

LIGA DE ENSINO DO RIO GRANDE DO NORTE
CENTRO UNIVERSITÁRIO DO RIO GRANDE DO NORTE
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANDRÉ LUÍS MONTEIRO PASSOS

**SCRUM E TEAM FOUNDATION SERVER: UM ESTUDO DE CASO DE
IMPLANTAÇÃO**

NATAL/RN
2013

ANDRÉ LUÍS MONTEIRO PASSOS

**SCRUM E TEAM FOUNDATION SERVER: UM ESTUDO DE CASO DE
IMPLANTAÇÃO**

Trabalho de Conclusão de Curso
apresentado ao Centro Universitário
do RN como requisito final para
obtenção do título de Bacharel em
Sistemas de Informação

Orientador: Profº Alexandre Luiz
Galvão Damasceno

NATAL/RN
2013

Catálogo na Publicação – Biblioteca do UNI-RN
Setor de Processos Técnicos

Passos, André Luís Monteiro.

Scrum e team foundation server: um estudo de caso de implantação. /

André Luís Monteiro Passos. – Natal, 2013.

46 f.

Orientador: Profº. Me. Alexandre Luiz Galvão Damasceno.

Monografia (Graduação em Sistemas de Informação) – Centro
Universitário do Rio Grande do Norte.

1. Sistema de Informação - Monografia. 2. Scrum - Monografia. 3. Team
Foundation Server - Monografia. 4. Métodos Ágeis - Monografia. I.
Damasceno, Alexandre Luiz Galvão. II. Título.

RN/UNI-RN/BC

CDU 004.43

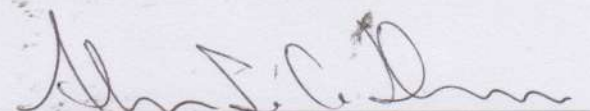
ANDRÉ LUÍS MONTEIRO PASSOS

**SCRUM E TEAM FOUNDATION SERVER: UM ESTUDO DE CASO DE
IMPLANTAÇÃO**

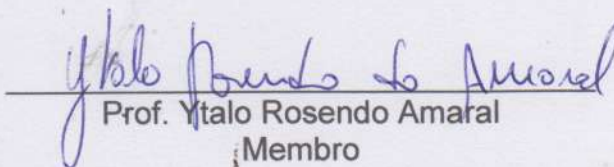
Trabalho de conclusão de curso
apresentado ao curso de Bacharelado
em Sistemas de Informação do Centro
Universitário do RN – UNI-RN como
requisito final para obtenção do título
de Bacharel em Sistemas de
Informação.

Aprovado em: ____/____/____

BANCA EXAMINADORA



Prof. Alexandre Luiz Galvão Damasceno
Orientador



Prof. Ytalo Rosendo Amaral
Membro

Prof. Itamir de Moraes Barroca Filho
Membro

*Dedico este trabalho a minha esposa,
minhas mães, pai, irmãos e
especialmente ao Alder (1960 – 2013 †)
que sempre me estimularam a correr atrás
desse sonho que é a concluir a
graduação.*

AGRADECIMENTOS

Agradeço primeiramente a Deus por me dar força e saúde a buscar mais essa conquista de concluir a graduação;

À minha esposa Larissa que sempre me apoiou nos momentos mais difíceis, pela compreensão nos momentos em que não pude estar presente durante todo curso e principalmente pelo amor e dedicação à mim.

À minha mãe Teresa que é um exemplo de vida para mim. Sempre me incentivou a buscar este sonho de obter uma formação e que serei eternamente grato pela incondicional dedicação para que nada me falte, principalmente o estudo.

À minha segunda mãe Sandra que apesar da distância sempre esteve presente em grande parte da minha educação.

Ao meu pai Geraldo pelo amor que me foi dado em toda minha vida e por sempre me apoiar nos momentos difíceis e principalmente por me direcionar nos caminhos da vida.

Um agradecimento mais que especial ao Alder, um verdadeiro pai, uma pessoa que jamais esquecerei e que serei eternamente grato por cobrar os estudos e me ensinar valores imprescindíveis a qualquer pessoa.

Agradeço a todos meus familiares e amigos que acreditaram em mim e que contribuíram com a minha formação. Finalmente eu consegui!

