International Conference on Advance Studies in Engineering and Sciences
Sri Satya Sai University of Technology and Medical Sciences , Sehore (M.P.) ICASES-17
2nd December 2017, www.conferenceworld.in ISBN: 978-93-86171-83-2

EVOLUTION OF PROJECT MANAGEMENT WITH DISTRIBUTED SCRUM AND PARALLEL PIPELINING

CFE Anitya Kumar Gupta[1] , Ms. Srishti Gupta [2]

Expertise Area Machine Learning and Virtual Reality, B.Tech Cloud Computing Apeejay Stya University (India)[1]

Microsoft Student Partner, Vice Chairperson ACM, B.Tech Electronics and Communication BVCE (India)[2]

ABSTRACT

Agile project management with the help of scrum framework has been derive from the best business in top notch companies which are achieving the highest productivity and approximately twelve times the quality of competitors. One of the company which first took their steps forward to make the agile environment with scrum framework to make the eye catch of the world on the new environment was 'SirsiDynix' whose approach was related to distributed scrum teams at IDX systems corresponding to the different approaches of the competitors like 'PMBOX' which are advocated by the alliances of the scrum. This paper will throws the light on best practices for globally distributed agile teams. It will also represents the parallel technique of pipelining of sprints in complex projects which is the future of scrum.

I.INTRODUCTION

Scrum is an agile development process which has been designed to strengthen different parameters like focus, clarity, energy, and transparency to project teams developing software systems. It controls the artificial life research by providing the permission to the teams to operate closely to an edge of disruption to promote evolution of rapid system. It basically focuses on subsumption of robot architectures by enforcing set of rules that will allow speedy self-organization of scrum teams to generate systems with evolving architectures. An implementation of the Scrum was been designed to increase the speed of an individual align and create a culture which has been driven by the performance, supporting the creation of the values by the shareholders, achieving stability, and consistently communication of performance at every level or iteration and enhancement of the quality of life.

The development process needs to support teams of enterprise where immediate virtual design generates working code. Key factors that influenced Scrum framework which was been imbibe with the fundamental issues in the development:

- Uncertainty which is inevitable in products.
- Moving towards the new iteration the requirements in the form of feedback is must require until after the users have used it.
- Defining the whole interactive system.
- Changing and ambiguous requirements.

So therefore the researchers designed and launched 'All – at – once 'model to overcome above issues. They assume the creation of software which will work on requirements, coding, testing then delivers the entire system to the representatives. These assumptions were made to be happen with the help of quantum methodologies.

II.SCRUM EVOLUTION

Evolution occurs in dynamic response. During the high percentage of availability of scrum masters retrospection can help guide future activities. In particular the main focus is on Scrum Theory, Scrum Evolution and Scrum Preconceptions. One of the research of Harvard University showcase the product development which was been separated into different phases: Type A, Type B, and Type C. Isolated cycles of work belonging to the methodology of Type A was been executed by NASA with the help of relay race process. NASA also approached Type B which was been popularly called 'Sashimi' because slices of the work has been overlapped. When the drastic pickup was been seen in scrum framework than Honda which was been using the approach of 'Rugby' where multiple phases of product development. It was been predicted that Scrum is Rugby formation and they viewed an 'all – at – once' process where team moves down the field passing the ball back and forth.
After the notion of various types of Scrum with certified scrum masters with the development team of top notch companies like Microsoft, Yahoo, Adobe, and other companies which applies quantum techniques to a higher level of thinking about three types of Scrum.

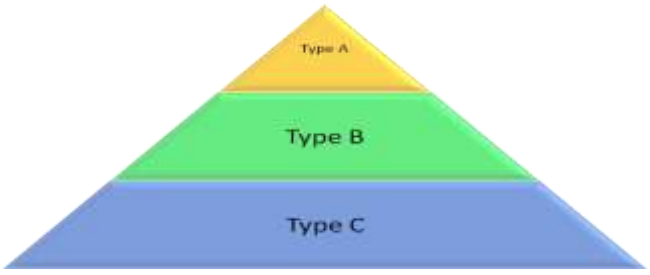


Figure 1 Scrum Types

If we talk about the practical practice, solution Type B has been automated in which the lost time and productivity between sprints has been eliminated. Within the sprint time box the throughput has been increased. During the period of 2001 Type C scrum has been start executed by different organisation. Sprint Backlog has been automated with improved with improved tools in order to maintain team focus. Automated regression testing was significally enhanced. Quality Assurance was then been modified to provide a small QA team for

each of four to six overlapping production software releases. Pair programming was been used and team programming was common where many programmers worked together around a table for the entire day. The result has been delivery of production code to a new set of enterprise customers for every Sprint with its maintenance. Scalability of the project with scrum framework has been improved during the Type C. But it requires the high level of product adoption by difficult and discriminating users. Its support for wireless networks across an enterprise, integration with financial systems in diverse IT infrastructures and certification by the customer.

