Master's thesis

Business Information Systems

2014

Karl-Johan Spiik

SHAREPOINT AND SCRUM

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Systems

2014 | 69

Tuomo Helo

Karl-Johan Spiik

SHAREPOINT AND SCRUM

SharePoint is a development platform that can be used as such once it has been installed. It is efficient tool to develop especially Intranets and Extranets as well as Internet portals. Scrum is an agile project method that presents what kind of meetings and roles there should be in a project. Scrum leads the development team to design their own work, be visible to customer, and estimate their speed of doing.

SharePoint complexity and Scrum method might work together. The thesis introduces theory and practice. There are one Intranet case and then there is description how the SharePoint Scrum process should go. Behind this thesis work is 7 years of experience in SharePoint development and 5 years of experience doing SharePoint projects with Scrum.

This thesis work introduces the special features of SharePoint and Scrum and their possible challenges. The thesis concludes with solutions to the challenges and recommendations for combining SharePoint and Scrum.

KEYWORDS:

SharePoint, Scrum, Agile, Microsoft, Software development

OPINNÄYTETYÖ (YAMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Business Information Systems

Kevät 2014 | 69

Tuomo Helo

Karl-Johan Spiik

SHAREPOINT JA SCRUM

SharePoint on kehitysalusta, jota voidaan käyttää myös sellaisenaan heti asennuksen jälkeen. Se on mahtava työkalu Intranet ja Extranet sovelluksiin, mutta soveltuu myös Internet portaaleihin. Scrum on ketterä ongelmistokehitys projektimuoto, joka esittelee mitä kokouksia ja rooleja projektissa tulee olla. Scrum ohjaa kehitystiimiä suunnittelemaan oman työnsä, tekemään siitä läpinäkyvää asiakkaalle ja arvioimaan oman työnsä nopeuden.

SharePointin monimutkaisuus ja Scrum metodi voivat toimia yhteen. Tämä työ esittelee teorian ja käytännön. Tästä työstä löytyy yksi Intranet esimerkki ja kuvaus siitä miten SharePoint Scrum prosessin kuuluu mennä. Työn taustalla on seitsemän vuoden kokemus SharePoint kehityksestä ja viiden vuoden kokemus tehtynä Scrum metodilla.

Työ esittelee SharePointin ja Scrumin erityisominaisuudet. Työssä esiintyy myös haasteita ja ratkaisuja miten käyttää SharePointia ja Scrumia yhdessä.

ASIASANAT:

SharePoint, Scrum, Agile, Microsoft, Ohjelmistokehitys

CONTENTS

LIST OF ABBREVIATIONS (OR) SYMBOLS

SharePoint	Microsoft Office family product
Out-of-the-box (OOTB)	Feature that SharePoint offers by default. After installation of medias one can use this feature without any customizations
TPM	Traditional project management, like waterfall method.
APM	Agile project management, like Scrum
Waterfall method	Project management method where there is only one cycle starting from defining ending to product release
Traditional project	Project done with waterfall method
PMLC	Project management lifecycle method
SM	Scrum master
ROI	Return of investment
Sprint	Iteration cycle in Scrum
PSP	Probably shippable product which delivered after each sprint
DOD	Definition of Done – a list of actions which must be done before a task is ready
PowerShell	MS-DOS style command prompt that run also C# code. It is a tool for configuring servers and SharePoint.
API	Application programming interface
ECM	Enterprise Content Management
SharePoint Online	Office 365 family product. SharePoint that is in cloud
On-Premises	SharePoint that runs on local server farm
UI	User Interface
AD	Active Directory, Microsoft authentication module that stores the user accounts
Result Source	Search component in SharePoint that narrows down the result source
Web Part	Small module in a web page that has some functionality and can be moved from one place to another or copied
Web Template	All sites in SharePoint are based on site definitions. Web template defines what lists and libraries a site contains and what is on the front page

Continuous integration	The staging environment is continuously building and creating itself to the latest version of the solution
Search Service Application	SharePoint component that takes care of search and all of its functionalities
PBL	Product Backlog
PBL item	Product Backlog item a.k.a. feature
Content type	SharePoint module that gathers together site columns
Field	Site column of column in content type or list
Receiver	Code block that is associated to site, list, field or content type

1 INTRODUCTION

SharePoint is a Microsoft product which is also a platform for software development. Scrum is a framework for effective team collaboration in projects. Scrum is one of the most familiar agile software development methods.

The goal of this thesis is to investigate working methods in Affecto EIM team and compare the results to theory and find out special characteristics and challenges of mixing SharePoint and Scrum. SharePoint development differs from traditional software development and that's why this thesis focuses also on the technical product. I will have one example project. The example is a lifecycle of an Intranet project from defining to final release.

The author of this thesis an IT BBA and this is his MBA master's thesis. I have worked in Affecto since 2007 and currently my title is senior developer. My main job is to design and implement SharePoint solutions in projects. We are not doing product development, we do customer projects. The author of the thesis is also involved in other phases in projects like defining, testing and educating. The author has on his responsibility to guide couple of trainees and teach them SharePoint. The author works as Scrum Master in projects he does not implement. The author has worked with Scrum since 2010.

The author maintains his technical knowledge by educating himself in the field of SharePoint and Scrum. The author has Scrum Master Certificate and for SharePoint he has all SharePoint certificates on SharePoint 2007 and SharePoint 2010. The author has succeeded two certificates on SharePoint 2013 and will continue to complete them all in SharePoint 2013.

This thesis introduces what is SharePoint and what you can do with it. This thesis will bring out the technical background, characteristics and the power of modularized product. This thesis will present what are different methods for software project and when should you choose Scrum. This thesis demonstrates the whole Scrum and the idea behind it. This thesis will tell about cycles, scopes, roles and

all the thing you need to know about Scrum to have a complete picture of its characteristics.

After the author have outlined SharePoint and Scrum theory and technical information he will introduce a typical SharePoint project which is an Intranet project. The author will go through different phases of the projects and tell how the production typically happens. After that the author will tell about the typical SharePoint development situations, installing phase upgrading process by referring the presented case.

Chapter two introduces the differences between traditional and agile project management and the theory of Scrum. Chapter three introduces SharePoint as a technical and functional product. Chapter four describes the special features of SharePoint and presents and specifies habits, needs and customs met on SharePoint development in Scrum projects. Chapter five contains an example of one SharePoint Intranet project. Chapter six presents the recommended process based on discovered features, challenges and situations in Chapters four and five combined to the theory of chapters two and three. The final chapters contains conclusions and recommendations.

2 WHAT IS SCRUM

2.1 Agile project methods

Traditional project management (TPM) acquired its current form in the 1950s. This method did not fit for IT projects. Endless trail of changes and reworks when client did not get what they needed. (Wysocki 2011, 340)

Project management always has a lifecycle which contains five process groups: scoping, planning, launching, monitoring and controlling, and closing to accomplish the goal of the project. Each process group is included at least once and can be repeated so many times that some requirement is met. In the figure 1 you can see how to choose the method. The method gives us guidelines how to set these process groups and those are called project management lifecycle methods (PMLC). (Wysocki 2011, 341-343)

The difference between traditional (TPM) and agile project management methods are that TPM works when the goal and the technology (solution) for the project are known and clear whereas in agile the solution is blurry. (Wysocki 2011, 34)

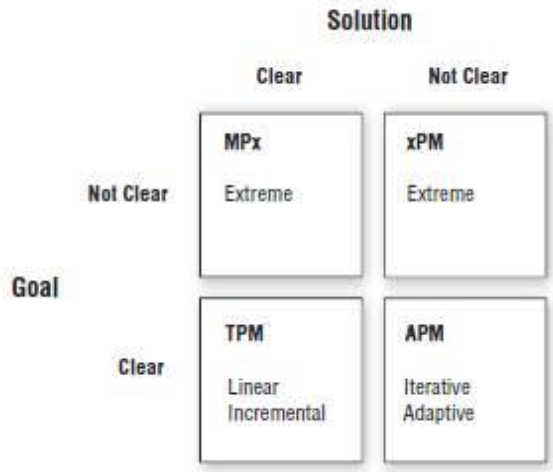


Figure 1: The four quadrants of the project landscape (Wysocki 2011, 34)

Figure 1 displays the four quadrants or the project landscape. When the solution is not clear should the chosen method for the project be Agile (APM). When neither the goal nor the solution is clear, the Extreme project management (xPM) method should be chosen. And when the goal is clear but the solution is not, the chosen method is called MPx. Research and development (R&D) projects are always MPx projects because the technique is chosen and explored without proper knowledge of the goal (Wysocki 2011, 34-36).

TPM is a linear method and suits for incremental models. APM is iterative and suitable for adaptive models. MPx and xPM are extreme models. According to Wysocki extreme models are projects without a clear goal. TPM is the simplest model and usually its process groups goes in this order: scope, plan, launch, monitor & control and in the end close the project. In incremental model the small parts are being delivered and added to the initial release to form a more complete solution. To use this incremental PMLC solution instead of linear is a market-driven decision. In both models the complete solution is the project outset. Seeing the incomplete or partial solution during project has both advantages and disadvantages. (Wysocki 2011, 34-36, 341-36)

According to Kendrick, when next project is very similar than the previous, the project manager should use linear model and waterfall method. On the other hand, when the project has less similarity one should use iterative methods and go step-by-step in the projects. (Kendrick 2010)

Agile model has the same process groups but the biggest difference is the repeating of them until the project is done. Scope is fixed but each round has the project groups plan, launch, monitoring & control and closing. After each round the scope keeps but new round has started to deliver a new part of the project. Agile methods use teams less than 30 professionals and they must be located near to each other or have good communication tools. The iterative PMLC method splits the project goal into small pieces. This gets us to plan the whole iterations and repeat them until the goal is reached. Each iteration is part of the big picture but they are specified during the project not before it as in TPM incremental PMLC. (Wysocki 2011, 383-397)

“The iterative PMLC model requires a solution that identifies the requirements at the function level but might be missing some of the details at the feature level.” Functionality is being built into the solution when the knowledge of possibilities and solution is coming to light. Project manager haven’t chose the incremental model because there will be some scope change requests. (Wysocki 2011, 385)

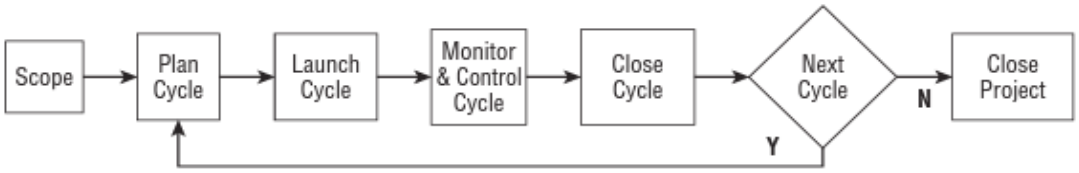


Figure 2: Adaptive PMLC model (Wysocki 2011, 399)

The extreme end of the APM is the adaptive PMLC model which is presented in figure 2. In adaptive PMLC model each cycle knows less of the solution itself. Here all the cycles cover up the blurry fields of the project or its specifications. The cycles may not be connected at all. In incremental the cycles completes each other. Iterative model splits the project into cycles. Adaptive model cycles collects together the uncertain or blurry parts making the project. That is what meant that each cycle knows less of each other. (Wysocki 2011, 398-400)

2.2 Scrum in a nutshell

Scrum is a way to develop software piece by piece so that after each cycle there is Probably Shippable Product (PSP). Figure 3 displays that there will be working piece of software doing something after every cycle. The cycles in scrum are called Sprints. Before each Sprint the development team and the customer meet and choose features to be developed in the next Sprint. After this the development team breaks the features into smaller pieces, tasks. All the features and tasks are prioritized so they are in a linear queue. Development team starts to implement tasks from the top of the list. The development team finishes features

chosen to this Sprint one by one. After the Sprint the new functionality is demonstrated to customer. All this is demonstrated in figure 3. (Schwaber & Sutherland 2011, 3 – 4 | Sutherland 2010, 11)

Customer sees the evolution of the developed software during the project. Figure 3 demonstrates how every one to four weeks customer sees how the software works and looks like. If there is something to be changed or some features seems to be useless, the scrum team can change the features to be developed in next Sprint. This is how the customer can change the course of the ship during the project to achieve a goal that fulfils his needs rather than get something that were defined a year ago like in Waterfall method. (Schwaber & Sutherland 2011, 7, 10 - 13)

