July 23-28
AGILE2006 International Conference
Minneapolis, Minnesota USA

Distributed Scrum: Agile Project Management with Outsourced Development Teams

Jeff Sutherland, Ph.D.
Patientkeeper
Newton, MA, US
jeff.sutherland@computer.org

Anton Viktorov
StarSoft Development Labs
St. Petersburg, Russia
anton.viktorov@starsoftlabs.com

Jack Blount
SirsiDynix
Provo, UT, US
jack@dynix.com

Abstract

Agile project management with Scrum derives from Takeuchi and Nonaka analyses of best practices in companies like Fuji-Xerox, Honda, Canon, and Toyota. Toyota routinely achieves four times the productivity and 12 times the quality of competitors. Can Scrum do the same for globally distributed teams? Two Agile companies, SirsiDynix using Scrum, and StarSoft Development Labs using Scrum with some XP engineering practices, achieved comparable performance developing a Java application with over 1,000,000 lines of code. Their Horizon project is a completely new implementation of a library system with over 12,500 installed sites. During the most recent year of the project, a distributed team of 56 Scrum developers working from Provo, Utah; Waterloo, Canada; and St. Petersburg, Russia, delivered 671,688 lines of production Java code. Using XP refactoring techniques they then systematically eliminated 275,000 lines of code to achieve better usability, performance, reliability, and maintainability. At 15.3 function points per developer/month, this is one of the most productive projects ever documented. SirsiDynix best practices are similar to those observed on distributed Scrum teams at IDX Systems, radically different than those promoted by PMBOK, and counterintuitive to some practices advocated by the Scrum Alliance. This paper analyzes and recommends new best practices for globally distributed Agile teams.

1. Introduction

Many U.S., European, or Japanese companies today outsource software development to Eastern Europe, Russia, or the Far East. Typically, remote teams operate somewhat independently and communication problems limit productivity. While there is a large amount of research and practice literature on project management, distributed development, and outsourcing strategies as isolated domains, there are few detailed studies of best project management practices on enterprise systems that are both distributed and outsourced.

Best current Scrum practice is for local Scrum teams at all sites to synchronize once a day via a Scrum of Scrums meeting. Here we describe something rarely seen on large, distributed, product development teams. At SirsiDynix, all Scrum teams consist of developers distributed across different sites. All members of the team work together on the same set of tasks on the same build of software. Any team member from any site can work on any team task. While some Agile companies operate in this unified distributed manner on a small scale, SirsiDynix has been successful in using fully integrated Scrum teams with over 50 developers in the U.S., Canada, and Russia, on a completely new implementation of the platform and system architecture for a complex Integrated Library System (ILS). An ILS system can best be compared to a vertical market ERP system with a public portal interface used by more than 200 million people. New best practices for distributed Scrum seen on this project consist of (1) daily Scrum meetings of all developers from multiple sites, (2) daily meetings of Product Owner team (3) hourly automated builds from one central repository, (4) no distinction between developers at different sites on the same team, (5) and seamless integration of XP practices like pair programming with Scrum. While similar practices have been implemented on small distributed Scrum teams [1] this is the first well-documented, successful project that has moved Scrum practices to a new level for large distributed/outsourced teams building complex enterprise systems.

2. Distributed Team Models

Best practices recommended by the Scrum Alliance include partitioning work across distributed Scrum teams while eliminating most dependencies between teams. In some cases teams are not cross-functional. Requirements may be created in the U.S. and developed in Dubai, or development may occur in Germany and quality assurance in India. We can consider this a Type A distribution model that partitions work across disparate teams. Cross-cultural communication problems are compounded by disparities in work types.

The latest thinking in the Project Management Institute Guide to the Project Management Body of Knowledge (PMBOK) models [2] 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 [3]. This 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 [4]. This is an extreme Type A distribution model.

