companies often outsource software development to offshore locations like Eastern Europe, Russia, or the Far East. 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.

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. The first is not easily achievable. 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. The PatientKeeper Board permanently terminated outsourcing after reviewing these ROI data.

Capturing external talent may also be a problem. Jack Blount, CEO of Dynix and former COO of Borland refused to outsource to India and China after he verified that annual turnover rates were 30-50% [9]. And increasing staff by outsourcing can often result in loss of core knowledge when offshore staff leaves a project.

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 fully distributed Agile teams that can maintain the same velocity as onshore teams and with onshore teams that maintain the same knowledge level as offshore teams.

3. Distributed Scrum team models

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 [6] that meets regularly across geographies.

Fully distributed Scrums – Scrum teams are cross-functional with members distributed across geographies.

Isolated Scrums as in the Google AdWords project have reported the need for improve communication practices. 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. The Fully Distributed Scrum model, as shown in Xebia's OneTeam model, has all teams fully distributed and each team has members at multiple locations. While this appears to create communication and coordination burdens, 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 organizes the project into a single whole with an integrated global code base.

Maximum business value is delivered in Scrum by implementing the Product Backlog in order of business value of features. Xebia product features are represented by user stories and size of a story is represented in story points [5]. Xebia teams measure cost in Euros per user story. The value of the feature divided by actual cost is the prime indicator of business value delivered and this is directly proportional to the velocity of the team in story points per iteration.

Xebia teams consistently validate that distributed velocity equals collocated velocity as measured by cost per story point, a direct indicator of business value. 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, counterintuitively, Xebia distributed teams have better focus on executing stories that fit customer needs than localized teams.

The best standard metric to compare productivity across projects is Function Points as it directly represents features delivered. Capers Jones demonstrated years ago that the number of features delivered in Function Points can be estimated by "back-firing" using lines of code delivered [7]. While this is a less direct measure of business value, it is the best measure available to compare teams industry wide.

One might argue that delivering lots of code may not produce business value. Scrum teams running XP engineering practices deliver more features per line of code than industry average project teams because:

- Scrum orders Product Backlog by business value and assures lines of code delivered maximize 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, and no other model in the history of software development has demonstrated this capability, the OneTeam model is recommended for distributed development by those Agile teams capable of executing it.

4. 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.

Time was critical due to previous waterfall team failures and meeting deadlines was 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.

4.1. 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 QA, Acceptance and Requirements management were set up with the customer.

After three weeks of project initiation, a collocated Dutch development team completed the first two iterations. Iteration length was set at two weeks throughout the project. Indian team members were included onsite starting with the third iteration. Both Dutch and Indian team members worked as a single collocated 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 and 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 first 5 iterations (10 weeks) the team established collocated hyperproductivity.

The project scaled up after Indian team members returned home. Engineers were added and two new teams were formed, each with 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.

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.

4.2. Advantages realized

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

4.2.1. Productivity. During the project, velocity is determined by the number of story points that the team can realize 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. While this only approximates

business value, it is the best means available to make comparisons over projects. Below is a table taken from a collocated 6 person Scrum (*) the SirsiDynix project (**) and extended with PUB data .

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

	Colocated Scrum*	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 shows Scrum projects easily outperform the waterfall project. Xebia Distributed Scrum is close to the collocated Scrum and 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 unique to the SirsiDynix environment.

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

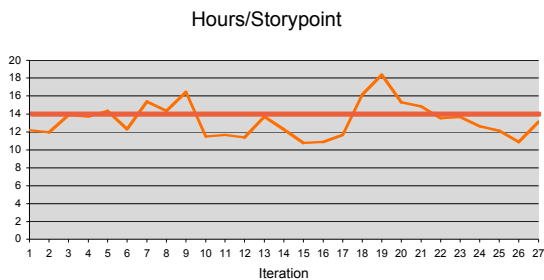


Figure 1: Project costs in hours per story point

It is important to note is that there is a 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.

4.2.2. Clear communication through Scrum. The Scrum meetings facilitate almost all necessary communication. This is 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 using a digital burn down chart 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 so that members on both shores contribute to the estimation process. Planning a distributed sprint took 4 hours on average.

The daily standup meetings are done when the Netherlands come to work. A distributed standup lasts no longer than 15 minutes.

The retrospective goes in the same fashion as the Sprint planning meeting. The distributed retrospective is completed in 2 hours.

The demo was not shared in this case to provide maximum focus and responsiveness to the customer. The Dutch members briefed the Indian members after every Demo.

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.