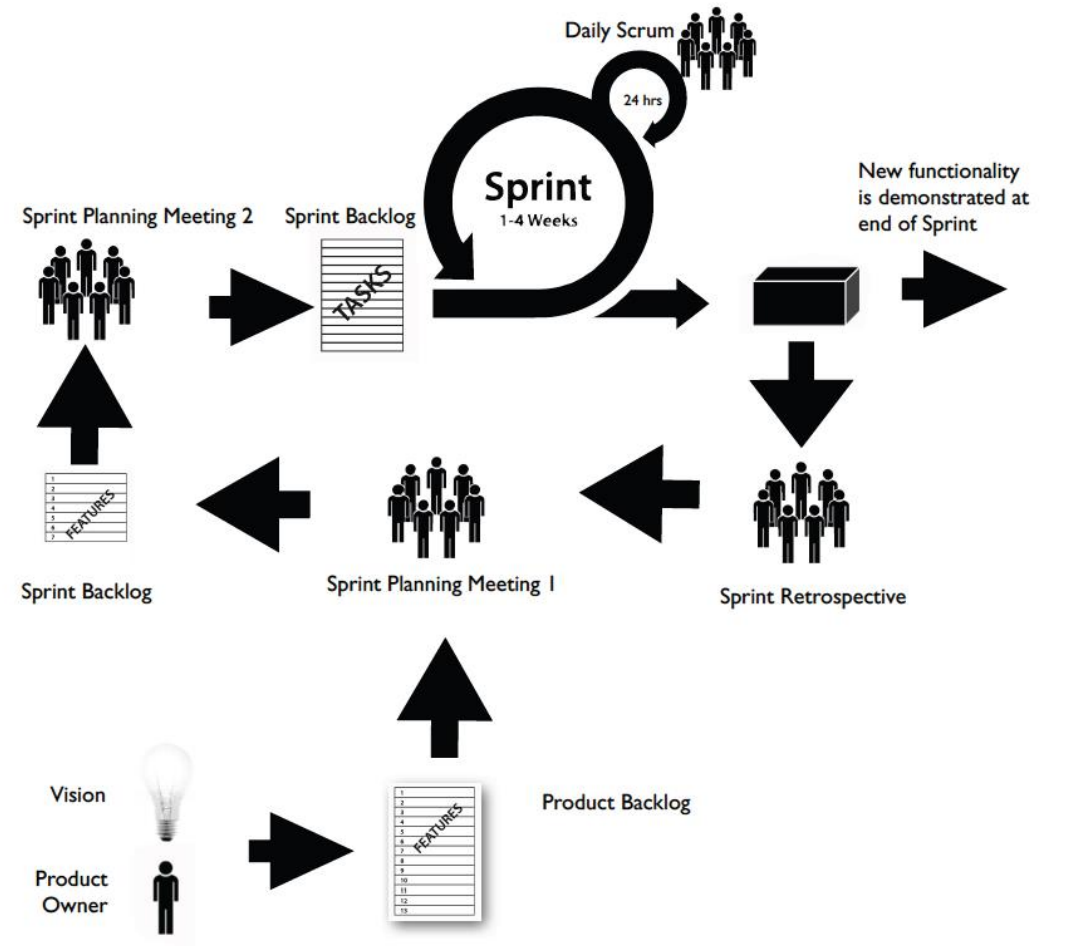


Figure 3: Scrum in a figure (Sutherland 2010, 11)

2.3 People and roles

Scrum team consists of Product Owner, ScrumMaster and development team. According to Kendrick project success depends on effective leadership and committed members of the project. Complex projects rely on teams drawn from several functions and this is why teams should be cross-functional. (Kendrick 2010) Development team should be gathered together of developers who varies from each other on technical skills. According to Scrum every member of development team must be able to do every sprint tasks so that there are no personified tasks. But there is more synergy when a team is heterogeneous and gathered together from different parts of organization. (Pham & Pham 2012, 6)

ScrumMaster is not a project manager. ScrumMaster is the person who fixes everything but won't take part to the actual development. His duty is to be invisible and look after that development team and Product Owner can communicate together. ScrumMaster serves and guides the Team. ScrumMaster must also guide the Product Owner to use of Scrum. ScrumMaster should be full-time ScrumMaster. Sometimes it is not possible to have fulltime ScrumMaster so he has to be in the both roles of member of the Team and ScrumMaster. (Sutherland 2010, 16 | Schwaber & Sutherland 2011, 6)

ScrumMaster is the one who looks after that the meetings are arranged and the Team participates them. ScrumMaster can arrange all meetings or just look after that someone arranges them. ScrumMaster is responsible if there is a meeting and it is missed or people are not called into it. One of the most important work of ScrumMaster is to look that the Team can work in peace during sprint and no one changes or adds the features for current Sprint. ScrumMaster must have knowledge of scrum, be able to serve, know about organization, be good at communication and be good at solving conflicts. Other needed qualities of great ScrumMaster is presented in figure 4. (Sutherland 2010, 16; Pham & Pham 2012, 173 - 176 | Schwaber & Sutherland 2011, 6)

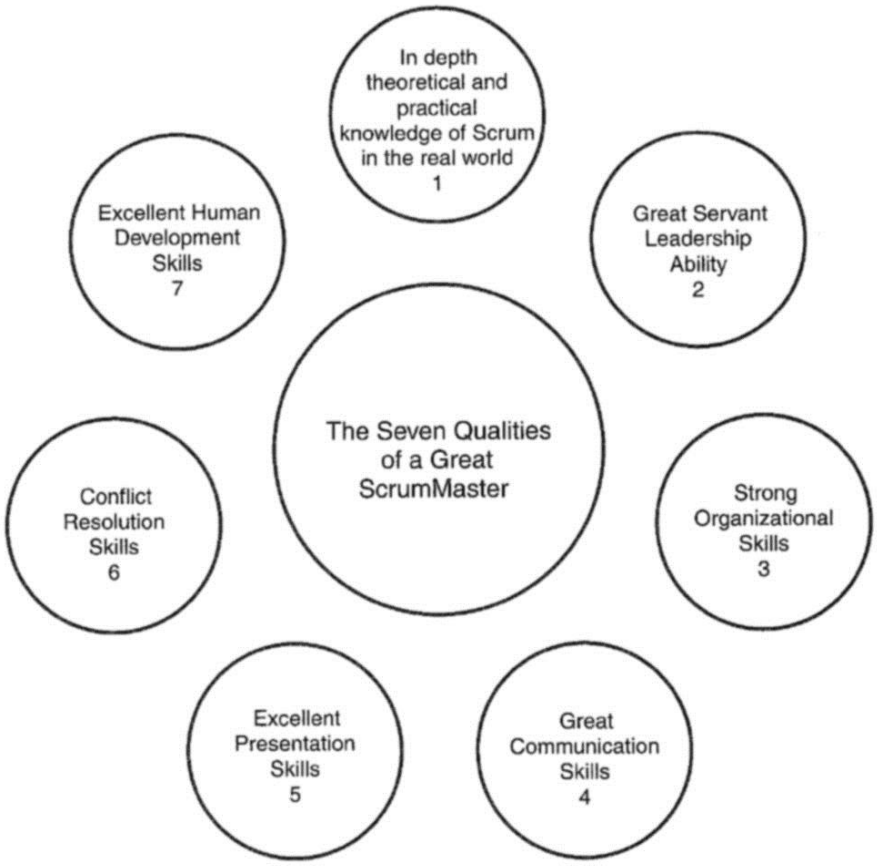


Figure 4: The seven qualities of a ScrumMaster (Pham & Pham 2012, 174)

Product Owner is the person who knows about the solution to be developed. His responsibility is to change all the wishes and requirements into reasonably understandable language that development team understands. It means he ensures that developed features forms into a prioritized list called product backlog. The Product Owner chooses backlog items into each Sprint and keeps the vision and knowledge of the solution clear in the mind. Product Owner should be available on daily basis for refining the backlog. (Sutherland 2010, 16; Pham & Pham 2012, 107 - 112 | Schwaber & Sutherland 2011, 4 - 5)

On company point of view Product Owner's most important job is to maximize return on investment (ROI). He must choose items that has the highest business value and lowest cost and keep the backlog up-to-date all the time. He is the final authority what comes on solution to be developed. Product Owner can be the

customer but if he is, he must be a well-informed one. (Sutherland 2010, 14 | Schwaber & Sutherland 2011, 4)

The qualities of great Product Owner are demonstrated in figure 5. Great Product Owner are available to answer to the questions of the Team. Great Product Owner is good communicator, organizer and servant leader. He understand stakeholders and can internalize the vision of the project and turn it into a good Product Backlog. (Pham & Pham 2012, 108)

Figure 5: The seven qualities of a Product Owner (Pham & Pham 2012, 108)

The (development) Team is the core of Scrum. The Team builds the software and has the cross-functional expertise to deliver the potentially shippable product after each Sprint. The Team is self-organizing. The members in the Team can decide together how to do things inside the Team. The Team can impose rules for itself as much as it wants. The Team has no leader and the members must be

dedicated. The Team is more stable if it can focus on one project at time. (Sutherland 2010, 15 | Schwaber & Sutherland 2011, 5)

The responsibility of the Team is to specify, design, build, test and deliver the probably shippable product. This means that the task list and communication among the Team must be clear and straightforward. The Team makes the Definition of Done (DOD) for the tasks. DOD must be continuously followed and developed. The Team calculates its velocity with the help of ScrumMaster and Sprint burndown charts. Velocity is the amount of estimated product backlog items that the Team can do in one Sprint. After Product Backlog items are chosen into Sprint, the Team splits them into tasks and place them into list called Sprint Backlog. The Team also helps Product Owner to split and refine items in Product Backlog. With the information of velocity and estimates the Product Owner can decide which items should be placed into Sprints and when to plan a release. (Pham & Pham 2012, 97 - 106 | Schwaber & Sutherland 2011, 5)

2.4 Product and sprint backlog

Product Backlog is a prioritized list of features that Product Owner have chosen to be developed in the project. Example of this is presented in figure 6. The items in it are ranked in order of value to the customers' business. The highest value items are at the top of the list. Product Owners responsibility is to update and maintain the Product Backlog during project. At start in the Product Backlog could be only one item i.e. "Create Intranet". The one item must be split into smaller items again and again until the items are ready to be placed into Sprint. Product Owner must remain the vision all the time. (Sutherland 2010, 10, 18 | Schwaber & Sutherland 2011, 11)

					New Estimates of Effort Remaining as of Sprint...					
Item	Details (wiki URL)	Priority	Estimate of Value	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart (see UI sketches on wiki page)	...	1	7	5						
As a buyer, I want to remove a book in a shopping cart	...	2	6	2						
Improve transaction processing performance (see target performance metrics on wiki)	...	3	6	13						
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	...	4	6	20						
Upgrade all servers to Apache 2.2.3	...	5	5	13						
Diagnose and fix the order processing script errors (bugzilla ID 14823)	...	6	2	3						
As a shopper, I want to create and save a wish list	...	7	7	40						
As a shopper, I want to to add or delete items on my wish list	...	8	4	20						

The Product Backlog leads the way ahead for the Scrum Team. It is maintained by the Product Owner.

Figure 6: An example of Product Backlog (Sutherland 2010, 18)

Items in the Product Backlog can vary much from each other. According to Sutherland the items are primarily new customer features ("*enable all users to place book in shopping cart"*), but also engineering improvement goals ("*rework the transaction processing module to make it scalable*"), exploratory or research work ("*investigate solutions for speeding up credit card validation*"), performance and security requirements, and, possibly, known defects ("*diagnose and fix the order processing script errors*"), if there are only a few problems. The requirements in the items are called "user stories" which lively tells how the solution or feature should work. (Sutherland 2010, 19)

It can be difficult to get stakeholders requirements into a form of a user story. Product Owner can use visual requirements gathering process to build the items. One way to identify goals is SMART rules which are

- Specific: Everyone will have the same understanding as to what the goals are (Pham & Pham 2012, 48)
- Measurable: We can objectively determine if the goals have been reached (Pham & Pham 2012, 48)

- Achievable: The stakeholders agree as to what the goals are (Pham & Pham 2012, 49)
- Realistic: We shall be able to achieve the goals for the project with the resources we have (Pham & Pham 2012, 49)
- Time-Based: We will be given enough time to achieve the goals (Pham & Pham 2012, 49)

With the help of these rules Product Owner starts to divide the vision into smaller pieces. After the process the vision of the project is divided into smaller parts and the goal is formed into stories. The items ready for development when refining and splitting the requirements are on that level that the items can be go through with the Scrum Team. (Pham & Pham 2012, 47 - 52)

After choosing Product Backlog items to a Sprint the Team broke down them into a set of individual tasks. These tasks follow the prioritization of the Product Backlog items and are placed into technical order for development. Sprint Backlog is the task list for the Team during a Sprint. Example of this is presented in figure 7. The Team decides the ordering of Sprint Backlog tasks to maximize the velocity of production and quality of “done” functionality. (Sutherland 2010, 10, 22 | Schwaber & Sutherland 2011, 13)

				New Estimates of Effort Remaining as of Day...					
Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart	modify database		5						
	create webpage (UI)		8						
	create webpage (Javascript logic)		13						
	write automated acceptance tests		13						
	update buyer help webpage		3						
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5						
	complete machine order for pRank		8						
	change DCP and reader to use pRank http API		13						

Figure 7: An example of a Sprint Backlog (Sutherland 1010, 22)

Usually one task is for one person in the team. Every day the team members estimates the remaining hours of the tasks and updates them into Sprint Backlog. Someone of the team adds up the remaining work of the Sprint and plots it on the Sprint Burndown chart. Example of Sprint Burndown chart is presented in figure 8. The goal is to get to the point when there is “zero effort remaining” in the end of the Sprint. This chart shows the progress and can be used for calculating the velocity of the team afterwards. (Sutherland 2010, 10, 26).