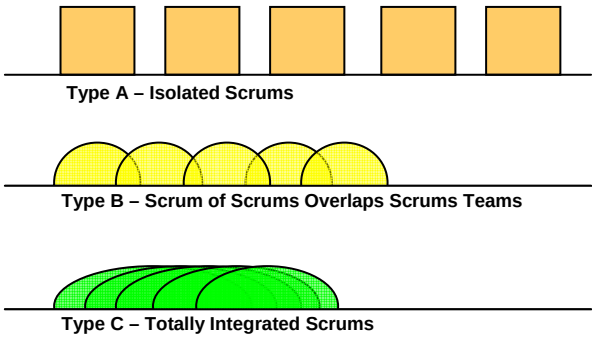

Figure 1: Type A, B, and C strategies for distributed Scrum teams [5].

A more effective approach advocated by current Certified ScrumMaster Trainers is to have distributed cross-functional teams that are relatively autonomous, linked by a Scrum-of-Scrums where ScrumMasters (team leaders/project managers) meet regularly together from all locations. This makes teams feel more co-equal and encourages communication, cooperation, and cross-fertilization. This can be viewed as a Type B distribution model where every team has overlap via the Scrum-of-Scrums meetings.

A Type C model has all teams fully distributed and each team has members at multiple locations. While on the surface, this appears to create communication and coordination burdens, the daily Scrum meetings help to break down cultural barriers and disparities in work styles. On large enterprise implementations, it can organize the project into a single whole with a rapidly evolving global code base. The virtual nature of this approach provides location transparency, which creates performance characteristics similar to a small co-located team. The hyperproductive Web team at IDX Systems during 1996-2000 achieved ten times the performance of the industry average for teams of large systems [1]. The SirsiDynix model outlined in this paper is a good example of best practices for distributed teams. This may be the most productive distributed team ever documented, delivering a large Java enterprise system with more than one million lines of code.

2. SirsiDynix and StarSoft Development Labs

SirsiDynix provides global technology solutions for libraries to assist people in discovering and using knowledge, resources and other valuable content for their educations, jobs and entertainment. In concert with key industry partners, SirsiDynix supports this strategic role for libraries by offering a comprehensive integrated suite of technology solutions for improving the internal productivity of libraries and enhancing their capabilities for meeting the needs of people and communities. SirsiDynix has approximately 4,000 library and consortia clients, serving more than 200 million people through more than 20,000 library outlets in the Americas, Europe, Africa, the Middle East and Asia-Pacific.

StarSoft Development Labs, Inc. is a fast-growing software outsourcing service provider in Russia and Eastern Europe. Headquartered in Cambridge, Massachusetts, USA, StarSoft operates development centers in St. Petersburg, Russia and Dnepropetrovsk, Ukraine, employing over 450 professionals. They have particular expertise in specialized business applications, enterprise portals, e-business solutions, and legacy migration projects. StarSoft has experience handling development efforts varying in size and duration from just several engineers working for a few months to large-scale projects involving dozens of developers and spanning over several years. A CMM Level 3 company, StarSoft successfully uses Agile development methodologies for the benefits of its clients.

Jack Blount, President and CEO of Dynix and now CTO of the merged SirsiDynix company, negotiated and signed the Russian portion of the project. StarSoft staffed the project with more than 20 qualified engineers in less than 60 days. Significant development milestones were completed in just a few weeks and all joint development

projects were efficiently tracked and continue to be on schedule.

The hidden costs of outsourcing can be significant beginning with startup costs. Barthelemy [6] surveyed 50 companies and found that 14% of outsourcing operations were failures. In the remainder, costs of transitioning to a new vendor often canceled out most of the company's savings from lower labor costs in other countries. The average time from evaluating outsourcing to beginning of vendor performance was 18 months for projects smaller than the SirsiDynix contract. As a result, the MIT Sloan Management Review counsels readers not to outsource critical IT functions and to spend more time planning. The German Institute for Economic Research analyzed 43,000 German manufacturing firms from 1992-2000 and found that outsourcing services led to poor corporate performance, while outsourcing production helped [7].

Figure 2 – SirsiDynix lines of new Java code over time. StarSoft contribution at end of 2004 more than doubled rate of growth of code base.

SirsiDynix sidestepped many of the hidden costs, directly outsourced primary production and used Distributed Scrum Type C to control the risk. They acted like Toyota by implementing a lean approach. Production velocity more than doubled when they increased the size of the 30 person North American development team and added 26 people from StarSoft on 1 December 2005 (see Figure above).