III.DIFFERENT MODELS

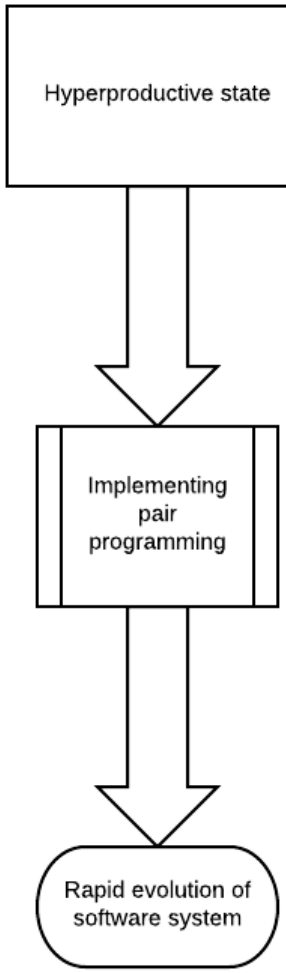
This is to note that before coming of discussion on different types of Scrum; we should understand about the different models.

Figure 2 Different Scrum models

3.I **All – at – once model:** This is the most simplest model which is a single super programme creating and delivering of the fruitful application from begin to end. Hence this model is the fastest way to deliver a product that has good internal architecture consistency and is the implementation of hacker model. IBM is the most popular company whose researchers defined the model with the help of documentation with help of surgical team and considered as the most productive approach in cohesion of distributed scrum.

The next level of this model is handcuffing two programmers together. This showcase's the better delivery of code and provides the best time complexity. During the execution of this model in the distributed environment of the scrum uses the Japanese approach to team – based new product development combined with the simple rules to enhance team. It was been observed some of the companies were already working on the iterative and incremental approach for building software's with great reliability and flexibility. Features were implemented in slices where an entire piece of fully integrated functionality worked at the end of iteration. The idea of building a self – empowered team in which global view of the product caused the team to self-organise the right idea.

3.II **Hyper productivity:** Scrum was been developed to allow average developers to generate high performance teams. Firstly, Scrum achieved a hyper productivity state because of various primary factor.



It represents how the hyper productivity is generated. In second step the implementation will took place and in the last Rapid generation of software system using the scrum framework will be generated. During the execution of this model the complexity which arises was "punctuated equilibrium". It occurs when species are stable for long period of time and then undergoes a sudden jump in capability. Using the scrum with integrated component design leads to unexpected, adaptive properties resembling the process of punctuated equilibrium. This aspect of self – organization is now understood as a type of Set Based Concurrent Engineering. Developers consider sets of the possible solutions and gradually limits possibilities to converge on a final solution. The most evolved component is selected 'just in time' to absorb new functionality which results in minimal coding and more elegant architecture.

IV.DIFFERENT TYPES OF SCRUM

Now moving forward to get some glimpse of variety of scrum used by the researchers and then further been implemented by different organizations.

4.I **First Scrum – Type A:** Some organizations thinks that this type of scrum is useful for education and training on the pace of the scrum and particularly suited to new scrum teams. It creates a loss of time between sprints when the team is recognizing for the next sprint. OOAD tool that incorporated provides six sprints for the first product release and the gap between sprints took at least a week. As a result we could only do 9 sprints a year hence losing 25% productivity as compared to running 12 sprints per year. Each month of delay cost millions of dollar of lost revenue and gave the competition the opportunity to overtake us.

In addition to the loss of productivity it takes time during the sprint for developers to get enough clarity about the user requirements to start coding. This created the adhoc environment between the connectivity of Product owner and Scum Team concerning lack of understanding of what to do next and dissatisfaction on part of the Product Owner with delays in feature delay. It is sometimes used to pilot scrum. It allows systematic application of the scrum process with enough time to refine operations and regroup between sprints.

The benefits of the Type A are: Total focus on iteration in process. Easy implementation, developing and understanding the pace of scrum. Clearly defined iterations. Hence, the issues with Type A are: Loos of time to market. Disruption of pace of scrum because lack of understanding about user experience. Loss of productivity.

4.II **Second Scrum – Type B:** The way to overcome the solution to the issues which were present in the previous type of the scrum is to insert the tasks in current Sprint that work for a subsequent sprint. A minimal specification of the user experience for a feature can be defined prior to the sprint where it is implemented. This will allow sprint to be executed continuously with the product backlog which is defined and ready at the beginning of each sprint.