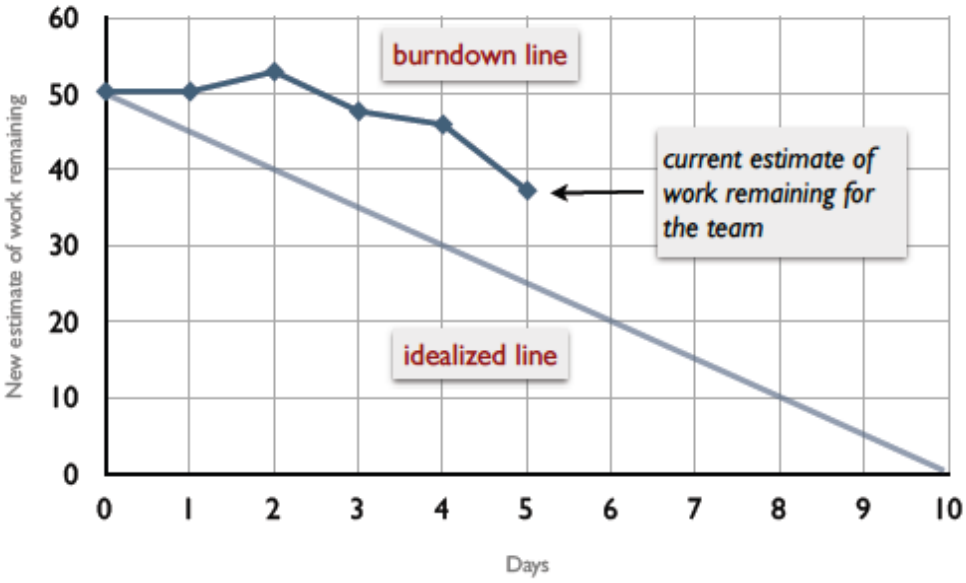


Figure 8: An example of a Sprint Burndown chart (Sutherland 2010, 25)

The team admins the Sprint Backlog and can add or delete tasks in it. Sometimes technical challenges require to add tasks during a Sprint. Noticing a bug or changed technology of platform can be examples of situations when there must be tasks added. Sometimes tasks can marked directly to done state if no actions are required. Sprint Backlog is maintained by the team and this gives much motivation when members of the team can design their own work. (Schwaber & Sutherland 2011, 13)

2.5 Meetings

Sprint planning is the meeting when Product Owner and the Scrum Team (with facilitation from the ScrumMaster) review the Product Backlog. The goal of this meeting is to choose features a.k.a. Product Backlog Items to commit to complete by the end of the Sprint. The items are chosen in the top of the Product Backlog in Sprint Planning Part One. The Product Owner and the Team review the “Definition of Done” that all items must meet, such as, “Done means coded to standards, reviewed, and implemented with unit test-driven development (TDD), tested with 100 percent test automation, integrated, and documented.” (Sutherland 2010, 20 | Schwaber & Sutherland 2011, 8, 14)

Sprint Planning Part Two takes place among the Team. This is more technical planning and each Product Backlog item are spit into individual tasks which are placed into Sprint Backlog. Sprint backlog order goes along the prioritization of chosen items from Product Backlog. After this planning the Team has estimates for work to be done in the ongoing Sprint. (Sutherland 2010, 21). If the work load seems to be too big or the solutions are dependent on other parts of the system of the concept can be split into two or more Sprints and releases (Pham & Pham 2012, 98 - 99 | Schwaber & Sutherland 2011, 9).

Daily Scrum is a meeting that is held every day. This meeting is also called Daily Standup. It should last at max 15 minutes and should be done while standing. Everyone in team attends to this meeting and it is for the Team not for the ScrumMaster. Each member of this meeting tells what he have done yesterday, what he will do today and do he have any problems that prevents their work. (Sutherland 2010, 24). If the Team cannot meet physically or hold the meeting daily, the Team must be creative. Online meetings can be arranged and the information about Daily Scrum contents should be shared among the Team. (Pham & Pham 2012, 150 | Schwaber & Sutherland 2011, 9 -10)

Each member answer to these three question in each daily Scrum

1. What did you do yesterday? (Pham & Pham 2012, 104)
   What they were able to get done since the last meeting (Sutherland 2010, 24).

2. What do you plan on doing tomorrow? (Pham & Pham 2012, 104)
   What they are planning to finish by the next meeting (Sutherland 2010, 24).

3. What prevents you from making progress? (Pham & Pham 2012, 104)
   Any blocks or impediments that are in their way (Sutherland 2010, 24).

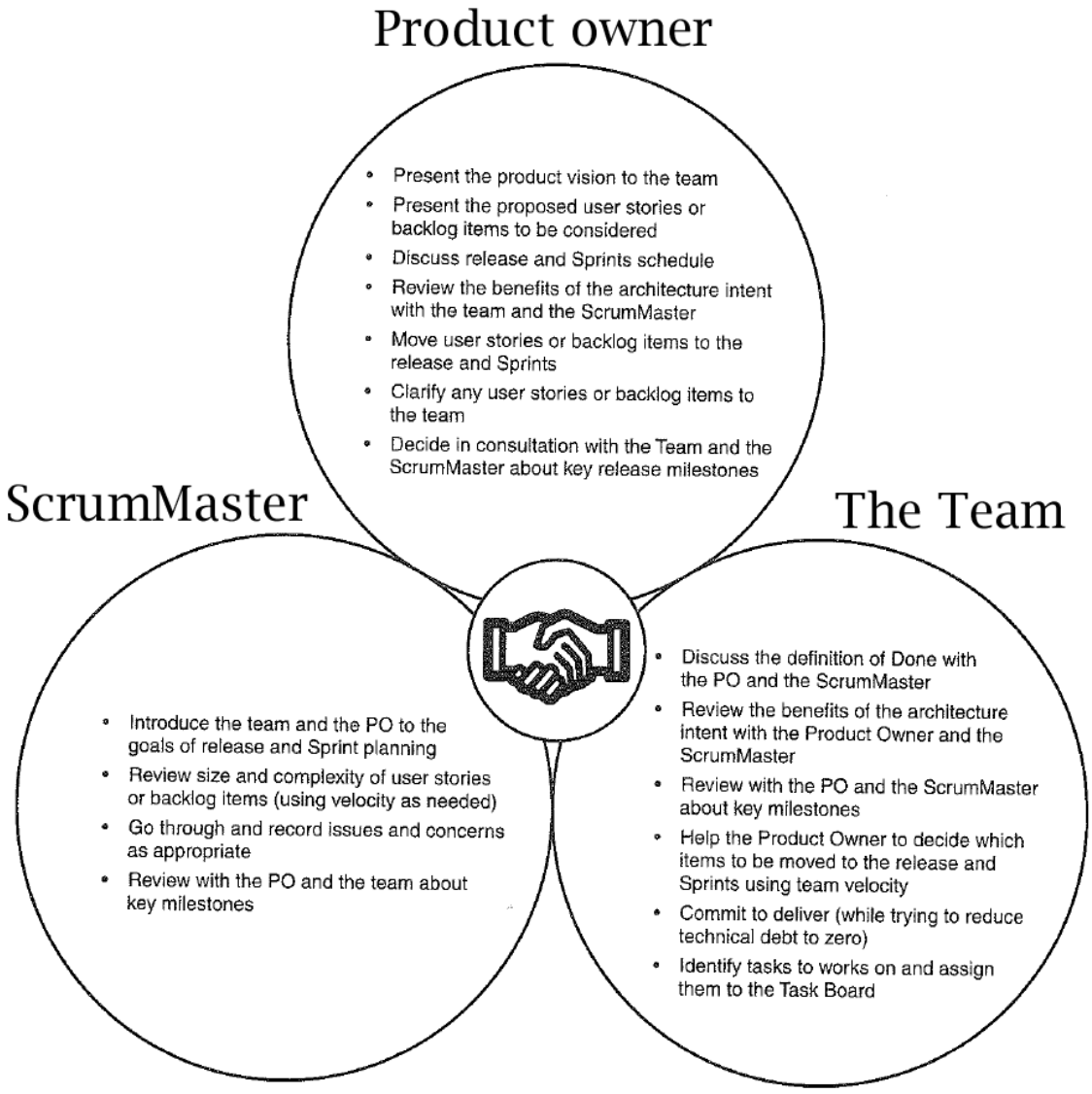


Figure 9: Collaboration for release and sprint planning (Pham & Pham 2010, 105)

Pham & Pham also suggests that there should be two more questions added to Daily Scrum so that the sub-teams can be easily synchronized. These questions asks what can one do to help ones sub-team and what are one doing that slows downs ones sub-team. Responsibilities in Scrum team are provided in figure 9. (Pham & Pham 2010, 104)

In the Sprint Review meeting the Team and the Product Owner meet. This meeting is facilitated by ScrumMaster. First there will be discussion which Product Backlog items have been done and which are not. Then the Team demonstrates what was built. This created solution is also known as the Probably Shippable Product (PSP). (Pham & Pham 2010, 10). This meeting is more like demo than discussion. Present at this meeting could also be customers, stakeholder, experts, executives and anyone else interested. (Sutherland 2010, 28 | Schwaber & Sutherland 2011, 10)

Sprint Retrospective is a meeting when the Team looks back the past Sprint and inspects it. Skipping this meeting is unacceptable. Self-organization requires always reflection. Retrospective gives visibility to Scrum and makes improvement possible. This meetings gives time to the Team to discuss what worked and what did not worked during the past Sprint. (Sutherland 2010, 30 | Schwaber & Sutherland 2011, 11). The meeting should be documented. Usually the Team lists things that was good and need to be maintained and what did not work and need improvement. Then the Team chooses one or two things to improve in the next Sprint. Only one or two things should be chosen so that the goal would be achievable, not to try fix all the problems at once.

Product Backlog Refinement should be organized weekly and the timeframe should be kept. If refining of item is in progress when the time goes out, should this item be refined as first in on the next refinement meeting. The meaning of this meeting is to detail requirements of Product Backlog items and split them into smaller ones. In the meeting could be also estimating the size of Product Backlog items. (Sutherland 2010, 27). This is the meeting where the Team and Product Owner specifies what to be done and goes through the Product Backlog so that Product Owner can manage the backlog and understand the value of the items.

3 WHAT IS SHAREPOINT

3.1 Introduction

SharePoint is a collaboration and publishing product that has widen its area for example to BI from ECM and CMS what is was at the beginning. SharePoint is mainly used for portal software like Intranet, Extranet and Internet. Usually Extranet and Intranet solutions consists document management and possible connections to other systems. SharePoint stands on Microsoft Windows platform and uses Microsoft SQL Server for database. (Pattison, Connel, Hillier, Mann 2011)

SharePoint is a content management system (CMS). CMS is designed to store unstructured data and manage it by setting metadata to the data. With the metadata, the system can retrieve and show right kind of data to users in wanted sites. SharePoint includes both storage and retrieval of the data. It is web-based so it can combine visual layout to data and provide the output in a browser. End user don't have to have client program and updates need only update on servers. (Microsoft 2010, p2-8)

SharePoint is managed through Central Administration (CA). The content itself is located in sites which are located on web applications. Multiple service applications provide services, for example, user profile, managed metadata or search for sites and users. Each site has a HTTP address and is used through a browser. Only the developers and maintenance persons are using PowerShell to configure and manage SharePoint. (Pattison, Connel, Hillier, Mann 2011, 7-8, 10, 24)

SharePoint also runs in Internet as part of the Office 365 product family. The SharePoint 2013 version introduced the App model and the App Catalog. These Applications are more suitable for Internet use and have changed the developing with SharePoint more versatile than before. There are more security problems when the code is in cloud services and shared outside the company itself. SharePoint upgrade method have changed so that it is easier to upgrade cloud services than on-premises installations. SharePoint developer must acknowledge the

whole target segment when developing custom features to SharePoint. (Hillier, Pattison 2013, 1-6)

3.2 Technical background

Technically SharePoint is built with C# and XML and runs on Internet Information Services (IIS) on Windows and .NET Framework. SharePoint is usually installed in server farms where servers are load balanced and servers have different roles. This is called on-premises install. SharePoint is web-based and offers templates to create sites, lists and libraries to store and organize content. Templates are used to create both new websites and various elements inside a website, such as lists, pages, and Web Parts. (Pattison, Connel, Hillier, Mann 2011, 3)

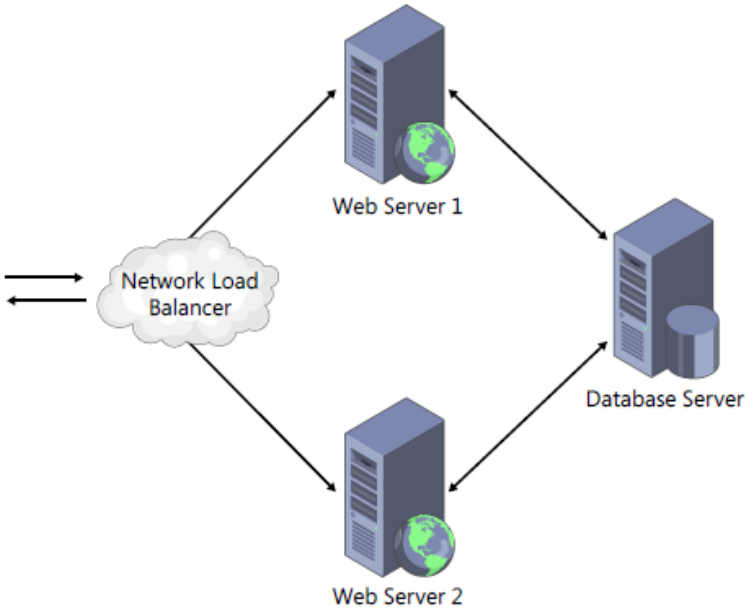


Figure 10 SharePoint is designed to scale out using a farm of front-end Web servers Web Parts. (Pattison, Connel, Hillier, Mann 2011, 3).

SharePoint is designed to be deployed as a farm where multiple servers handle the load and keeps up the various sites that the farm is hosting. This is demonstrated in Figure 10. If the farm load is too much, one can simply add more servers to farm and get more power out of the farm. SharePoint itself distributes the in-

stalled files and solutions to the new server. The solutions are provided to SharePoint in Windows Server Package (WSP) format (Pattison, Connel, Hillier, Mann 2011, 5-6, 40).

SharePoint uses solution files on disc but stores everything else to SQL Database. SharePoint developer never queries directly to SQL database. With SharePoint there is Collaborative Application Markup Language (CAML) to retrieve data from lists in SharePoint. The problem with CAML was that developer never knew if the query was build right before running the code. In version 2010 Language Integrated Query (LINQ) was introduced. The structure of the query was in the syntax so the developer knows the query is right if the code compiles. (Pattison, Connel, Hillier, Mann 2011, 129, 321, 329).