3. Intent of Type C Distributed Scrum

An Agile company building a large product and facing time-to-market pressure needs to quickly double or triple productivity. The local talent pool is not sufficient to do this. Outsourcing is only a solution if Agile practices are enhanced by capabilities of the outsourced teams. The primary driver is enhanced technical capability. Cost savings are a secondary driver.

4. Context

Software complexity and demands for increased functionality are exponentially increasing in all industries. When the lead author of this paper flew F-4 aircraft in combat in 1967, 8% of pilot functions were supported by software. In 1982, the F16 software support was 45%, and by 2000, the F22 was augmented 80% of pilot capabilities with software [2]. Demands for ease of use, scalability, reliability, and maintainability increase with complexity.

Large software projects are very high risk. The 2003 Standish Chaos Report show success rates of only 34%. 51% of projects are over budget or lacking critical functionality. 15% are total failures [8].

SirsiDynix was confronted with the requirement to completely re-implement a legacy library system with over 12,500 installed sites across the globe. The large number of developers required over many years in the midst of a changing business environment threatened to obsolete many feature requirements in the middle of the project. To complicate matters further, the library software industry was in a consolidating phase and mergers and acquisitions were on the horizon. Dynix started the project in 2002 and merged with Sirsi in 2005 to form SirsiDynix.

Fortunately, Dynix started the project with a scalable Agile process that could adapt to changing requirements throughout the project. Time to market demanded more than doubling of output. That could only happen by augmenting resources with Agile teams. StarSoft was selected because of their history of successful XP implementations and their experience with systems level software.

The combination of high risk, large scale, changing market requirements, merger and acquisition business factors, and the SirsiDynix experience with Scrum combined with StarSoft success with XP led them to choose a Type C distributed Scrum implementation. Jack Blount's past experience with Agile development projects at US Data Authority, TeleComputing and JD Edwards where he had used Type A and Type B distributed Scrum also was a key factor in his decision to structure the project as a Type C project. Frontline production was outsourced and teams were fully integrated across locations.

4. Forces

4.1 Complexity Drivers

The Systems and Software Consortium (SSCI) of large defense contractors has outlined drivers, constraints, and

enablers that force organizations to invest in real-time project management information systems. Scalable Scrum implementations with minimal tooling are one of the best real-time information generators in the software industry.

SSCI complexity drivers are described as [2]:

1. Increasing problem complexity shifting focus from requirements to objective capabilities that must be met by larger teams and strategic partnerships.
2. Increasing solution complexity which shifts attention from platform architectures to enterprise architectures and fully integrated systems.
3. Increasing technical complexity from integrating stand alone systems to integrating across layers and stacks of communications and network architectures.
4. Increasing compliance complexity shifting from proprietary to open standards.
5. Increasing team complexity shifting from a single implementer to strategic teaming and mergers and acquisitions.

SirsiDynix faced all of these issues. Legacy products were difficult to sell to new customers. They needed a new product with complete functionality for the library enterprise based on today's technologies that was highly scalable, easily expandable, and used the latest computer and library standards,

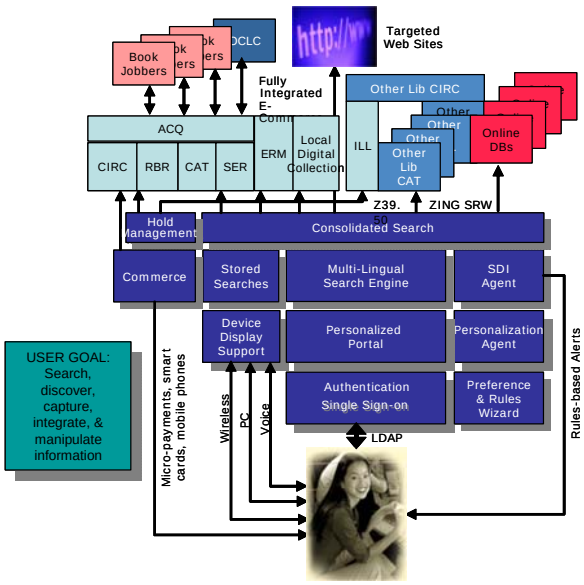

Figure 3 – SirsiDynix Horizon 8.0 Architecture

The Horizon 8.0 architecture supports a wide variety of users from publication acquisition to cataloging, searching, reserving, circulating, or integrating information from local and external resources. The decision was made to use Java with J2EE, a modular design, database independency, maximum use of free platforms and tools, and wide support of MARC21, UNIMARC, Z39.50 and other ILS standards.