Regarding the use of specifications there is a significant relationship between the completeness of the functional specification and productivity. Hence there is weak relationship between the completeness of the detailed design specification and defect rate. The outcome of this are more productive to the degree that a complete functional specification exists prior to coding. With the help of Scrum developers use a minimum amount of documentation and do not require completeness of the specification to start Scrum. There is a strong correlation between adequate product specifications and productivity.

This type allows the developer to start their work without false starts enhancing the delivery feature within sprints and improve throughput. The biggest resource bottleneck on a software project typically occurs as shortage of expert developers whose skills are not easily transferable. Scrum teams must decide on what tasks can be implemented in a sprint and who will implement them using a normal work week as the standard way to do business. The key indicators that scrum is working must be visible in Type A before moving to Type B:

If the developers loads the development queue without building a sustainable pace of development will generates the negative impact. On complex development projects it can take a new developer and require additional few months to come up with full productivity. For instance if the turnover is 20%, we are effectively under resource by 20%. If morale drives the pace of development down further that you may cut productivity in half due to dynamic motivation factors.

Conversely, if scrums are running well, functional specifications in the right way in a Type B Scrum will eliminate false starts within Sprint and downtime between sprints. This has more than doubled productivity for some experienced scrum teams.

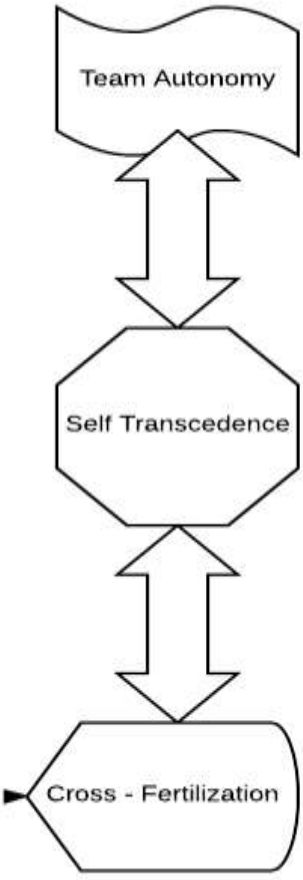


Figure 3Key indicators

4.II.a. **Functional Specifications for Type B sprints:** Maintaining the agility of scrum process requires a minimalist approach to functional specification. Minimum amount of documentation for a product feature may be few pages instead of executing the entire slot. Lesser execution will provide the developer fair

understanding of the user experience. Moving to the Type B scrum which requires analysis and design resources from the development team in order to help the Product Owner create functional specifications. Members of the development team work with Product Owner from the beginning of requirements creation.

The real gain from the Type B is to have Product Backlog fully loaded, prioritized, and ready for the breakdown into Sprint backlog tasks at all times. A developer never wonders what to do next because the queue is always full. If the sprint backlog is automated, team members simply logon at the beginning of the day and self-manage work of the queue in front of them. The Scrum Master is the leader of the Scrum Team which acts as strong technical contributor. Product Owner and Scrum Master continuously work closely to peel the items off the product backlog , initializing breakdown of the product backlog items into sprint tasks and assign tasks to a developer's queue. The developer further decided how to order the work, refines the granularity of tasks and assigns them appropriate team members.

4.II.b. **Product Owner:** Product development Scrum creates a cross functional team that is totally responsible for the product. Product Owner owns the business plan for the product, the functional specification for the product, the product backlog for the product, and prioritization of the product backlog. The member of the Scrum works simultaneously with the Scrum Master to introduce product backlog items into a sprint where they are broken down into tasks by the team for execution as Sprint Backlog.

The best way to visualize scrum responsibilities is to think scrum team as analogous to a high performance car in rally race. Product owner is the navigator and Scrum Master is the driver. The team is the engine, and the drive train. Scrum Master follows the navigational directions of the product owner and drives the car adroitly. At the end of every sprint there is a demo where other players can suggest modifications to improve production in next sprint.

4.II.c **Enables Hyper productive Teams:** When product owner is given accountability for the sprint backlog to build strong product owners with the hands on control of the product feature and function. Firstly, Scrum begins to execute Type B Scrum as they mastered the process. They are able to enter the zone using the techniques where they could deliver faster functionality than the customers. The feeling of power on a development team that can deliver more product than anyone can absorb exhilarating and allows the team to focus on higher goals.

Scrum was designed for hyper productive state to get an ordinary developers to function as a champion team. It only starts happen when the teams shift their gears to the Type B scrum. The doubling of throughput from a team that is already very productive results in an organizational breakthrough.