Version 2013 leans much more on Client Side Object Model (CSOM) to retrieve data for user. CSOM can be JavaScript but also C# code that runs on any computer running .NET framework. There is a Representational State Transfer (REST) API provided to use when creating client side code. This is directed to developers who create apps that can be independent of SharePoint context. REST API makes the SharePoint calls easier and they use client-side proxy to change data with web services in SharePoint. (Hillier, Pattison 2013, 46, 58, 77)

3.3 Usages

SharePoint is mainly used for Intranet, Extranet and Internet portals. Intranet usually contains internal workspaces and Extranet contains workspaces for collaboration with people outside the company. Basically SharePoint has all the parts for Intranet so the need for custom solutions is not required. Basic types of Intranets are communicative, social and virtual desktop. SharePoint can also implement Extranet with out of the box functionality but Internet portals needs planning and customization, usually because of layout and browser challenges. (Roine & Anttila 2013, 18 – 19, 22, 38, 34)

Data in SharePoint is stored on libraries and lists where users can share the data to other users. Social feeds and communities share the social side of SharePoint.

Company can focus all the functions in one place by using for example SharePoint sites, task-list and site mailboxes. One part of SharePoint is its search called Enterprise Search. Users can find data or people all over the SharePoint farm. Enterprise Search offers wide scale of functionality for example attaching other information systems to SharePoint or index public Internet sites and disk drives. (SharePoint 2013 Overview | Roine & Anttila 2013, 48)

Administrative users can build their sites from different features and functionality like document libraries or task, announcement and link lists. They can also add a calendar or a wiki to a site. The SharePoint app store offers much more custom features to the out of the box features coming with the install. This all means that SharePoint administrative users can be self-orientated and develop their tools and processes according their skills by using and activating features in SharePoint. (SharePoint 2013 Overview | Roine & Anttila 2013, 24)

SharePoint offers all the tools for Enterprise Content Management (ECM). First the admin user has to have a site collection and a site where is be a document library. Then the user has to make a content type which is a collection of site columns also known as. fields. Each document in the document library has a content type which defines what kind of metadata the document has. Metadata can be for example author, title, country, department and language. The content type also defines whether a field is required (not empty) or not. (Roine & Anttila 2013, 25 - 28)

SharePoint is also a collection of pages. The pages have content types for declaring what kind of information the page contains. Usually page content types are built by developer and not administrative users. Creating the pages by using developer customised page layouts is job for administrative users of content providers for example in case of Intranet or Internet. (Pattison, Connel, Hillier, Mann 2011, 554)

SharePoint 2010 brought the metadata model much further. It introduced managed metadata. With managed metadata users can create reusable enterprise taxonomies and folksonomies. “Taxonomies are hierarchical sets of terms, or

tags, usually centrally defined. Folksonomies are similar to taxonomies in that they are a collection of terms, but they have no organization or hierarchical structure.” With metadata users can tag data or refine searches if there are too many results with search words. (Pattison, Connel, Hillier, Mann 2011, 559 - 560)

3.4 Choosing SharePoint

SharePoint is easy to use and it integrates to other Microsoft products very well. On the other hand it can be very expensive system to use and moves the company towards Microsoft products. In some companies there are discussion can for example the ECM implemented with lighter software. SharePoint offers also Office 365 products which do not require dedicated server and can be paid monthly. (SharePoint 2013 Overview | Roine & Anttila 2013, 82, 98)

When SharePoint needs be modified for company needs and other systems SharePoint offers good API’s for extending the services. SharePoint has good scalability and there are multiple tools for developing SharePoint solutions. Microsoft is a big provider and one can trust that it will keep up maintaining the product and service support. (Pattison, Connel, Hillier, Mann 2011 | Roine & Anttila 2013, 80)

SharePoint is ready platform which can be used as it is after installation. It is mainly advised that it should not be customized. SharePoint is full of features but they are not very finished and because of this the customizations might be required if the out-of-the-box features does not fulfill the needs of the customer. It helps a lot that SharePoint can be used as it is and the features can be tested before tailoring it. Basic SharePoint project requires a partner that has expertise on SharePoint and helps on installing and demonstrating the features. If the owner needs more the partner delivers the customizations. (Roine & Anttila 2013, 6 - 9)

4 SHAREPOINT DEVELOPMENT AND SCRUM

4.1 Defining the project: Refinement and Product Backlog

Definition phase answers to question “What are we doing?” The output of this phase is functional specification which customer approves. The needs of the customer dictate what people are doing in the project. Project contract states many agreed subjects among others what are the resources and timeframe for this project. Some predefining might be done for the contract but we shall concentrate on the defining process throughout the whole project not before it.

Basically SharePoint Scrum project contract can only state that provider delivers a portal and everything else is defined during the project. Most of the times the customer wants to have a more specified scope for the project. Sometimes the project is sold as waterfall project but still done with Scrum inside the organization. The developers cannot start the coding if the Product Backlog is not done before the first Sprint starts. The definers who make the backlog do not need to know about SharePoint but there has to be items in it.

Sometimes it is better than people defining the project know about SharePoint because the work will be much more efficient. On the other hand, it is also good that people think what this system does do not know about SharePoint because the SharePoint knowledge could steer the defining. Usually SharePoint is bought as platform and there is more advantage to know about SharePoint than not to know. Then the SharePoint out-of-the-box features are not defined but all features rely on the existing features of SharePoint Server.

The Product Backlog must be created in the beginning of a Scrum project. First there will be larger entities like news, workspaces, blogs and general information. These features will be refined and split into smaller and more detailed features. If the Product Owner is familiar with the Scrum process or the ScrumMaster have enough time to guide Product Owner, the Product Backlog items will be like user

stories. User stories do not need much refinement and can be split into tasks quickly when taken into Sprint.

Making user stories with SharePoint does not always tell enough how the feature should be implemented. In SharePoint same results can be achieved by many different ways. User story might tell the developer how the customer wants something to happen but SharePoint might already have a feature that achieves the same results bit differently. User interface (UI) design play big role in this. If the project has customized UI the defining is better to start with the layouts. When an external Art Director (AD) has made the UI already, it is faster to define the system into features and user stories. AD must know about SharePoint or the created UI will be laborious to implement with SharePoint.

Defining continues during the whole project. Product Owner job is to put value and effort to asked features of the solution and choose whether to put them into Product Backlog or not. Product Owner meets the Team in the Refinement meeting. There will be discussion whether certain features can be done almost in the same way with SharePoint out-of-the-box (OOTB) features or not. Sometimes Product Backlog items are modified so that SharePoint implements them and sometimes certain features have so great value that is better to customize the feature to SharePoint.

The level of defining done during the project gets more intelligent every time. This is because seeing the new features after each Sprint and getting the possibility to play with SharePoint, gives the definers more expertise on SharePoint and the project to be built. User interface designing goes along with defining of the features because it is easier to perceive the whole picture by seeing the UI and knowing the environment. In SharePoint projects usually the Team makes the UI design but if not, the UI defining is job for Customer and Product Owner with the help of AD.

4.2 Designing the project: Refinement, Sprint Planning and Sprint Backlog

Designing phase answers to question “How are we doing this?” The output of this phase is technical specifications made by developers. Top level design needs to be done in the beginning of SharePoint and Scrum project. It has to be decided whether to create one or more site collections and should the solution be built on Online or On-Premises solution. These decisions are usually made by lead developer or the architect. Almost all other design happens during the project among the Team.

Refinement meetings can contain a lot of technical designing if there are questions about should there be customization this or using SharePoint OOTB features. It is recommended to write down the conversation and the ideas during this discussion because the value of the feature is measured based to these ideas. The workload estimates are done also when Product Owner is deciding should some feature be done.

Refinement meetings contain defining and designing. Sometimes refining an item brings out that there is no user interface for that. If the interface can be done with SharePoint easily the Team does not need the help of AD. If UI layouts are needed the refined Product Backlog item can be made more detail but dropped down on the backlog waiting for the UI layouts and taken into future Sprint. Missing of the UI layouts can be noted during even Sprint Planning part 1. If the AD co-operation is good and fast, the Team and Product Owner can take feature into next Sprint and AD delivers the layout fast.

If the project does not have external AD or AD at all the UI layouts are done in designing phase by developers. Then the define phase of PBL items is much more difficult because SharePoint has so many functions itself. Even if the project has external AD the UI design might take place during project but usually before the technical design.

The actual designing happens when developers see which features Product Owner chooses in Sprint Planning part 1. The planning process kicks off in the

minds of the Development Team. In Sprint Planning part 2 starts the technical design how are these features going to be build. One by one the Team splits the features into tasks. For example, the News feature might need first a News-site. Then there must be a content type for news page. The page must have page layout. The News site must also have layout how to show the News list. Should there be filters for the News list or how are the filters going to be implemented. Then the created task must be detailed and put into order so that the tasks can be done in SharePoint project. For example the site must exist before creating view for all news and content type must be done before creating page layout because page layout is always based on content type and its metadata.

With SharePoint functionalities can be done in many different ways. For example the position of News site can be hardcoded in the solution or set as parameter. The code can find the news site by comparing the site inner base type or deduce it from some other parameters or metadata. It is very important to discuss about different ways in the Sprint Planning part 2 and decide which to use. Then the same method should be used throughout the whole project.

SharePoint is based on data, lists and libraries. The information is stored somewhere and viewed elsewhere. When there are much data there is many different ways to bring up the data. Developer can seek it from the search index, do site data query or read it from the cache. This is another important thing to notice that the data extraction is done the same way everywhere. If some pull depends on indexing, the other on caching and third uses heavy instant pull the information is showing up with different rules. Designing the data pulls together gives the developers time to tell their point of view and agree with others. Once the decision is made for the features everyone knows what to do.

Implementing UI layouts to HTML and CSS is one job that can be done in many ways. Are the features having different style files or is the whole solution using same ones. How is the naming of technical parts done? What way serves the purpose of this project the best? There are many questions that raises time to time and must be re-thought or reminded to each other. If the Sprint Planning part 2 is done properly and the task descriptions are clear the Sprint is half done.

Re-designing can be done also if something did not succeed or there are some other technical difficulties. Sometimes the first design just does not work or there is something that wasn't noticed. Then the Team can refactor the tasks. The point is to decide together so that everyone knows how the solution is going to be built and what is designed.

4.3 Coding the project: Daily Scrums and Team

The actual coding of the project starts after Sprint Planning part 2 has been completed and there are tasks. Usually developers should struggle with one PBL item at time and always aim to make it finished before starting another. With SharePoint the PBL items usually are pages in a portal or Web Parts in a page. A page might contain a web template before it can be added to solution and a Web Part might need a result source before it can be added to page. This means that it might be better that each developer concentrate on one feature instead they build one together.

Starting a new SharePoint project needs the Visual Studio solution core and project. This should be done only by one developer or two together back-to-back and not separately. The design must be done on the design phase so precise that there is only one way to do the core according to the specifications. The same goes with the Sprint Planning part 2 phase when the Team makes tasks so that there won't be multiple different implementations that do almost the same thing.

One example is to have a page with multiple Web Parts that shows query results done from the Search Service Application. First there have to be a web template to provide the site collection root site to http://developmentaddress. And then a web template for the content site or just a feature that provides the default.aspx page for wanted site. Then the first Web Part can be added to the page. If the Web Part data source is general for example "All Events in site collection Development Site Collection in address http://developmentaddress" there should be a Result Source done in the SharePoint farm instead of doing the query rules directly to the Web Part. If there will be many of that kind of Web Parts it is easier

to manage the query rules in on Result Source than modify multiple “Events” Web Parts in the whole site collection – in development phase and especially on maintenance phase. Then there must be implementation of a Content Search Web Part that uses the “All Events…” Result Source and there must a display template set to the Web Part that defines how the data is shown.

If all Web Parts are provisioned and Result Sources are named in different way the project is a mess and management of the site is difficult. The Team itself takes a lot of time to learn how to use the site they have built. But if all the things are done on the same way in on project it is easy to deduce how the rest of the solution work. Even the customer can be easily orientated how the SharePoint solution works. This needs Daily Scrums and good communication among the Team even if the tasks are done as designed.

Daily Scrum a.k.a. daily stand-up is the place where the members of the Team tell what they did yesterday and what they will do today. It is the place to open a bit done tasks and especially if something was made differently than it was designed. The technical discussion should be after the daily so that ScrumMaster and other not needed personnel don’t have to listen that talk. Usually it could still be better to talk fast some of the technical things in the daily so that everyone involved knows how things are done in this project.

The Team should also have some kind of Definition of Done where states that all tasks that are ready are documented and there are a test cases related to them. Those documents can be viewed during the Sprint so that everyone tries to do them in the same way. The ScrumMaster or someone in the Team should look after the documentation and test cases during the first Sprints that they will be done. Off course the lack of documentation or test cases show up in the end of the Sprint but it should be monitored couple of times during the Sprint. It must be checked that are everyone following the plan and the rules made together.

The project could stop evolving if a member of the Team checks in code that does not build or a solution that cannot be provisioned as site collection. When a developer takes the latest version from TFS and tries to provision the site collection

he cannot continue if there are errors made by others. The others must wait for one to fix the problem and time is wasted. The worst thing is that it reduce motivation and commitment to the process. “Why should I do if the others won’t?”

The coding phase is very mechanical and machinelike. The Team executes tasks and communicates together if they have problems. This should go as planned and the workload estimates should prevent a stress. If someone is staying too long or too many days in one task it will be noticed because every day basically people should have new tasks and finish the old ones. This is hard situation for creative people or people used to be very social because process should just squeeze out the designed work of the Team and rules must be obeyed. There is no space for going solo.

During a Sprint the developers provision their personal development environment site collections multiple times a week. If there is problems in the code, in the configuration XML or in the script it will be noticed. The Team must trust each other and their skills so that there is no scolding among the Team. If someone always does the same mistake there could be some discussion but the point is that Team learns from each other’s mistakes and everyone don’t have to do the same mistakes.