The project uses a three-tier architecture and uses Hibernate as a database abstraction layer. Oracle 10g, MS SQL, and IBM DB2 support is provided. The JBoss 4 Application server is used with a Java GUI Client with WebStart bootstrap. It is a cross-platform product supporting MS Windows 2000, XP, 2003, Red Hat Linux, and Sun Solaris. Built-in multi-language support has on-the-fly resource editing for ease of localization. Other key technologies are JAAS, LDAP, SSL, Velocity, Xdoclet, JAXB, JUnit, and Jython.

4.2 Top Issues in Distributed Development

The SSCI has carefully researched top issues in distributed development [2], all of which had to be handled by SirsiDynix and StarSoft.

1. Strategic: Difficult leveraging available resources, best practices are often deemed proprietary, are time consuming and difficult to maintain.
2. Project and process management: Difficulty synchronizing work between distributed sites.
3. Communication: Lack of effective communication mechanisms.
4. Cultural: Conflicting behaviors, processes, and technologies.
5. Technical: Incompatible data formats, schemas, and standards.
6. Security: Ensuring electronic transmission confidentiality and privacy.

The unique way in which SirsiDynix and StarSoft implemented a Type C Distributed Scrum carefully addressed all of these issues.

5. Solution: Type C Distributed Scrum

There are three roles in a Scrum: the Product Owner, the ScrumMaster, and the Team. SirsiDynix used these roles. Scrum itself solves the strategic distribution problem of building a high velocity, real-time reporting organization with an open source process that is easy to implement and low-overhead to maintain [9].

5.1 Team Formation

The second major challenge is process management, particularly synchronizing work between sites. This was achieved by splitting teams across sites and fine tuning daily Scrum meetings.

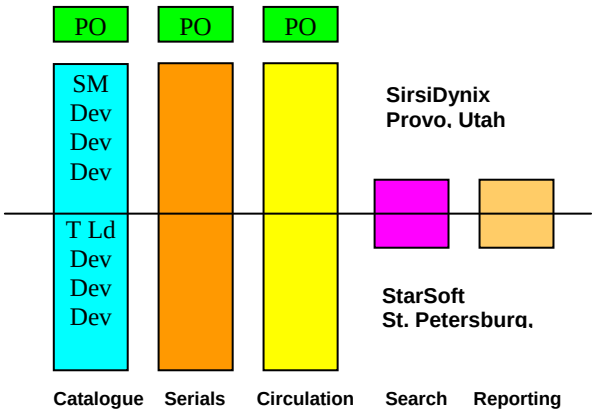

Figure 4 – Scrum teams split across sites. PO=Product Owner, SM=ScrumMaster, TLd=Technical Lead.

Teams at SirsiDynix were split across the functional areas needed for a integrated library system. Half of a Scrum team is typically in Provo, Utah, and the other half in St. Petersburg. There are typically 3-5 people on the Utah part of the team and 4 or more on the St. Petersburg portion of the team. The Search and Reporting Teams are smaller. There are smaller numbers of team members in Seattle, Denver, St. Louis, and in Waterloo, Canada.

The ScrumMasters are all in Utah, Canada, and Colorado. The Product Owners who are assigned to teams are all located in Utah. Architecture meetings are in Utah with Utah and Colorado based architects. This makes it easy to coordinate and communicate efforts across teams.

There are 30 team members in North America and 26 team members in St. Petersburg on this project. The St. Petersburg team has one project leader, 3 technical team leaders, 18 developers, 1 test lead, and 3 testers. This is a low developer/test ratio and makes it impossible to have a fully tested package of code at the end of the Sprints. This is an area for improvement.