V.DISTRIBUTED SCRUM

Many small, collocated teams have achieved high productivity of effect. The different organisations of US, Japan outsource software development to Easter Europe, Russia. Typically, remote teams operate independently and communication problems limit productivity. Current Scrum practices is for local scrum teams at all sites to synchronize once a day via scrum meeting. All scrum teams consist of developers from multiple sites. While some agile companies created geographic transparency on a small scale which helps in building a new implementation of platform and system architecture for a 'complex integrated library system'.

Best practices for distributed scrum are the daily scrum team meetings of all developers from multiple sites and also the daily meetings of product owner. It hourly automates one central repository and no distinction between developers at different sites on the same team.

VI.DISTRIBUTED TEAM MODELS

Here we will have clear vision of different types of distributed scrum models which are in practice:

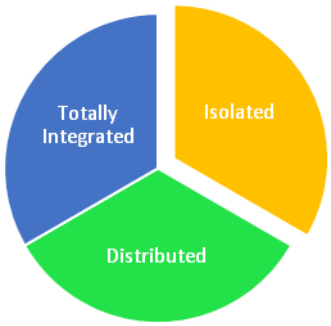


Figure 4 Different distributed models

Isolated Scrum: Teams are isolated across geographies.

Distributed Scrum: Scrum teams are isolated across geographies and integrated by a scrum that meets regularly across geographies.

Totally Integrated Scrum: Scrum teams are cross functional with members distributed across geographies.

Most outsourced development efforts use a degenerative form of the isolated scrum model where outsourced teams are not cross functional and not agile. The latest thinking is the PMBOK model which is degenerative case of isolated non scrums teams. This methodology layers with the UML and RUP onto teams which are cross functional. Distributed models partition the work across cross – functional, isolated teams eliminating the most dependencies between teams.

Integrated 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 help to break down

cultural barriers and disparities in work styles. On large enterprise implementations, it can organize the project into single whole with an integrated global code base. This model is recommended for experienced agile teams at multiple locations.

VII.HIDDEN COST

The hidden cost of outsourcing are significant, beginning with start-up costs. In the remainder costs of transitioning to a new vendor often cancelled out anticipated savings from low labour costs. The average time from evaluating outsourcing to beginning of vendor performance was 18 months.

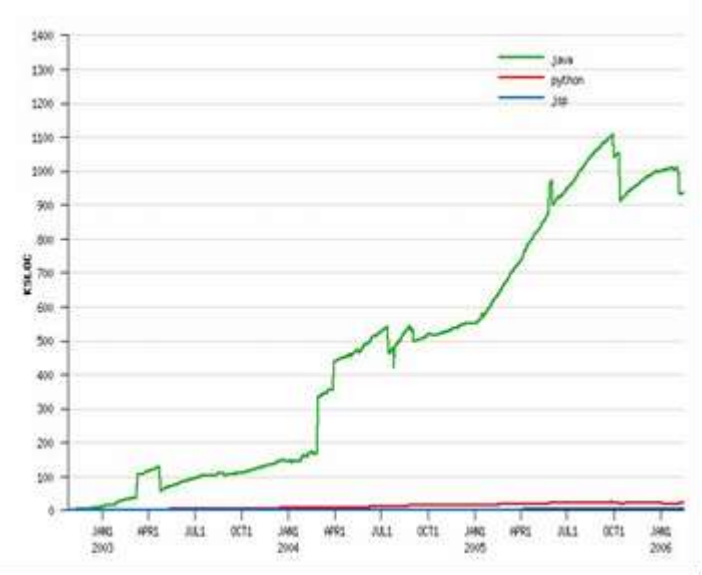


Figure 5Hidden costs

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. The goal of increasing output per team member and linearly increasing overall output by increasing team size were archived.

VIII.INTENT OF INTEGRATED SCRUM

An Agile company building a large product and facing time-to-market pressure needs to quickly double or quadruple productivity within a constrained budget. The local talent pool is not sufficient to expand team size and salary costs are much higher than outsourced teams. On the other hand, outsourcing is only a solution if agile practices are enhanced by capabilities of the outsourced teams. Cost savings are a secondary driver.

IX.FORCES – COMPLEXITY DRIVERS

The Systems and Software Consortium (SSCI) 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. Increasing

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

New product with complete functionality for the library enterprise based on new technologies that were highly scalable, easily expandable, and used the latest computer and library standards.

X.TOP ISSUES RELATED TO DITRIBUTED DEVELOPMENT:

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

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

XI.SOLUTIONS: INTEGRATED SCRUMS

There are three roles in a Scrum: the Product Owner, the Scrum Master, and the Team. SirsiDynix used these roles to solve 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.