4.4 Implementation: Install and Sprint Review

When the Sprint is ending and the Sprint Preview is coming the development should freeze. Usually the day before Sprint Preview the developed solution is installed to staging and test environment. Then the solution is tested with test cases created during the Sprint. If there are errors that cannot be fixed before Sprint Preview, the related PBL items are not presented as ready.

SharePoint solution install goes with WSP package. The old one in the farm is retracted and deleted or upgraded. Usually the site collection is rebuilt in the development phase. Upgrade and rebuilt can be done manually but it is more efficient to do a script for that. Developers have script for upgrading the package that

they use during sprint. The same script runs on the other environments. By doing this the Team can avoid human errors in install and site provision phase.

Installing usually takes only one developer so the rest of the Team can continue development for the next sprint. Basically the install phase is fast thing to do with the script so the Team should share the tasks. Two members start to install staging and test environments and the rest begins to go thought the test cases after the install. If all goes well the install and test phase is over before the day ends and the Team can continue next Sprint development.

It is not according to Scrum that the Team does anything that the Product Owner have chosen for the next Sprint but there have to be a one day cap in the development for the install and test phase. Agile methods do introduce continuous integration but with SharePoint there can be so many things go wrong that it is better to have the whole Team to assist if error occur during test. When the solution is ready for Sprint Preview there have to be some work for the Team to do. So this phase needs a bit bending of the rules if the developers don't have some other job to do or they can leave early that day.

In the Sprint Preview the Team presents to the Product Owner and other participants what they have done in the ongoing Sprint. They can go through the test cases or just follow Product Backlog items in the Sprint Backlog and show the functionality of those features. If the Product Owner approves the changes the Team can install the solution to production environment. This is the first task of the next Sprint. The Team also writes down all the changes that Product Owner wants to have to the developed features. It is normal that the customer want some changes when they see the solution working. Changes can be brought up in Refinement meeting if customer want to test the solution first.

If the customer is not familiar with SharePoint some of the OOTB features will be criticized. The Team have to explain to the customer what they have built and tailored and that comes naturally from SharePoint. With SharePoint things can be done in many different ways and that can also confuse the customer. Usually the Sprint Preview is long enough so that the Team can demonstrate and explain

all the things related to the created features and make the customer satisfied for the past Sprint.

4.5 New releases and update: Next Sprint

As long as the site collection are re-provisioned in the end of the Sprint with the new package upgrade everything goes quite the same as in developer environment. After each Sprint the data is wiped out and the new site and content type structure is built to the staging environment. Mainly the WSP is removed and re-installed so that no files could stay upgraded because of file handle or DLL locks.

In the product release the solution is installed to production and the site is provisioned. After this point the site is not anymore re-provisioned. This means that the upcoming updates in Sprints or maintenance phase must be done differently. The package can be just upgraded and the new features are installed to farm but the site is not removed. If there are structural changes in web templates or list definitions the changes do not affect to the created sites and lists in the production site. The update must be done differently.

Upgrade for structural changes takes double work after the release because the WSP has to contain the changes when new sites and lists are done. If the existing sites and lists needs the changes there have to be a script or code that does it. Some changes cannot even done after creating for example the security bits of a list or does a library allow folders. In the creation of a library SharePoint makes a table in the database and there are no features for alter all the deepest changes afterwards. Then there have to be a new list that is created from the new package and data from the old list is moved to the new list. After that the old list can be deleted. This can be done manually but it is recommended to make a feature or script that does it programmatically. The Team must discuss about the cases and write down for example into DOD how certain updates are designed to be done in the project.

Upgrading a production environment makes a pause for the sites and sometimes the index must be cleared there are new managed properties introduced. Production updates usually are done outside office hours so that users don't have stop in the service during the work day. This must be also acknowledged when the first release is installed to production environment.

4.6 Project Models

SharePoint projects can be divided into groups according the genre like Intranet, Extranet and Internet. That is the most common way because it tells so much about the service provided. The second important thing to recognize is do the project contain web templates and site provision. The nature of the project changes if there are only features that will be activated after install or are there site collection provided.

Genre specifies the project but so does the size. There whole project model and installation setup changes whether the project is a couple weeks or a half a year project. If the project is very small there could be only some install scripts done but if the project is huge a more complicated installation script should be done. Usually the script evolves during the project. At first it just install the WSP and maybe provisions the site. After a while it can create navigations, set various settings, create result sources, managed paths, managed properties, provision various site collections, make connection between site collections and create test data.

Organization might have project templates but a common way is to copy from the core from another same size project and modify the parameters suitable for the current project. This should be done in Sprint 0 or in the first Sprint if there is no Sprint 0. The script should also have parameterized settings related to environment so that running the script is as easy as possible. Just click run and everything happens automatically. This might take some time to implement if there is no ready base for that. Doing the scripts beforehand saves time a lot. If the project is provisioned to a new development environment it takes 5 to 20 minutes and

provisioning is ready. Installing to staging or other environments takes only 5 to 20 minutes and installer person can be sure that it is the same as development environment. This is important because in Scrum the site is re-provisioned and the package installed again after each Sprint.

Sometimes if project is very small and it does not provision a site there might be a reason not to use Scrum. Scrum is used for project that might not have very specific specifications or the specifications will specified later on. The project should be done more like traditional project if the customer provides detailed specifications, time limit and budget in the start. The project can contain all the characteristics of Scrum but there is only two Sprints. The Team makes the system in the first Sprint and then the PSP is tested against the specifications. The second Sprint is done only if the solution differs from the specifications.

Very small and detail specified projects are not for Scrum. In Scrum the point is develop the specifications and maybe offer more for the customer and let the customer steer the ship called project. If there is no need for development and no time for the extra work the project method does not have to be flexible. Some customers get irritated if the developers demonstrate improvements or propose something against the specifications.

For the developer the projects should feel the same as agile regardless the project is sold as agile or not. It is good always to have a planning part 1 and 2 and the daily standups so that everyone in the project knows what happens. The review does not need to contain the whole team and there don’t have to be a Retrospective if the project lasts only one Sprint or two. The test cases and documentation should always be done. It is good to say out loud at the beginning of the project and agree together which parts of Scrum characteristic is used if the project is not done with Scrum.

All projects more than one months should be done as Scrum even if they are sold with fixed time limit and features. Scrum method reduces the stress of a developer, tells what and how to do and keeps the project and Sprint goal fresh in mind and gives indicators if the project is completed on time. There should not be

Scrumbuts but sometimes they are better than no project method at all. More about Scrumbuts in the next chapter.

5 CASE “SHAREPOINT INTRANET”

5.1 Starting the project

The project was huge. It was a yearlong project that had three or four developers doing it simultaneously. Scrum was the chosen project method and Sprint duration was three weeks. The administrative project manager that Affecto needs to have for typing the hours for billing and taking responsibility was chosen to be the ScrumMaster. Developers were the Team and there was no Product Owner because the customer could not engage to the projects so much because of other responsibilities in the job and they did not have enough experience on Scrum to make the Product Backlog.

There was external project manager hired for the customer due to the lack of time on the customer end. This external project manager was kind of a Product Owner but the ScrumMaster was the one maintaining the Product Backlog. The team of customer project team were about ten persons. The Scrum was altered so that the actual Refinement was done with Product Owner, ScrumMaster and the customer team. Sprint Preview and Planning part 1 were facilitated so that there was the Team, ScrumMaster, Product Owner and other people from customers end.

The upcoming Sprint Backlog items were chosen and specified during the ongoing Sprint. In the Sprint Planning part 1 the Team and the Product Owner discussed about the backlog items. The ScrumMaster had to be more technical and know SharePoint to do the Refinement and specifications with the customer during the Sprint. This made a gap between the Product Owner and the Team and changed the ScrumMaster role to be more like a traditional project manager who says what to do.

Customer had experience on SharePoint and part of the current system ran on SharePoint 2010. The migration of the SharePoint part to version 2013 was another project running on the same time with the same developer resources. This was also ported to the technical solution afterwards. The customer did not knew

much about SharePoint 2013 and its features and neither did the developers. The new technique and migration was new to all of the members in the project. It was not a problem because I've been on the same situation when the version change from 2007 to 2010. It means that developer should success a few certificates and learn the new version features and techniques on the same. It also means that there were more Googling and searching for articles on the new version solutions but nothing that was not familiar to the developers.

There were one junior developer on the project and to him everything was new from the project method to the SharePoint development. The author of this thesis was the senior developer to guide and teach him during the project. So there were many pitfalls to trip over. The possible problems needed to be paid attention and a plan was to be made how to manage the possible problems. There was discussion about the possible problems but there could be more communication about them.

There was not Sprint 0 and the developers made the Visual Studio solution for the project if the first Sprint. The theme for the first sprint was to make staging environment, create development environments and tools. The first Sprint started and we get to the Sprint planning part 2. All the tasks were made and development started. We used OneNote for documentation, our SharePoint workspace for storing the documentation and test cases. There was an install script modified from old project and was developed bit further. DOD stated that script must run without errors, codes are checked in to TFS, the work should be documented and test case made to workspace and if the task was last task on PBL item the PBL item status should be changed to Done. Everything was good and ready to start.

If the contents of DOD is copied from another project and not evolved during the Sprint the Team does not commit to it so much. We agreed all the parts together but I was afraid that the commitment to Scrum and the DOD was not good as it could be.

5.2 Getting known to Scrum

Making the tasks small as possible and taking only one task at time was difficult to understand at start. Earlier experiences for the developers was to take one feature and do that until it was ready. Scrum was forcing people to document and test the solution daily and piece by piece. There was a lot of conversation and arguments why this method was good. During the first Sprint there were lots of reminders and notifications for the Team how to do Scrum. Even to ScrumMaster.

After couple of Sprints we had quite similar way to make tasks but the descriptions of the tasks were incomplete. There were also problems with the DOD, all the members of the Team did not follow it. I asked if there can be Retrospective. The first Retrospectives was made on my lead. I told to the others why these are for and how do we do them. Then we had a meetings where to improve our process. First Retrospectives felt like I was speaking alone and they were arranged only that I can complain to others. That was not very far away from the truth. Finally the other members of the Team realized why the tasks must be so clear and what the point of designing all the things was in the beginning of the Sprint. If there is very vague title and description, each solution is done differently.

After three or four Sprints the Team started to work more like a Scrum Team and the communication was better. Sometimes the Daily Scrums were too long because there were too technical explanations. We had half the men in another city so online meeting was the only choice for us. It was better to talk bit more so that there were some kind of contact among the members. There is always challenges when people that are working together cannot have real life interaction. The feelings aren’t transmitted so well and there will be more easily argues. We managed to do the work and met once in a three week in the Sprint Preview and Planning.

There were no meetings where to share SharePoint technical knowledge. We used online meetings in a group or person to person to go through features in the project. Sprint Planning part 2 was only for design but when there had to be changes or show others how I had solved this problem it had to be done through email and online meetings. At start it felt quite irritating to stay an hour in online

meeting and go through technical stuff. After couple months we got used to it and because our everyday life.

Scrum methodologies and characteristics needed to be reminded once in a while and some part I lost my enthusiasm into guiding the Team. There were two reasons for that. First was that I felt that I was the only one who wanted to have Scrum method. It felt that I was all the time complaining or reminding other that could you please do this what Scrum says or please remember to follow the DOD that we all have agreed with. The second was that it felt like the ScrumMaster did not back me up. Like the ScrumMaster didn't care so much if we followed the rules of Scrum or not.

We were in Sprint four or five when the ScrumMaster started to add Product Backlog items to Sprint during the Sprint. They were just popped into the Sprint Backlog and in Daily Scrum the ScrumMaster informed us that these items needed to be split into tasks. Our project started to be quite far away from Scrum when the ScrumMaster acts like a project manager in traditional project and does not care about the agreed rules.

5.3 Understanding SharePoint

The idea in a project is to recognize what the customer needs and what is the best solution to do that with SharePoint. SharePoint version 2013 leans very much on Search and the best solution is try to use the out-of-the-box solutions and configure them to show data from index. Then there would not be so much server and database processing and the solution works as is should be. Although there were challenges among the members of the project to follow up the project method the quality of the Sprints output was good.

After couple Sprints the Team realized that we should do all the similar features in the same way in this project, for example a result source, managed properties, content search web part and display template which all was used to show data with given rules. This combo contained search configuring, frontend coding and SharePoint modules for setting a web part with given properties into a page. In

community sites the combo needed also backend coding so that the properties of the web part was changed depending on what kind of community the site was and what was the name. The output was a web part that displayed current community tasks, events and documents. By going through together these techniques in Sprint Planning everyone knew how to do this.

In the Team there were a junior that needed guidance and then there were user interface oriented person that did not know much about backend coding or SharePoint development. And then there were the architect and me who could do actually do all the tasks in the project. I and the architect were the leading core of the SharePoint knowledge and the project was designed by us. The UI guy was in charge of the HTML and CSS implementation from the layout pictures. The junior developer was supposed to learn from us during the project and do tasks that he could.

Junior developer didn't know even basics of SharePoint like content types but he learned very quickly. The improvement in the work of the junior was amazing. In the beginning of the project he needed help in every task but after a while he could seek solution for problems by himself. SharePoint development is very much Googling for solutions to problems. The junior learned to imitate our examples and find solutions to problems so well that in the end he didn't needed to be guided at all. He had become a member of the development team and communication with him was like to the others in the project. This kind of overall and huge Intranet project is the best way to teach someone what is SharePoint and how it should be developed.

The user interface guy a.k.a. Art Director was supposed to do only UI. After a while he was very interested how SharePoint development works. He started to add modules and features to the project. Sometimes he asked help and sometimes we had to correct him a bit but after that he could act also as a developer. He coded JavaScript very much and made structural changes to the Visual Studio solution and its modules. He understood backend coding but didn't do that much but that was not needed in traditional SharePoint development. The UI guy became also a part of the developer group.