5.2 Scrum Meetings

Teams meet across geographies at 7:45am Utah time which is 17:45 St. Petersburg time. The team has found it necessary to answer the three Scrum questions in writing and distribute the answers by email before the Scrum meeting. This shortens the time needed for teleconference on the joint meeting and helps overcome any language barriers. Each individual reports on what they did since the last meeting, what they intend to do next, and what impediments are blocking their progress.

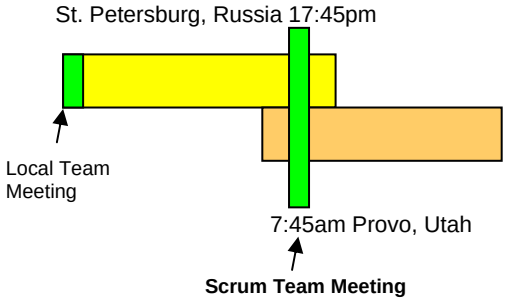

Figure 5 – Scrum Team meetings

Local sub-teams have an additional standup meeting at the beginning of the day in St. Petersburg. Everyone is using the same process and technologies and daily meetings coordinate activities within the teams.

ScrumMasters are all in Provo, Utah or Waterloo, Canada, and meet in a Scrum of Scrums every Monday morning. Here work is coordinated across teams. An Architecture group also meets on Monday and controls the direction of the project architecture through the Scrum meetings.

5.3 Sprints

Sprints are two weeks on the SirsiDynix project. There is a Sprint planning meeting that is the same as an XP release planning meeting in which requirements from User Stories are broken down into development tasks. Most tasks require a lot of questions from the Product Owners and tasks occassionally take more time than initial estimates.

The lag time for Utah Product Owner response to questions on User Stories forces multitasking in St. Petersburg and this is not the ideal situation. Sometimes new tasks are discovered after querying Product Owners during the Sprint with additional feature details.

Code is feature complete and demoed at the end of each Sprint. If it meets the Product Owner's functional requirement, it is considered done. It is not deliverable code and SirsiDynix wants to strengthen its definition of "done" to include testing. Failure to do this allows work in progress to cross Sprint boundaries, introducing wait times and greater risk into the project.

5.4 Product Specifications

Requirements are in the form of User Stories used in many Scrum and XP implementations. Some of them are lengthy and detailed, others are not. A lot of questions result after receiving the document in St. Petersburg which are resolved by in daily Scrum meetings, by instant messaging, or by email.

Story for Simple Renewals Use Case - *Patron brings item to staff to be renewed.*

Patron John Smith checked out "The Da Vinci Code" the last time he was in the library. Today he is back in the library to pick up something else and brings "The Da Vinci Code" with him. He hands it to the staff user and asks for it to be renewed. The staff user simply scans the item barcode at checkout, and the system treats it as a renewal since the item is already checked out to John. This changes the loan period (extends the due date) for the length of the renewal loan. Item and patron circulation history are updated with a new row showing the renewal date and new due date. Counts display for the number of renewals used and remaining. The item is returned to Patron John Smith.

Assumptions:

Item being renewed is currently checked out to the active patron
No requests or reservations outstanding
Item was not overdue
Item does not have a problem status (lost, cr, etc)
No renew maximums have been reached
No block/circ maximums have been reached
Patron's subscriptions are active and not within renewal period
No renewal charges apply
No recalls apply
Renewal is from Check Out (not Check In)
Staff User has renewal privileges

Verification (How to verify completion)

1.1. Launch Check Out
1.2. Retrieve a patron who has an item already checked out but not yet overdue
1.3. Enter barcode for checked out item into barcode entry area (as if it is being checked out), and press <cr>.
1.4. System calculates new due date according to circ rules and agency parameters.
1.5. The renewal count is incremented (Staff renewal with item)
1.6. If user views "Circulation Item Details", the appropriate Renewals information should be updated (renewals used/remaining)
1.7. Cursor focus returns to barcode entry area, ready to receive next scan (if previous barcode is still displayed, it should be automatically replaced by whatever is entered next)
1.8. A check of the item and patron circulation statistics screens show a new row for the renewal with the renewal date/time and the new due date.

For this project, St. Petersburg staff like a detailed description because the system is a comprehensive and complex system designed for specialized librarians. As a result, there is a lot of knowledge that needs to be embedded in the product specification.