For large programs, a Chief Scrum Master to run a Scrum of Scrums and a Chief Product Owner to centrally manage a single consolidated and prioritized product backlog is essential. SirsiDynix located the Scrum of Scrums and the Product Owner teams in Utah.

11.I **Team Formation:** Major challenge for large projects is process management, particularly synchronizing work between sites. This was achieved by splitting teams across sites and fine tuning daily Scrum meetings.

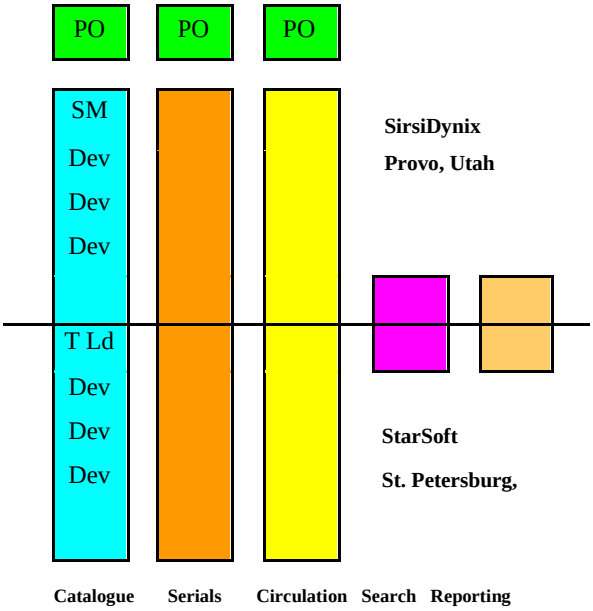


Figure 6 Splitting scrum team across sites

11. II **Scrum meetings:** Teams found it necessary to distribute answers to the three Scrum questions in writing before the Scrum meeting. This shortens the time needed for the join meeting teleconference 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.

Email exchange on the three questions before the daily Scrum teleconference was used throughout the project to enable phone meetings to proceed more smoothly and efficiently. These daily team calls helped the people in Russia and the U.S. learn to understand each other. In contrast, most outsourced development projects do not hold formal daily calls and the communication bridge is never formed. Everyone uses the same process and technologies and daily meetings coordinate activities within the teams.

1. III **Sprints:** There is a Sprint planning meeting similar to 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 some tasks 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 an ideal situation. Sometimes new tasks are discovered after querying Product Owners during the Sprint about feature details. Code is feature complete and demoed at the end of each Sprint. If it met the Product Owner's functional requirement, it was considered done, although full testing was not completed. In the Scrum Framework all activities needed for the implementation of entries from the Scrum Product Backlog are performed within

Sprints (also called 'Iterations'). Sprints are always short: normally about 2-4 weeks. Each Sprint follows a defined process as shown below.

Each Sprint start with two planning sessions to define the content of the Sprint: the WHAT-Meeting and the HOW-Meeting. The combination of these two meeting are also defined as Sprint Planning Meeting. In the WHAT-Meeting the Scrum Team commits to the User Stories from the Scrum Product Backlog and it uses a HOW-Meeting to break the committed User Stories into smaller and concrete tasks. Then implementation begins.

At the end of the Sprint a Sprint Review Meeting is conducted to allow the Scrum Product Owner to check if all of the committed items are complete and implemented correctly. Additionally a Sprint Retrospective Meeting is conducted to check and improve the project execution processes: What was good during the Sprint, what should continue as it is and what should be improved.

XII.EVOLUTION OF TYPE C

It is useful in any context where the activity requires constant change in direction, unforeseen interaction with many participants, and the need to add new tasks as work unfolds. A decision was made to become a platform as well as Application Company by building a software framework and open application programming interfaces (APIs) that would allow integration with many development partners on both the backend and the frontend. A web services platform with a services oriented architecture was selected.

This framework provided open APIs and a software development kit that allowed third party vendors and end users to tightly integrate their mobile applications with other applications already available on a handheld device. The tight integration between software components required similar integration of software development teams internally and externally with partners and offshore developers. This, combined with time to market pressure and rapid growth of new deployments in large enterprises on a monthly basis, forced a new type of Scrum to be implemented.

XIII.SOLUTIONS TO TYPE C FRAME

The Type C Scrum Solution required several innovations that affected all parts of the company. In effect, the company became a Scrum company with all activities tied to an automated data system that reflected release planning and Scrum implementation, as well as installation, support team, and customer feedback.

- A MetaScrum was created to allow company leadership to manage multiple simultaneous product releases.
- Daily Scrum of Scrums became the major Scrum meeting.
- Quality Assurance was reorganized
- Build process ran more frequently and needed to be more robust and easier to fine tune.
- Regression testing automation improved significantly.
- New tools for automated data collection and reporting were developed.
- The company became totally transparent.