The architect was hired to the company in the beginning in the project. I had been working in Affecto as the head of SharePoint more than six years. The architect is skilled with SharePoint and coding, so am I. He was more than ten years older than me. In the beginning of the project there was a bit power struggle between us. Sometimes our solutions to problems had very different approaches. Sometimes tasks descriptions had two solutions or the solution of the one of us who was typing the task in Sprint Planning part 2. After two or three Sprints the situation was eased. After seeing each other's expertise the trust became to evolve between us. When you respect the other it is much easier to tolerate other person's special characteristics and weird feeling approach to problems. Working to together started to bear fruit and we started to learn from each other. Somewhere in Sprint five or six the design of the tasks started to be very similar with us. We had found a common way to design and do features in this project.

The output coming out after each Sprint had good quality. If someone noticed some errors or faults we he made a task for fixing it, and these tasks were done and test cases done according to them. Our teamwork worked despite the total following of Scrum method did not work. Everyone documented their work and made test cases so that we could test the solution after installing and before Sprint Preview. Growing together and evolving in a project was what happened.

When the project expanded and there was migration to old project and workspaces we realized that there will be hundreds of site collections. All the site collections needed to have similar layout and top navigation. In design phase we split the feature into tasks. UI guy made the layouts and put it into SharePoint in modules. I created the feature and feature receiver code that pinned the navigation terms from the root site collection and the architect made the script that activated this feature to the migrated site collections. This is how we all used our expertise and solved a difficult problem together with Scrum. We had prioritized order in the tasks so we knew what needed to do first and what last. If the problem would be on one developer the solution could take much more time and probably would be not so great. Scrum gives the conditions and boundaries how the process works.

5.4 Scrum, but

Term Scrumbut comes from situation when people are not doing Scrum 100%. It means that they take parts of Scrum they like and live the rest. When someone asks “Are you doing Scrum?” the answer goes like “Yes, we are doing Scrum, but this and this we are not doing”. To work Scrum needs to be taken 100% as working method. There aren’t parts that can be dropped off.

Between Sprints seven and eight there could not be an ongoing Sprint because it was holiday season and people had their holidays in different time. On customer side there could not be anyone be present in the Sprint Planning or Preview. Developers made tasks from Product Backlog items so that there were enough work for everyone to work through the holiday season. After holiday season the Scrum model did not start again as it was supposed to. First there were not enough data in the Product Backlog and then ScrumMaster did not have time enough to participate in the project. Part of the Team wasn’t so committed to Scrum so they expressed their opinion that it was better to take bit slower.

An effective cause was that customer could not participate to Sprint Planning part 1 and Refinement meetings enough to produce enough items in the Product Backlog for new Sprint. Because of the situation and personnel the project model had to be changed. The project was not anymore Scrum. Project manager explored different project models and found Kanban. Kanban is a project model which does not have iterations, DOD, proper roles or other things we needed in our project. The manager found an article telling about Scrumban which is mixture of Scrum and Kanban. It is like having all other Scrum things but not Sprints. The Product Backlog items are split into tasks couple at time when new tasks were needed. We decided to try this one.

It was a real example of a Scrumbut project. Sprint used to have certain time limit and certain amount of features. In the half way of the Sprint it was easy to sort out are we in schedule or not. When doing Scrum there were evaluates on tasks and usually they kept quite well. The basic evaluation was small, middle size or big. Small was couple of hours, middle size meant half day and big was one day

job. After a while there were no estimates at all done for the tasks. There was no use to make the estimates because making them took time and there was no clear time limit to fit the tasks. We did not have Sprint so we did not have goal related to time, no deadline.

From time to time the Sprint and Product Backlog changed. Some Backlog items popped into the Sprint backlog with higher prioritization number than the others. It meant that we had to make tasks for those after next Daily Scrum. Sometimes the descriptions of the PBL items were very poor or we did not find the specifications for them. Sometimes the project manager was out of reach and the communication and questions were delivered by email. Efficiency in making the tasks reduced and same time the old tasks that were made went further away in the backlog. This meant that there could be false design when we get new features to design. Other problem was that developers simply forgot something they designed more than four weeks ago. At least if the description of a task was poor the time gap between design and implementation was too big.

In Scrum people are having success feeling every three weeks when all the items in Sprint Backlog are done and presented to customer. “We made it again!” is the common feeling. If time was running out in the end of Sprint people worked faster. Now we did not have Sprint Backlog. We had just a backlog. No plan what to do in which time. Work is fine but the basic motivation disappeared because there is no goal. No time limit or feature amount to achieve.

Off course the other side was that there were no stress but truly we never had stress in this project. If in Scrum some feature did not get completed in time it just were moved to next Sprint. In Scrum person is supposed to finish one PBL at time. Now the Backlog and the items weren’t taken in order. Because the lack of the rules everyone started to take tasks from here and there based on feeling or how detailed the task was. This comes to that point when multiple PBL items are in progress and none are done. In Scrum the point is usually have PBL items in progress only couple at time. All the other are in To Do –state or in Done-state. So that in the end of Sprint there are N couple ready features and only couple or none that cannot be presented to customer.

This is a great example what happens if you only drop some parts of Scrum method and don't follow up if this way is really working. Because of the customer demands this still worked. Customer can get "Sprint Preview" when they want. They see all the things that are done and the project is improving but not with speed that it could. Even the Retrospective meetings dropped off because there were not clear process to have first Sprint Preview, then next Sprint Planning Part 1 and 2 and then Retrospective. It was just forgotten or left away.

There are many factors about people, processes and technique in this situation. Some people like to re-think various times. Some like that every time you take a new task you have go through other tasks, ask people what they have done and read the specifications and documents to find out what was supposed to be done in this task. This working is slow but easy and there is no stress. On the other hand there are people who like to comply with the rules. These people don't like to make effort every time to take a new task and ask around what was going on. This is about personalities and efficiency.

After noticing these failures the project team decided to continue with Scrum. One part of Scrum is to let the members to slip out and realise what the Scrum method really brings to work. Nothing was failed or no good quality was delivered. The Team just took a nap and woke up. This brings the Team together much more than hearing a lesson about Scrum.

5.5 Project summary

We used Scrum, we had experts and juniors, and we had test cases and documentation. The team shared information and evolved as a group. The individuals evolved as persons and developers. The output of the project has quality and looks nice. Everyone in the project are one more experience richer and have taken one step towards to be a better professional. This is because Scrum forces to split features into small tasks and discuss together about the solution. All members in the Team are involved and they learn the same time the others solves the problem.

Scrum method is efficient and works as it is. Orientation to the chosen project method should have done better. The members of the project group did not know enough about Scrum to implement it as it should be. Most of the members in the project did not see the value of having pure Scrum project. If only one is motivated to do Scrum and starts to teach the others the motivation using the Scrum can collapse. The members of the groups should be committed more to the project and chosen project method in the beginning of the project. Luckily the people in the project were committed to the project itself and SharePoint development. The need to be better helped to fulfil the tasks in the project.

Despite the project method implementation did not work the project sill had a project methodology to follow. That is important that if the agreed rules does not work or the changing situations forces the project to change there are change and project manager and members are capable to leave the old rules and make new ones. This is flexible and learning organisation which adapts to changing situations.

In the Scrum perspective the project did not succeed so well. Overall the project succeed. Customer got what they wanted. They get it as they wanted and when they wanted it. The output quality is good and the solution works as it should. There could be stricter follow up on the project method. This project taught to each of its members something showed to us that we will satisfy customer needs nevertheless the situations aren’t for us.

I cannot describe the project in more details without revealing too much of the customer or the project. The goal was to tell how one yearlong Intranet project was done and couple of general SharePoint development cases solved with Scrum method.

6 SHAREPOINT SCRUM PROCESS

6.1 Project start, first Sprints and DOD

Usually Sprint 0 is when ScrumMaster and Product Owner have time to create Product Backlog items and the Team has time to make the core for the project solution. With SharePoint it means Visual Studio solution containing the needed features for the project. If the project needs new site collection, it is good to make web templates, content types, site fields, page layouts and install scripts ready for general SharePoint project. Everything is ready for tailoring the solution for project specified needs.

Developers insert the solution into Microsoft Team Foundation System that hosts the source control, backlog lists, test case lists, iterations and all the data needed during the project. Product Owner types the Product Backlog items into TFS with the help of ScrumMaster. The Team and Products Owner can have Refinement meeting and decide if they have enough items to cover the project or first Sprints.

When the Product Backlog is created the first Sprint can start. SharePoint already offers working platform so it is easier to start with basic features for example News-section. News-section is very general to Intranet and Internet sites, and even Extranet sites have some announcements. The other technical core of solution can be done also in Sprint 0. The main thing is have items that can be taken into Sprint backlog and the items are detailed enough that developers can create a solution based on them.

Before starting the actual tailoring the Team have to make tasks that they do have the technical core for solution and test case list. After that the Team must create a DOD. There should be at least the core checked in to TFS. It is better to attach documentation and making a test case into the DOD. If documentation and making test cases are set into individual tasks, it will easily get forgotten during Sprint. If developer A codes a module, it is very hard for developer B to document or make a test case for it. If there will be time schedule pressures then at least those

tasks are bypassed. It is better that everyone documents and tests their own work and gets used to it that these things are done always no matter on hurry or pressures.

During the first Sprints the Team gets known to each other and finds out the level of each's expertise level. After couple planning meetings there can be seen if there are architect or main designer or do the whole Team really design the work together. One hint of balanced team is if the role of typing the tasks changes. Then the Team is very balanced and working together but it is not a must. Sometimes there are people from different level of expertise and it is better that one leads and other assists. Especially with SharePoint because the designer must know the whole system to discover the best solution.

6.2 Documentation

The level of documentation must be checked properly in Retrospective meetings. The whole Team must agree if the level is enough or should they make better documentation. The Team must decide and agree to whom they are documenting. The final documentation or for the Team. The level of technical documentation should be chosen. For example if OneNote is used, making of links to between modules and whether to document even ID:s of fields and content types or not are things to be discussed about.

Normally the IDs of modules are not needed but relations between them are. Figure 11 displays a list of web templates created in the project. Clicking the name of it opens the template and description. There description of what is the web template and why it is used for. There are also links to features, web front page and other settings like web default page template. The documentation has also print screen from the site but it cannot be attached here in thesis like the other features in the web template. The front page link opens a page that contains screenshot from the front page and describes what Web Parts it contains. Default template link goes to page layout description page. The documentation is like a Wikipedia and easy to navigate.

Figure 11: OneNote documentation with links

The Retrospective meetings should be documented every time. It takes less than hour to have the meeting. All things mentioned should be written down. This is how everyone of the Team can feel that their opinions are heard and processed even if their things are not chosen to be the next development target. Figure 12 shows a simple template for documenting the Retrospective meeting.

Team should list all functional, technical and other things during the Sprint that was good. They have to think why it was good and then how to maintain the level. The same goes with the things that needs to be improved. The point is not to write the most specific reasons and fine language. The point is to get it documented and the Team discuss about the things that even one member felt was good or bad. As it can be seen in Figure 12 the last thing is to choose one or two improvement points for the next Sprint. These chosen points are checked at the next Retrospective meeting. Usually all the discussed things gets better because the making those as discussion topics makes the things more visible.

Documentation should contain all needed information about the project and the environments. All the environment information like domains, urls, server names and ips, database names, username and passwords must be saved to secure place. It is the responsibility of each member to save the needed information

when they came aware of it. Usually one person is contacting the customer person that provides the information and that contact person is responsible to document the information he is provided.

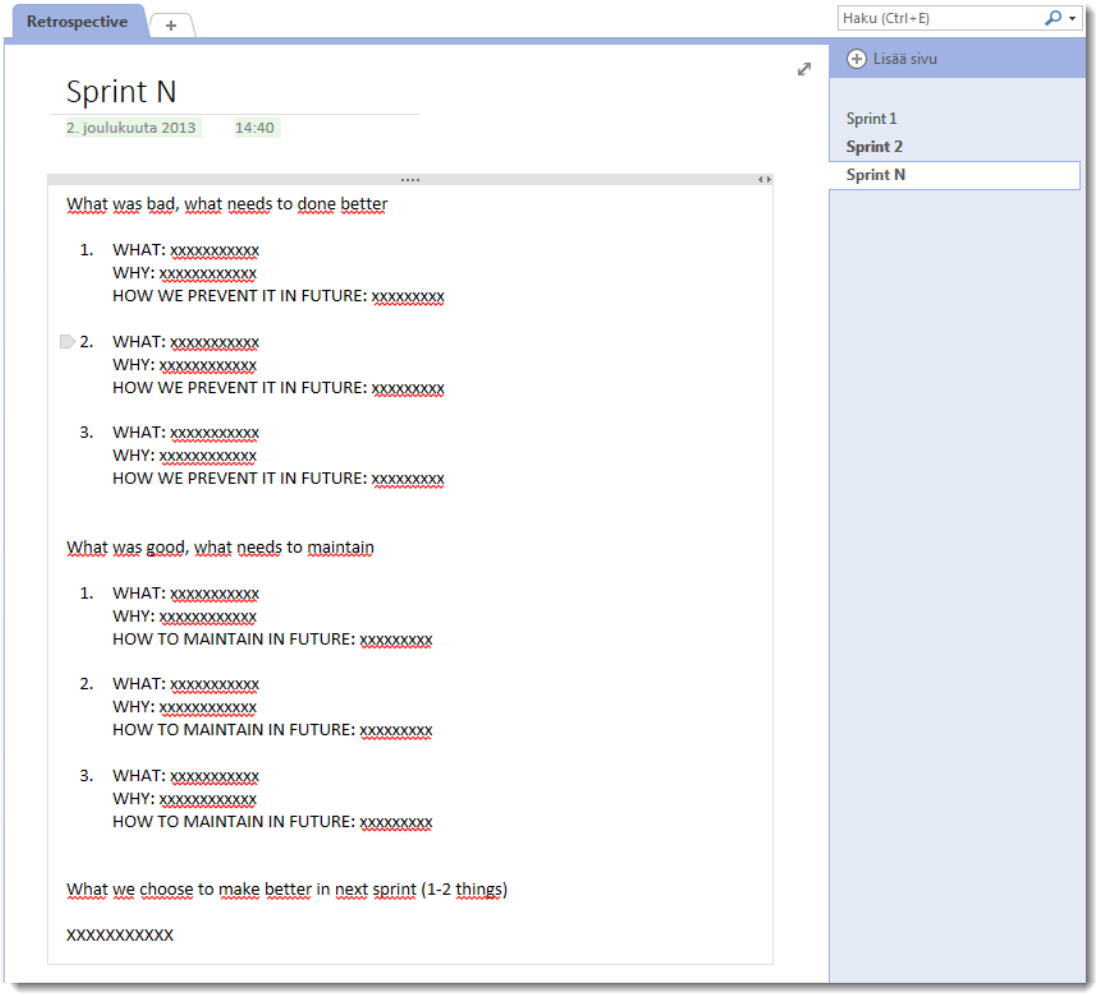


Figure 12: Retrospective documentation

There are more important documentation that must be done like test cases, install instructions and install log. They are introduced in the upcoming chapters related to the topic.