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

4.2.3. 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 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.

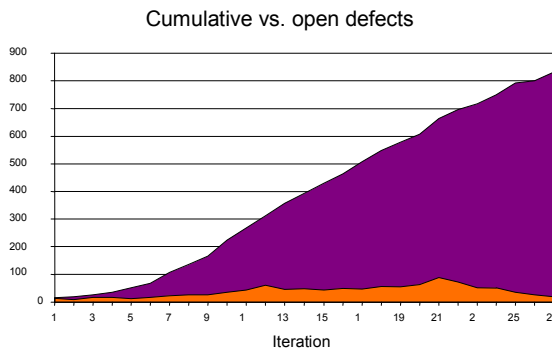


Figure 2: Cumulative vs. open defects in project

All issues that are found outside the iteration are measured and solved as shown in the graph above which shows that the number of open defects remains constant (around 50) and 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 issue 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 [10]. Fully Distributed OneTeam Scrums applying XP practices produce extremely high quality.

4.3.4. Transparency and control. The project could accurately estimate required time and budget by combining the product backlog with good estimates and velocity measured over time [11]. This gave the client all required information to be in control. Providing transparency and proof of progress by

showing working software after every iteration created visibility and built trust.

4.3. Challenges faced

While Scrum is simple to understand, it is not that easy to implement. Distributed development adds another layer of complexity. The PUB project encountered a number of challenges in these area's.

4.3.1 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 stand-ups, and by being part of the same team we started building relationships that were focused on the person. 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.

4.3.2. 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 we scheduled regular traveling, always-open Skype connections, a project news gazette after every iteration and informal updates by the product owner.

4.3.3. Managing customers new to Agile. Although the Scrum process does not require a formal project manager, Xebia does add a project manager to projects. He or she handles financial arrangements and client expectations and most importantly does whatever it takes to make Scrum work for this customer. As the client did not have previous Agile experience, the project manager worked with the client as a Meta ScrumMaster / coach to bring the organization into an Agile way of working and acted as proxy product owner. This provided the teams and Scrum masters a clear product backlog and interface to the client organization from day one. The proxy product owner / project manager ensured proper Agile planning and was in continuous dialog with the client about deadlines, scope and progress. The ScrumMasters focused on the sprints, the process and the quality.

4.3.4. Some work is local. While all development work can be distributed there is project work that is not easily done in a distributed way. The fourth Scrum team (see 4.1), 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.

4.3.5. Tooling for communication and process. In this project ScrumWorks 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. Conclusions

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

6. About Xebia

"We feel that our customers have a right to work with the most effective teams, to have those efforts focused on the priorities that our customer determines, to get high quality software and to have full transparency and control over the project planning."

Xebia is an international Agile software development company, with offices in the Netherlands, France and India. The company is specialized in Java technology, Agile offshoring & projects, Agile consultancy and training, IT Architecture and Auditing. See <http://www.xebia.com/>.

7. References

- [1] Sutherland, J., Viktorov, A., and Blount, J.: "Adaptive Engineering of Large Software Projects with Distributed/Outsourced Teams". Proc. International Conference on Complex Systems, Boston, MA, USA, 25-30 June 2006.
- [2] Sutherland, J.: "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects". Proc. AGILE 2005 Conference, Denver, CO, July 24-29 2005.
- [3] Beck, K.: "Extreme Programming Explained: Embrace Change", Addison-Wesley, Boston, 1999.
- [4] Takeuchi, H., and Nonaka, I.: "The New New Product Development Game", Harvard Business Review, 1986, (January-February).
- [5] Cohn, M.: "User Stories Applied : For Agile Software Development", Addison-Wesley, 2004.
- [6] Sutherland, J., and Schwaber, K.: "The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method", Scrum, Inc., Boston, 2007.
- [7] Jones, C.: "Software assessments, benchmarks, and best practices", Addison Wesley, Boston, Mass., 2000.
- [8] Teasley, S., Covi, L., Krishnan, M.S., and Olson, J.S.: "How Does Radical Collocation Help a Team Succeed?". Proc. CSCW'00, Philadelphia, PA, 2000, pp. 339-346.
- [9] Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N.: "Distributed Scrum: Agile Project Management with Outsourced Development Teams". Proc. HICSS'40, Hawaii International Conference on Software Systems, Big Island, Hawaii, 2007.
- [10] Humphrey, W.S.: "Introduction to the Personal Software Process", Addison Wesley, 1996.
- [11] Cohn, M.: "Agile Estimation and Planning", Addison-Wesley, 2005.