All data was available to everyone in real time all the time. Here we describe the company infrastructure required to run a successful Type C Scrum.

XIV.METASCRUM

Managing multiple simultaneous releases of software requires regular review and fine tuning of release schedules. Since every Sprint resulted in a production release of software, Sprints must be carefully coordinated.

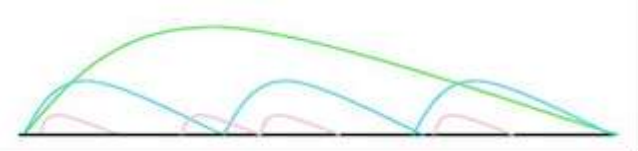


Figure 7 Overlapping of sprints

Weekly Sprints are maintenance fixes or minor enhancements typically generated by issues preventing a customer from going live or causing the system to fail during production. Although these are highly impacted by changing requirements, they are scheduled and committed to customers on fixed dates.

Monthly releases are targeted as a set of customer go-live dates that required specific enhancements for each customer. If a new customer is added to the queue or a customer drops out of the queue, the release might have to be reorganized. Quarterly releases of major functionality can only be completed when weekly and monthly releases are on schedule. Priorities might change during a quarter because of market changes or new customer priorities. A large partner such as Cerner or GE Healthcare could highlight new demands, request acceleration of functionality, or cause delays.

Each release is reviewed. Sprints may be added, changed, or deleted as appropriate. Sales staff must make their case in this meeting for any change in product rollout. Developers must argue for architectural resources. Any company or customer impact is dealt with during the meeting. For example, a change that would directly impact multiple customers will result in an action plan with specific people identified to talk with each customer before the end of the day.

XV.SCRUM TEAM ORGANIZATION

All members of the development team are present for 15 minute meetings. Team leaders do most of the reporting, although any contributor may speak to the following:

- What did each of the six integrated teams complete in the last 24 hours? The Scrum of Scrums leader logs what tasks were completed and sends out an email to the company immediately following the Scrum of Scrums.
- What blocks were found in performing tasks in the last 24 hours? These are logged, reported, and followed-up after the meeting.

- What tasks will be worked on today? Team members volunteer for tasks. The ScrumMaster and the Lead Architect may help bring focus to appropriate tasks.

The Scrum of Scrums meeting takes place at the same time and place every day. An open space was secured by the development team for this purpose. Pair programming is done primarily on tasks with difficult design and coding requirements. Many of the developers stay in the open meeting space for the entire day working together as a group. Innovative and open cube space and a few offices and conference rooms are provided for those who need quiet, focused time.

Every Sprint in the Type C Scrum results in a production code release and all Sprints produce the ultimate demo, i.e. the software goes live and satisfies real customers. The rapid pace of delivery of releases initially created a Quality Assurance (QA) bottleneck. The solution was to assign a small QA team to every release. QA was expanded to four small teams of 2-4 people. This enables them to work with the development team continuously on regression testing and packaging the four top priority releases that are functionality complete while simultaneously doing early testing of developer code on the others. QA is part of the Scrum of Scrums and reports on daily status of ongoing testing.

Thus every Sprint there is a development phase to functionality complete and a packaging phase where the system is regression tested and all critical bugs are eliminated. Developers and QA engineers work closely together from beginning to end of the Sprint. In the packaging phase, developers are focused on eliminating bugs as fast as QA can find them and doing testing when needed. While some view this as a mini-waterfall, it is simply executing the Scrum prime directive – do what makes common sense. Code must be frozen to be regression tested prior to shipment in order to deliver a quality product.

XVI.TOOLS FOR REPORTING AND DATA COLLECTION:

A PERL expert on the development team was assigned to build utilities around GNATS to support Scrum. These were addition of required data items, new queries, minor changes to the user interface, and automated file dumps for management reporting via Excel.

It was decided that sprint tasks would be treated like problem reports. This minimized new data entry requirements and allow tasks and bugs to be packaged together seamlessly for a release. Only three data items were added to GNATS for developer entry:

- Initial estimate
- Days invested
- % complete

The first estimate was fixed at initial entry and could never be changed in order to allow for accurate historical reporting of estimates versus actual time to complete tasks. Two additional data items were added for reporting, purposes. These are automatically calculated from the three items above.

- Days remaining
- Actual time to complete

If the initial estimate is 2 days, for example, and no work has been accomplished, the days remaining are 2 days. If a developer has invested 1 day and states that it is 25% complete, GNATS calculated the days remaining as 3 days. Initial estimates are automatically expanded based on real time data. This approach can be generalized to be used for tracking of development tasks within any bug tracking system. Ideally the tracking system integrates with the code versioning system. Tools such as Trac integrate with the Subversion code versioning system and support comprehensive integration of development, error tracking, and code management. These are current candidates for upgrading the GNATS system.