6.3 Installation cases

SharePoint installation is quite simple because of the WSP-files that has everything in them. There should be install script that removes the old WSP-file, installs

new one and creates site collection again. This is the basic case when creating a new SharePoint portal in Scrum project. Until the project is installed to production and released the site collection will be re-provisioned multiple times. It is because this way all the structural changes are provisioned to the SharePoint site. The changes are for example changes in web templates or Web Parts on pages or even site hierarchy changes. The script should do all the configurations so that there would not be human errors. These configurations are for example creating Result Sources, Managed Properties, Blob cache enabling, Web Application properties and creating Managed Navigation TermSets.

The install script should built so that there are environment related variables already in it. Developer simply runs SharePoint Management Shell with administrator privileges and then the script. This is demonstrated in Figure 13. First the script loads the variables needed in the install. After that it re-installs the WSP-files in resource directory. Once the WSPs are deployed the script re-provisions the site collections and then does other configuration needed.

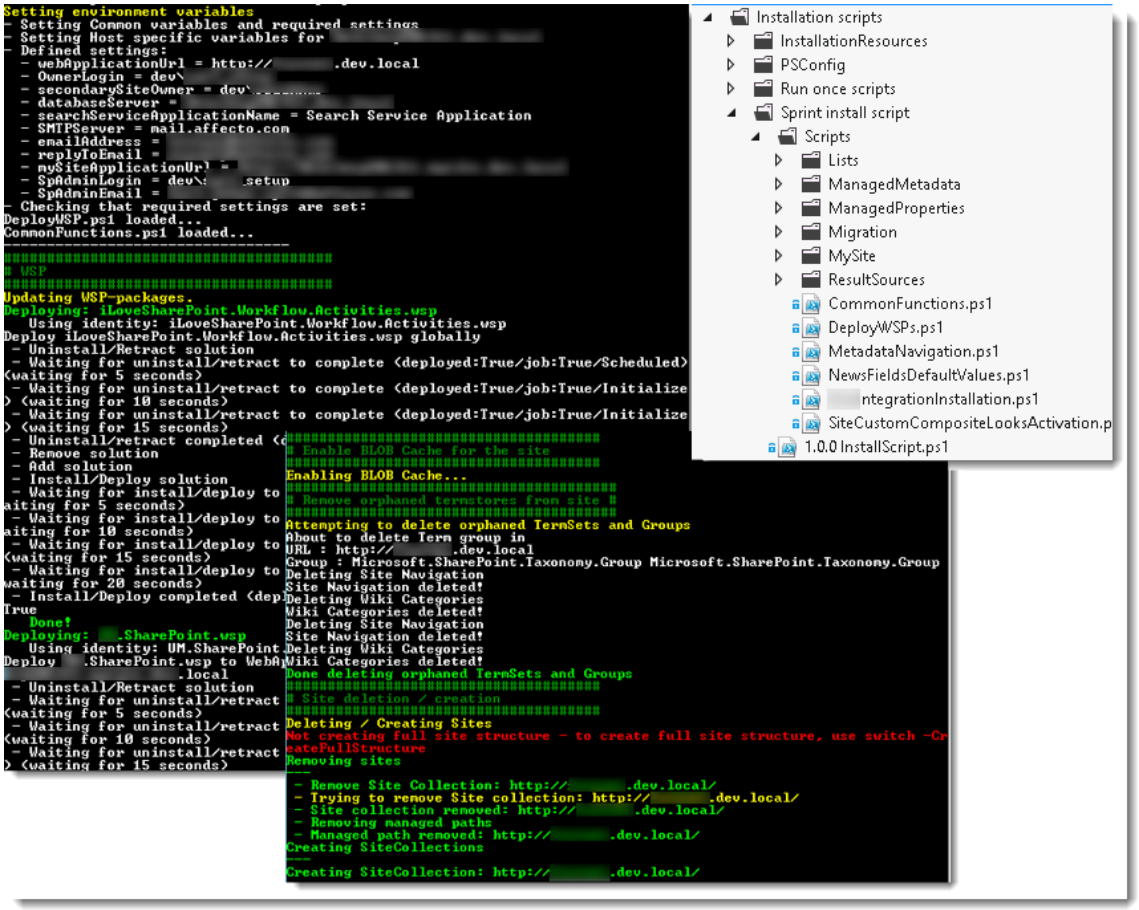


Figure 13: Example of install script

When the solution is released and installed to production the development gets more challenging. All the development should be done twice, one for building a new site from scratch and another for modifying the existing one. The running of the script is still the same. Each Sprint install script is developed during Sprint and in the install phase the developer runs the script which updates the WSP-files and configures necessary things and creates possible new sites.

All install tasks must be included in the tasks of a Sprint. It is best to put into DOD that if task output requires some steps into installation for example creating new managed property, index reset and full crawl it should be done in during the task. This means that task estimates must contain also time for this operation and testing it. Other practice is make individual task for install script but it has the same problem than the testing tasks. Making of these tasks easily are forgotten or the tasks are dismissed because of hurry or person change in PBL item. The new

person does not exactly know what to do and leaves the task undone. This is why it should be in DOD so that all the members of the Team get used to this practice.

The installation of the PSP at the end of Sprint should be one task. It can be calculated in the end of the Sprint but it is easier to make an item for ending the Sprint into Sprint backlog and estimate the time for each environment where the solution is deployed. Then the same item should also contain tasks for testing the different environments. This is the way ScrumMaster have all the work in the backlog and estimates directly to Burndown charts.

All install activities must be documented. The documentation should not be too long or hard to write. Just the basic things like who run the script and when. Was there problems and how the developer solved them? If the same problem in the environment exists the developer can just view the install log from previous install and solve the problem the same way or even make a fix for the problem in that environment. If there are any changes made to the environment manually they must be written down also. This is how we know what our company did if there will be some problems. Off course we fix the problems other people caused but then it will be charged.

6.4 Testing SharePoint

SharePoint is ready platform and that’s why it is easy to provide demonstration directly after first Sprint. In other platforms like ASP.NET developers have to build everything from scratch compared to SharePoint. Pros with SharePoint is that it has huge platform and many things ready but the cons are that you cannot build so easily continuous integration that run by itself. Continuous build is possible but running it can take up to hours if there are many sites to build. Normally the site just builds itself but with SharePoint there could be errors related to environment and context during the site provision and this is why the human assistance and testing is required if the environment is going to be presented on Sprint Preview.

The human interaction can be compensated with UI and unit tests. With Share-Point the UI test can be done quite easily with Microsoft products but it needs

more expertise than with ASP.NET. Doing the unit tests with SharePoint are more complicated because in SharePoint there is always a site and SharePoint context. The custom code mainly is for the SharePoint and unit tests must mock the SharePoint context and making unit test to SharePoint compared to ASP.NET takes five times more. There are platforms for that but unit test to SharePoint are not as common as in other platforms.

The easy way is to make test cases that test the basic functionality of the created features. Test case basically test the user story and most of all the test case is like a user story. PBL items should be in form of a user story but most of the cases they are nothing like that. Test cases are like user stories created by developer. Sometimes it is even impossible to create the user story before because the Team does not know exactly how the problem is solved and Product Owner does not know how the system work or should work. This is against the theory but if the practice does not meet the theory we must be flexible.

The Team have to decide what level the test cases are be done. One level is that the test cases are done for SharePoint administrator or developer from another project. It means that user knows about SharePoint but nothing about this project so the cases can say “Go to site B and look that content type G is there”. Other level could be that the cases are for customer that does not know about SharePoint so the same event must be written more detailed like “1. Go to site B. 2. Open site settings from the cogwheel menu. 3. Select ‘Site Content Types’. 4. Check that you find content type G under title U.”

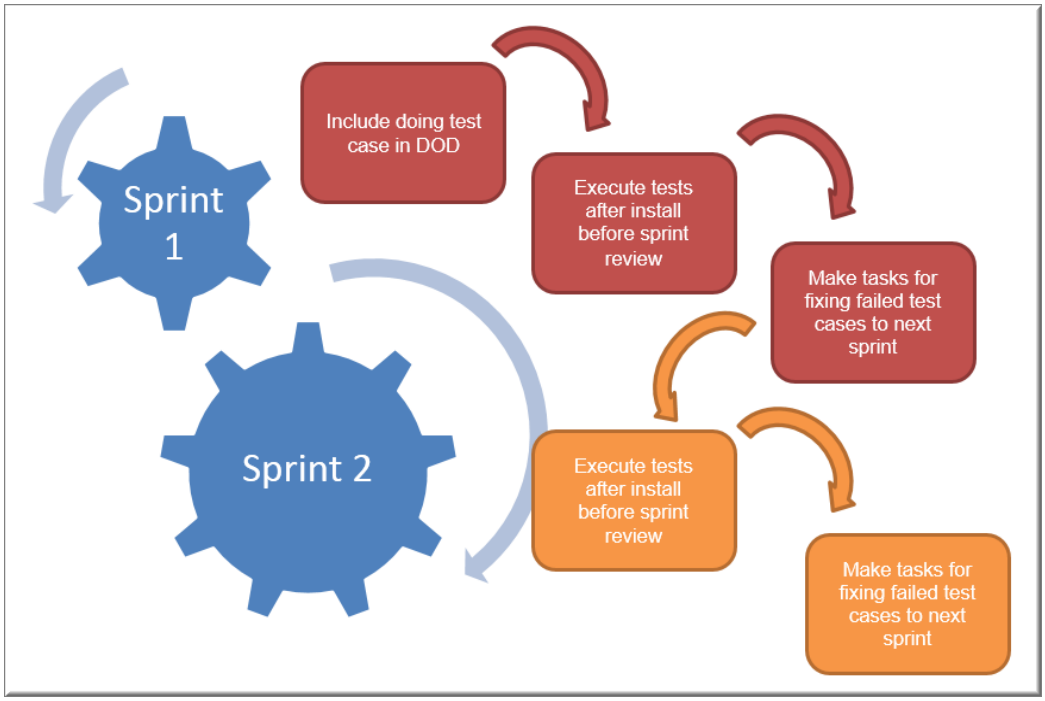


Figure 14: Binding test cases to Sprints

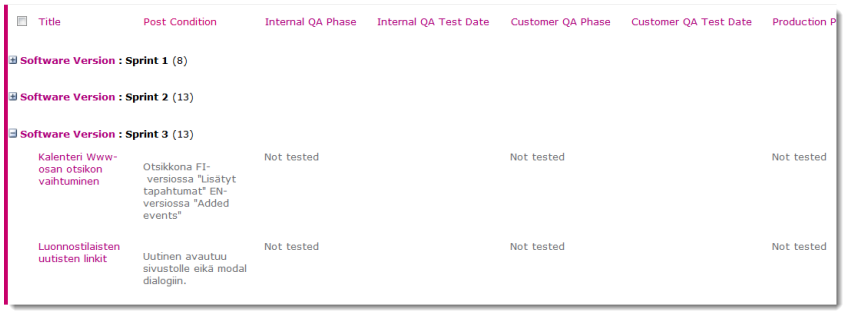
The creation of a test case must be included into DOD. When the Sprint is ending and solution is installed into one environment the testing begins. In figure 14 there are connections between test cases, executing them and Sprints. If there are failed test cases, must there be task for fixing them in the next Sprint. Then there must be new test case be done or copied from the old one that the fixed problem will be re-tested after next Sprint.

Test case list should contain columns introduced in list 1. The list will be done with SharePoint and placed into the workspace for the project or customer. The creation of the test case list should be done in the beginning of the project. Title, Software version, Related task IDs, Steps and Post conditions columns must be required fields.

Title	Text-field	Required	
Software version / Sprint	Choice	Required	Choices: Sprint 1, Sprint 2, Sprint N …
Related task IDs	Text-field	Required	
Preconditions	Note-field		
Steps	Note-field	Required	
Post conditions	Note-field	Required	
Internal QA phase	Choice		Choices: Passed, Failed, Not tested
Internal QA tester	User		
Internal QA test date	DateTime		
External QA phase	Choice		Choices: Passed, Failed, Not tested
External QA tester	User		
External QA test date	DateTime		
Production phase	Choice		Choices: Passed, Failed, Not tested
Production QA tester	User		
Production test date	DateTime		

List 1: Test case list columns

SharePoint list editing can be done simultaneously and there attachments can be added into them. Compared to old Excel sheets this is more flexible. TFS also offers test cases and test manager but since TFS is not opened to customer this is better solution so that customer can see the test cases in real time in shared workspace. All the columns can be edited for example if there are more or less environment in the project. With SharePoint list there could be various views that show different kinds of reports of test cases for example grouping by software version as shown in picture 1. Versioning should be turned on in the test case list for test case history.