The ways libraries work in St. Petersburg are very different than English libraries. Russian libraries operate largely via manual operations. While processes look similar to English libraries on the surface, the underlying details are quite different. Therefore, user stories do not have sufficient detail for Russian programmers.

5.5 Testing

Developers write unit tests. The Test team and Product Owners do manual testing. An Automation Test team in Utah creates scripts for an automated testing tool. Stress testing is as needed.

The test first approach is encouraged although not mandated. Tests are written simultaneously with code most of the time. GUIs are not unit tested. Manual testing does not need to be complete for Sprint Demo leading to a lot of open work in progress.

During the Sprint, the Product Owner tests features that are in the Sprint backlog. Testers receive a stable Sprint build only after the Sprint demo.

5.6 Configuration Management

SirsiDynix was using CVS as source code repository when the decision was made to engage an outsourcing firm. At that time, SirsiDynix made a decision that CVS could not be used effectively because of lack of support for distributed development, largely seen in long code synchronization times. Other tools were evaluated and Perforce was chosen as the best solution.

StarSoft had seen positive results on many projects using Perforce. It is fast, reliable and offers local proxy servers for distributed teams. Although not a cheap solution, it has been very effective for the SirsiDynix project.

Automated builds are run every hour with email generated back to developers. It takes 12 minutes to do a

build, 30 minutes if the database changes. StarSoft would like to see faster builds and true concurrent engineering. Right now builds are only stable every two weeks at Sprint boundaries.

5.7 Pair Programming, Refactoring, and other XP practices

StarSoft is an XP company and tries to introduce XP practices into all their projects. Pair programming is done on more complicated pieces of functionality. Refactoring in planned for future Sprints and not done in every iteration as in XP. Some radically refactoring has occurred as the project approaches completion (see Figure 1).

5.8 Measuring Progress

The project uses the Jira <http://www.atlassian.com> issue tracking and project management software. This gives everyone on the project a real-time view into the state of Sprints. It also provides comprehensive management reporting tools. The Figure below shows the Sprint burn-down chart and a snapshot of Earned Business Value on the project along with a synopsis of bug status.

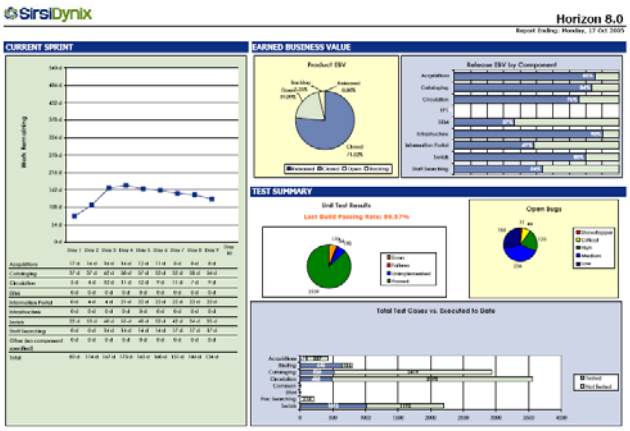

Figure 6 – SirsiDynix Horizon 8.0 Project Dashboard

Data from Jira can be downloaded into Excel to create any requested data analysis. High velocity complex projects need an automated tool to track status across teams and geographies. The best tools support bug tracking and status of development tasks in one system to avoid extra work on data entry by developers. Such tools should track tasks completed by developers and work remaining. They provide more detailed and useful data than time sheets, which should be avoided. Time sheets are extra overhead that do not provide useful information on the state of the project, and are de-motivating to developers.

Other companies like PatientKeeper [10] have found tools that incorporate both development tasks and defects that can be packaged into a Sprint Backlog are highly useful for complex development projects. Thousands of tasks and dozens of Sprints can be easily maintained and reviewed simultaneously with the right tool.

6. Type C Distributed Scrum Resulting Context

Collaboration of SirsiDynix and StarSoft turned the Horizon 8.0 project into one of the most productive Scrum projects ever documented. For example, Mike Cohn provided excellent data on a project that was done initially with a waterfall team and then re-implemented with a Scrum team.