XVII.RESULTING CONTEXT

The move to a Type C Scrum to improve development productivity had far reaching effects on the company making it more flexible, more decisive, more adaptable, and a better place to work. The same effects commonly seen on Scrum teams were reflected throughout the company.

Project management was totally automated. The result is paperless project management and reporting, largely without human intervention. Scrum execution has become exceptionally efficient and the automated tracking system has become mission critical.

Burn down charts have evolved to frame the entire status of a project on one chart. The chart below instantaneously reflects project state for Release 3.20 at a glance to those familiar with the data. With all tasks entered at 16 hours or less and bug fixes typically less than a day, the aggregate number of tasks can be monitored and downward velocity is highly predictive of delivery date. Information is presented as follows:

Diamond – 320 current open – cumulative work remaining

Triangle – 320 daily closed - items closed by QA each day

Star – 320 total closed - cumulative closed (on scale at right)

Square – 320 current verification - current total in verification (items QA needs to test and close)

X – 320 daily open – new tasks opened per day

The cumulative closed (right scale) is much higher than the starting number of about 160 tasks (left scale). The reason for this is that QA is finding bugs, often generating multiple tasks that can be closed with one developer fix. Product development is adding tasks primarily because of customers moving in and out of the mix for installs. Development is discovering new tasks as they flesh out technical design.

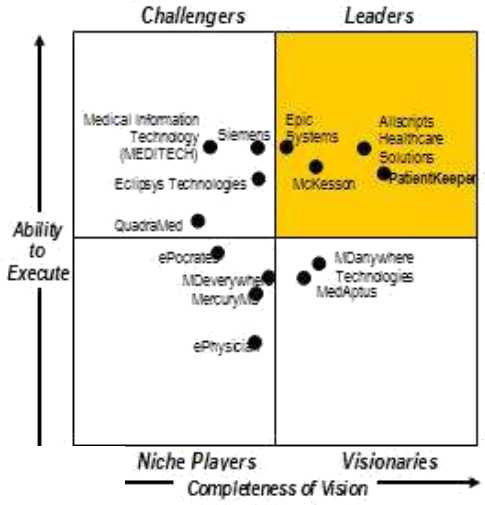


Figure 8 Magic Quadrant

XVIII.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. During the Sprint, the Product Owner tests features that are in the Sprint backlog. Up until 2006, testers received a stable Sprint build only after the Sprint demo. The reason for this was a lower tester/developer ratio than recommended by the Scrum Alliance.

The test-first approach was initially encouraged and not mandated. Tests were written simultaneously with code most of the time. GUIs were not unit tested.

Component	*Test Cases*	*Tested*
Acquisitions	529	384
Binding	802	646
Cataloging	3101	1115
Circulation	3570	1089
Common	0	0
ERM	0	0
Pac Searching	1056	167
Serials	2735	1714
Sub Total	11793	5115

Functional Test Example

Functional Area	Reserve Book Room
Task Description	Check that items from Item List is placed under Reserve with "Inactive" status
Condition	1. User has right for placing Items under Reserve 2. At least one Item List exists in the system 3. Default Reserve Item Status in Session Defaults is set to "Inactive"
Entry Point	Launcher is opened
Test Data	No specific data
Action	1. Reserve > Reserve Item 2. Select "Item Search" icon 3. Select "Item List" in the Combo box list of search options and enter appropriate Item list name 4. Press Enter

XIX.CONCLUSION

This case study is a proof point for the argument that distributed teams and even outsourced teams can be as productive as a small collocated team. This requires excellent implementation of Scrum along with good engineering practices. The entire set of teams must function as a single team with one global build repository, one tracking and reporting tool, and daily meetings across geographies.

Outsourced teams must be highly skilled agile teams and project implementation must enforce geographic transparency with cross-functional teams at remote sites fully integrated with cross-functional teams at the primary site. In the SirsiDynix case, the teams were all run from a central site giving strong central control.

It is highly unlikely that distributed outsourced teams using current Agile Alliance best practices of distributing work to independent Scrum teams across geographies could achieve the level of performance achieved in this

case study. Therefore, SirsiDynix sets a new standard of best practices for distributed and outsourced teams with a previously demonstrated high level of Agile competence.

Moving to a Type C Scrum is not for the faint of heart. It requires Scrum teams that can execute a standard sprint flawlessly, an automated data collection and reporting system that is easy to implement and update, and a corporate culture that embraces change. Going to a Type C Scrum will transform a company into an organization where Scrum becomes mission critical for the entire enterprise, not just software development.