Picture 1: Basic view of test case list

It might sound like explaining that I recommend to use SharePoint list and manual test cases. The fact is that our company is doing projects and not software development. The customer does not want to pay for time consumed by creating highly developed UI tests that might fail in each layout update. Our projects are flexible, the test cases are stabled to ongoing Sprints and require very little time to set up list or create test cases. The customer can learn to use the solution by going through the test cases which are quite like user stories. Usually the person who makes the instructions for the tailored solution might know about SharePoint but does not know about the project. The test cases is very easy way to see how the solution is designed to be used and what way the features will work as designed.

6.5 Maintenance

Once the project is ready and the final release is installed, tested and accepted the project is closed. The project or the solution is changed into maintenance mode. Incidents, service and change requests are handled according ITIL processes. There are still some of the features from SharePoint and Scrum process present in the maintenance mode.

The updates should be collected together by time or category if the changes are not critical and must be done immediately. Even in the hurry cases there should be the same parts of the process. There must be proper planning for the fix or change. Then there will be estimates. There should be DOD for the work so that the documentation will be updated, install script created and test case done. After install there should be install log written.

Basically all the parts of the SharePoint Scrum process are there. It is because when the development or fix cycle get longer like six months the people in the project can change. Or people involved in the project simply forgets things related to the project. It is very important that the process is same kind as the development process so all the information can be found easily. When the process is in order the updates to many years old projects is not a problem even to a beginner because there are ready instructions and scripts that does all the work. Even if

there are no development environment and the for the two hour fix the developer must create a new development environment. The developer simply gets new SharePoint development machine with correct version of SharePoint and Visual Studio. Takes the project down from TFS, adds new file for environment variables and starts the install script. After 15 minutes the developer has fully functioning development environment. If the developer does not know how something should work, he can look at the test case and repeat the steps to figure our does the solution work as the original developer designed it.

Most of the maintenance case hours goes for searching for information about the project. With proper process that is known throughout the team the case should not be time consuming or hard. It also saves the nerves of the maintenance person to have everything on its place rather than start to look from the source code what does this solution do.

6.6 Tools

Visual Studio is compulsive tool for developing SharePoint projects. If possible there should be TFS where to store the source codes. TFS also offers Product and Sprint backlogs and queries to it. When developer checks code in he can associate the change set to a PBL item or a task. The queries should be done for all Product Backlog items, Sprint items and Sprint TODO items which means a query that does not show finished or deleted PBL items.

There are example of queries in figure 15. There are multiple different queries for the whole project and part projects. In the beginning of each Sprint new queries are made and the old ones are deleted, at least the TODO-queries because they are empty after Sprint. In the figure is one query opened. There is the PBL item and tasks under it edged with red rectangles.

During Sprint the documentation is done to OneNote notebooks. After a Sprint or when Project is finished the documentation can be improved and changed into Word format if the customer needs that. All the documentation are in a SharePoint workspace. There is also test case lists.

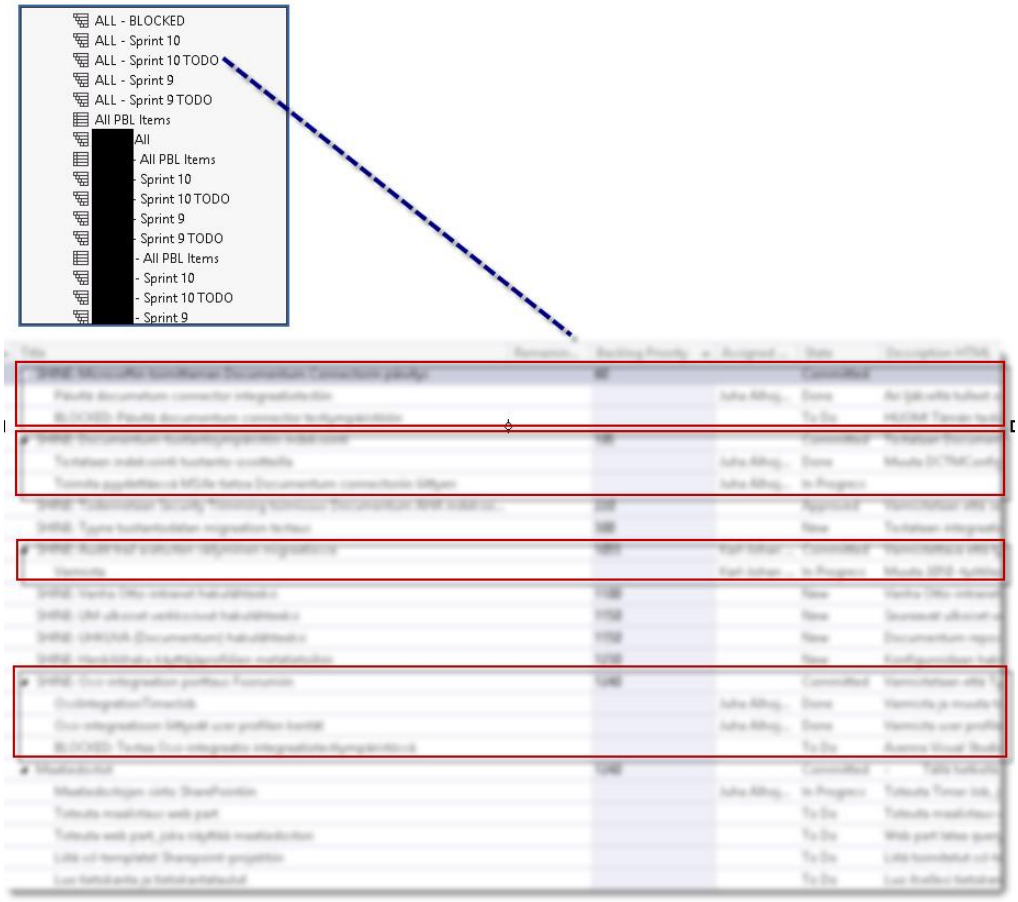


Figure 15: TFS Queries and Sprint Backlog

Development machines are VMWare virtual machines running on developer computer or virtual machines running on blade. Local virtual machines are administrated locally and the server versions are managed through remote desktop.

These are the minimum tools required in SharePoint development and Scrum process.

7 CONCLUSION

Scrum is a very effective tool to develop software or run projects. Scrum gives the framework about which meetings and roles a project should have. SharePoint has characteristics and challenges compared to other development platforms. With Scrum, SharePoint projects can be better controlled and the stress of all participants can be reduced. Scrum provides the place for estimates and tools for the Team to follow up their work. Scrum also raises the visibility of the challenges in a project of SharePoint solution to the customer.

If the members of the project team cannot understood the power of Scrum, they will not follow the rules. Scrum says that a good Team has many rules. The members must be committed to the project. The best way is to have the Team design their own work. This is the way members in the Team manage to go into the project and can motivate each other to develop an even better solution than what was originally specified. If some of the parts of the Scrum are removed, such as iterations or proper planning sessions, the motivation and effectiveness will be reduced. This is the way a project goes back to traditional project management where one person says to others what to do.

Defining and designing solutions in SharePoint is challenging because everything can be done in many ways. If everyone in the project does the way they know, there can be multiple solutions delivering almost the same functionalities. Scrum brings the Team together and makes them to discuss about the functional and technical solutions. The members of the Team expand their knowledge and experiment methods they would not try without Scrum. Daily meetings make the Team still discuss the solutions and weekly Refinements meetings bring the information for the customer also.

A few small SharePoint projects could be done with traditional project method. In most cases, Scrum seems to be the method for SharePoint projects, especially when the development team consists of people with different skill levels. I believe this thesis work achieved its goal to figure out whether SharePoint and Scrum fit

together and to outline the challenges and special characters when using SharePoint and Scrum.

Found special characters was that everybody must be in the Sprint Planning meetings because with SharePoint things can be done in so many ways that it is better that everybody is on the same page. Everybody must decide together which level of documentation and test cases they do, because the audience of the output can be a person who knows about SharePoint or has never used it. Being familiar with portals does not mean that one can use SharePoint.

On development perspective, the single PBL item per developer is a special rule with SharePoint. If the design is changed that a better method or tool is found, it must be told to others during Sprint so that there would not be multiple ways to do things. Checking in code which breaks the build is a common rule, but with SharePoint there must be the site provisioning test included in DOD. In Scrum, the difficult situations are solved together and everyone brings their expertise and contribution to the solution. With SharePoint projects, there can be different expertise area so everyone is needed.

On product perspective, the SharePoint OOTB features will be criticized. The customer cannot understand the boundaries between SharePoint OOTB features and customized features if the customization work is done well enough. Scrumbut reduces efficiency and motivation from everyone in the project. Missing meetings pushes the customer further away from SharePoint and solutions that serve them best. Test cases will catch environment failures and teach the customer to use both the customization and SharePoint. By doing the test cases during the Scrum, all the members in the project evolve at the same time and can be on the same page when the customer goes through the test cases.

8 RECOMMENDATIONS

When the chosen method for SharePoint project is not Scrum, I recommend to have parts of Scrum anyway. There should be refinement meetings with the customer and the development team, documentation and test cases made during the project, backlog containing features and tasking meetings. This is the way customer sees and hears what the nature of SharePoint is and why some estimates are what they are. Sometimes there are ideas to make things even better than originally designed. Visibility makes this happen and commits the Team, customer and project manager to the project more than in traditional method. In traditional method developers and the customer might never meet. Then all information is second or third hand information.

Documentation during the project reduces gaps in the documentation. Documentation is not a popular task and that is why it should be distributed among the project. Doing this everyone gets used to make and maintain the documentation. Test cases make the developer think what they are doing and how this module should be used. It also prevents bugs in different environments and presenting the solution to the customer goes more smoothly when there are demo cases which are already tested and functional. The customer can also have instructions how to use the modules from the test cases.

Splitting the project into features and refining them gets everyone involved into the project. Giving the developers the ability to design the technical side commits them to the project much more than if they have 100page specifications and a time limit when it should be done. Designing goes faster when there are multiple persons doing it. Individual tasks and estimates for tasks give the developer time to work in peace, which reduces bugs and problems. Everyone knows that nothing should be done in hurry. Having feature and task lists gives the opportunity to achieve the goal, more than just work 100 pages of specifications in three months.

All the things mentioned already are my recommendations and the core power of doing SharePoint projects with Scrum. Scrum brings so much more to the working and reduces SharePoint bugs afterwards. I recommend trying Scrum as it is, and follow my SharePoint Scrum Process method in chapter six. After one project in which Sprints are full or almost full Scrum, try another project without Scrum or strip things away. Do not start experimenting with Scrumbut, otherwise the project does not give the advantages of Scrum and distorts the perspective of Scrum project.

SharePoint is a very complex framework. The definition phase should include SharePoint experts who know the features of SharePoint. There is no use to make some module 20 days if there are almost same feature out-of-the-box in SharePoint. When the definition is done and project is sold to customer, be sure that the Team comes to the first Sprint Planning. It is very important that the Team and the customer meet so that there will be trust and bond when doing the project, especially if there will be online meetings instead of in real life meetings.

When trying Scrum let people commit mistakes and learn from it. If following the Scrum method is too tight, the motivation will lack. During the Retrospective meetings, the team can think about what was good and what was bad. At start, choose only one thing to improve in the next Sprint. Leave room for the Team to grow together. And mostly important, if the Team does not have any Scrum education, make sure that the ScrumMaster has the certificate, proper education and experience. Remember to educate the Team on Scrum before starting the project and hopefully the customer knows about Scrum. If the customer does not know about Scrum, there cannot be a Product Owner in the customer side.

SharePoint experience is needed. It must be noticed that if there are only beginners with SharePoint the project will not be successful. The project will teach the Team the Scrum method and the knowledge about SharePoint is shared. Scrum does not make juniors to seniors, it is not silver bullet for everything. Scrum will help and teach the Team members to share information and design together. Scrum also motivates the members if someone likes to educate himself and

shines in the planning meetings. Let great developers be in the project because the rest will follow the lead.

REFERENCES

Hillier S, Pattison T, 2013. Microsoft SharePoint 2013 App Development, O'Reilly Media Inc.

Kendrick, T 2010. The project management tool kit: 100 tips and techniques for IT project management

Microsoft 2010. Designing and Developing Microsoft SharePoint Server 2010 Applications. Product number 10232A

Pattison T, Connel A, Hillier S, Mann D, 2011. Inside Microsoft SharePoint 2010, O'Reilly Media Inc.

Pham A, Pham P-V, 2012. Scrum in Action: Agile Software Project Management and Development

Roine J, Anttila J, 2013. SharePoint-opas, SharePoint HPR: Hyvät, pahat ja rumat

Schwaber K, Sutherland J, 2011. The Scrum Guide. Retrieved 1.9.2012, available at

http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20FI.pdf

SharePoint 2013 Overview, 2014. Retrieved 19.3.2014, available at

http://office.microsoft.com/en-us/sharepoint/sharepoint-2013-overview-collaboration-software-features-FX103789323.aspx

Sutherland J, 2010. Scrum Handbook. Retrieved 1.9.2012, available at

http://jeffsutherland.com/scrumhandbook.pdf

Wysocki, R, 2011. Effective Project Management: Traditional, Agile, Extreme (6th Edition). Hoboken, NJ: Wiley.