The waterfall team took 9 months with 60 people and generated 54000 lines of code. It was re-implemented by a Scrum team of 4.5 people in 12 months. The resulting 50,803 lines of code have more functionality and higher quality.

Capers Jones of Software Productivity Research has published extensive tables on average number of function points per lines of code for all major languages [11]. Since the average lines of code per function point for Java is 53, we can estimate the number of function points in the Scrum application. The waterfall implementation is known to have fewer function points.

	SCRUM	Waterfall	SirsiDynix
Person Months	54	540	827
Lines of Java	51,000	58000	671,688
Function Points	959	900	12673
FP per dev/month	17.8	2.0	15.3
FP per dev/month (industry average)	12.5	12.5	3

Distributed team working on Horizon 8.0 generated 671,688 lines of code in 14.5 months with 56 people. During this period they radically refactored the code on two occasions and reduced the code based by 275,000. They have not been penalized for radical refactoring as that

is rarely done in large waterfall projects in the database from which Capers derived his numbers.

Jones has also shown from his database of tens of thousands of projects that industry average productivity is 12.5 function points per developer/month for a project of 900 function points and that this drops to 3 for a project with 13000 function points [12].

The SirsiDynix project is almost as productive as the small Scrum project with a collocated team of 4.5 people. For a globally dispersed team, it is one of the most productive projects ever documented at a run rate of five times industry average.

7. Conclusions

It is extremely easy to integrate Scrum with XP practices even on large distributed teams. This can improve productivity, reduce project risk, and enhance software quality.

What is new in this paper is that single teams with members distributed across sites can enhance code ownership and improve autonomy essential to team self-organization. One Scrum meeting a day was necessary which included all team members across geographies. Written communication prior to joint meetings was needed to improve communication and reduce misunderstandings due to cultural and distance barriers. Project leaders in Provo, Utah, and St. Petersburg had a remarkable common view of the project because of the transparency and frequency of communications.

Automated communication of Product and Sprint backlogs throughout the organization combined with upward reporting of Scrum status to management can tightly align a global organization.

The issues of Product Backlog being “ready” for implementation in a Sprint and working software being “done” at the end of a Sprint are key areas where even the best teams need improvement.

7. References

1. Sutherland, J., *Agile Can Scale: Inventing and Reinventing Scrum in Five Companies.* Cutter IT Journal, 2001. **14**(12): p. 5-11.
2. Nidiffer, K.E. and D. Dolan, *Evolving Distributed Project Management.* IEEE Software, 2005. **22**(5): p. 63-72.
3. Zanoni, R. and J.L.N. Audy. *Projected Management Model for Physically Distributed Software Development Environment*. in *HICSS'03*. 2003. Hawaii: IEEE.
4. Poppendieck, M. *A History of Lean: From Manufacturing to Software Development*. in *JAOO Conference*. 2005. Aarhus, Denmark: EOS.
5. Takeuchi, H. and I. Nonaka, *Hitotsubashi on Knowledge Management*. 2004, Singapore: John Wiley & Sons (Asia).
6. Barthelemy, J., *The Hidden Costs of Outsourcing.* MITSloan Management Review, 2001. **42**(3): p. 60-69.
7. Gorzig, B. and A. Stephan, *Outsourcing and Firm-level Performance*, in *DIW Berlin.* 2002, German Institute of Economic Research.
8. StandishGroup. *2003 Chaos Chronicles*. 2003 [cited; Available from: http://www.standishgroup.com/press/article.php?id=2.
9. Sutherland, J. *Scrum Evolution: Type A, B, and C Sprints*. in *Agile 2005 Conference*. 2005. Denver, CO.
10. Sutherland, J., *Future of Scrum: Parallel Pipelining of Sprints in Complex Projects with Details on Scrum Type C Tools and Techniques*. 2005, PatientKeeper, Inc.: Brighton, MA. p. 1-27.
11. Jones, C., *Programming Languages Table, Release 8.2*. 1996, Software Productivity Research: Burlington, MA.
12. Jones, C., *Software assessments, benchmarks, and best practices / Capers Jones*. Addison-Wesley information technology series. 2000, Boston, Mass.: Addison Wesley. xxiii, 659 p.