Type C Scrum increases speed of development, aligns individual and corporate objectives, creates a culture driven by performance, supports shareholder value creation, achieves stable and consistent communication of performance at all levels, and enhances individual development and quality of life. It also drives functionality out into the marketplace at a pace that can overwhelm competitors and achieve industry dominance.

REFERENCES

[1] C. G. Langton, "Life at the Edge of Chaos," in Artificial Life II, SFI Studies in the Sciences of Complexity, Held Feb 1990 in Sante Fe, NM, 1992, pp. 41-91.

[2] R. A. Brooks, "Intelligence without representation," Artificial Intelligence, vol. 47, pp. 139-159, 1991.

[3] H. Ziv and D. Richardson, "The Uncertainty Principle in Software Engineering," in submitted to Proceedings of the 19th International Conference on Software Engineering (ICSE'97), 1997.

[4] W. S. Humphrey, A Discipline for Software Engineering: Addison-Wesley, 1995.

[5] P. Wegner, "Why Interaction Is More Powerful Than Algorithms," Communications of the ACM, vol. 40, pp. 80-91, May 1997.

[6] P. DeGrace and L. H. Stahl, Wicked problems, righteous solutions : a catalogue of modern software engineering paradigms. Englewood Cliffs, N.J.: Yourdon Press, 1990. [7] Matisse Software, "The Emergence of the Object-SQL Database," Mountain View, CA 2003.

[8] F. P. Brooks, The Mythical Man Month: Essays on Software Engineering: Addison-Wesley, 1995.

[9] K. Beck, Extreme Programming Explained: Embrace Change. Boston: Addison-Wesley, 1999.

[10] W. A. Wood and W. L. Kleb, "Exploring XP for Scientific Research," IEEE Software, vol. 20, pp. 30-36, May/June 2003.

[14] M. Beedle, M. Devos, Y. Sharon, K. Schwaber, and J. Sutherland, "Scrum: A Pattern Language for Hyperproductive Software Development," in Pattern Languages of Program Design. vol. 4, N. Harrison, Ed. Boston: Addison-Wesley, 1999, pp. 637-651.

[15] S. J. Gould, The structure of evolutionary theory. Cambridge, Mass.: Belknap Press of Harvard University Press, 2002.

[16] W. D. Hillis, The Connection Machine. Cambridge, MA: MIT Press, 1985.

[17] S. Levy, Artificial Life : A Report from the Frontier Where Computers Meet Biology, 1st ed. New York: Vintage, Reprint edition, 1993.

[18] D. K. I. Sobek, A. C. Ward, and J. K. Liker, "Toyota's Principles of Set-Based Concurrent Engineering," Sloan Management Review, vol. 40, pp. 67-83, 1999.

[19] J. Sutherland, "Agile Can Scale: Inventing and Reinventing Scrum in Five Companies," Cutter IT Journal, vol. 14, pp. 5-11, 2001.

[20] C. Jakobson, "The Magic Potion for Code Warriors! Maintaining CMMI Level 5 Certification With Scrum.," J. Sutherland, Ed. Aarhus, Denmark: Agile 2007 paper in preparation, 2006.

[21] H. Takeuchi and I. Nonaka, Hitotsubashi on Knowledge Management. Singapore: John Wiley & Sons (Asia), 2004.

[22] K. E. Nidiffer and D. Dolan, "Evolving Distributed Project Management," IEEE Software, vol. 22, pp. 63-72, Sep/Oct 2005.

[23] R. Zanoni and J. L. N. Audy, "Projected Management Model for Physically Distributed Software Development Environment," in HICSS'03, Hawaii, 2003, p. 294. [24] M. Poppendieck, "A History of Lean: From Manufacturing to Software Development," in JAOO Conference, Aarhus, Denmark, 2005.

[25] J. Barthelemy, "The Hidden Costs of Outsourcing," MITSloan Management Review, vol. 42, pp. 60-69, Spring 2001.

[26] B. Gorzig and A. Stephan, "Outsourcing and Firm-level Performance," German Institute of Economic Research October 2002.

[27] StandishGroup, "2003 Chaos Chronicles," The Standish Group International, 2003.

[28] J. Sutherland, "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects with Details on Scrum Type C Tools and Techniques," PatientKeeper, Inc., Brighton, MA May 30 2005.

[29] M. Cohn, User Stories Applied : For Agile Software Development: Addison-Wesley, 2004.

[30] C. Jones, "Programming Languages Table, Release 8.2," Software Productivity Research, Burlington, MA 1996.

[31] C. Jones, Software assessments, benchmarks, and best practices / Capers Jones. Boston, Mass.: Addison Wesley, 2000.