RESUMO

Este trabalho visa abordar um estudo de caso realizado em uma fábrica de software sobre a implantação da metodologia ágil Scrum e o Team Foundation Server (TFS). Apresentando como é possível utilizar ambos em um ambiente corporativo com múltiplas equipes de desenvolvimento que utilizam linguagens e ferramentas de desenvolvimento diferentes. Realizou-se uma revisão bibliográfica os conceitos do Scrum como papéis e responsabilidades, artefatos e cerimônias, relacionando-os com o processo empírico de desenvolvimento de software. Foi abordado também conceitos de Application Lifecycle Management (ALM), seus pilares, fases e disciplinas, estes são essenciais para compreender o que é o TFS. Além de detalhar os pontos mais importantes da arquitetura e funcionamento da ferramenta TFS. Finalmente, apresenta-se um estudo de caso sobre a implantação do TFS juntamente com o Scrum detalhando as customizações realizadas na ferramenta para atender o processo de desenvolvimento da fábrica de software. Para a fábrica de software os resultados obtidos foram excelentes uma vez que foi possível unificar o processo das equipes de desenvolvimento e observou-se uma melhora na produtividade dos times.

Palavras-Chave: Scrum. Team Foundation Server. Métodos ágeis.

ABSTRACT

This work seeks to address a case study conducted in a software company on the deployment of agile methodology Scrum and Team Foundation Server (TFS). Showing how use both in a corporate environment with multiple development teams that use different languages and development tools. Performed a literature review of the concepts like Scrum roles and responsibilities, artifacts and ceremonies, relating them to the empirical process of software development. Concepts of Application Lifecycle Management (ALM), its pillars, stages and disciplines were also discussed, these are essential to understanding what is TFS. Well as to describe the main points of the architecture and operation of the tool TFS. Finally, presents a case study on the deployment of TFS with the Scrum detailing the customizations made to the tool to meet the development process of software company. For the software company the results were excellent as it was possible to unify the process of development teams and there was an improvement in the productivity of teams.

Keywords: Scrum. Team Foundation Server. Agile Methods.

LISTA DE FIGURAS

Figura 1 - Ciclo do Scrum.....	18
Figura 2 - Gráfico do Sprint Burndown	22
Figura 3 - Pilares do ALM.....	27
Figura 4 - Fases do ALM.....	28
Figura 5 - Funcionalidades do TFS	31
Figura 6 - Product Backlog Item	39
Figura 7 - Bug.....	39
Figura 8 - Fluxo de estados.....	40
Figura 9 - Quadro de tarefas	43

LISTA DE GRÁFICOS

Gráfico 1 – Produtividade (horas/ponto).....	41
Gráfico 2 - Bugs corrigidos e pendentes	41
Gráfico 3 – Velocidade da equipe	42
Gráfico 4 - Esforço durante as sprints	43

LISTA DE TABELAS

Tabela 1 - Novos campos	37
Tabela 2 - Campos adicionados 2	38

LISTA DE ABREVIATURAS E SIGLAS

ALM – Application Lifecycle Management

API – Application Programing Interface

ROI – Retorno de Investimento

SDLC – Systems Development Life-Cycle

TI – Tecnologia da Informação

XP – Extreme Programing

WIT – Work Item Tracking

SUMÁRIO

1 INTRODUÇÃO	14
1.1 Motivação	14
1.2 Objetivos	14
1.3 Organização do trabalho	15
2 SCRUM	16
2.1 Papeis e Responsabilidades	18
2.2.1 Dono do produto / Product Owner	18
2.2.2 Mestre Scrum / Scrum Master	19
2.2.3 Equipe de Desenvolvimento / Scrum Team	19
2.3 Artefatos	20
2.3.1 Product Backlog	20
2.3.2 Sprint Backlog	21
2.3.3 Sprint Burndown	21
2.4 Reuniões / Cerimônias	22
2.4.1 Estimativa	22
2.4.2 Sprint Planning	23
2.4.2.1 Sprint Planning 1	23
2.4.2.1 Sprint Planning 2	23
2.4.3 Reunião diária	24
2.4.4 Review	24
2.4.5 Retrospectiva	24
3 APPLICATION LIFECYCLE MANAGEMENT (ALM)	26
3.1 Pilares	26
3.2 Fases	27
3.3 Disciplinas	28
3.3.1 Gerência de requisitos	29
3.3.2 Gerência de configuração	29
3.3.3 Montagem e Integração	29
3.3.4 Engenharia de Distribuição	29
3.3.5 Gerenciamento de Defeitos	30
3.3.6 Teste unitário, Integrado e Regressão	30
3.3.7 Análise de código	30
3.3.8 Teste de Sistema	30
4 TEAM FOUNDATION SERVER	31
4.1 Arquitetura	32

4.2 Item de Trabalho (Work Item)	33
5 ESTUDO DE CASO	35
5.1 O Processo	35
5.2 Motivação	36
5.3 A Implantação do Team Foundation Server	36
5.4 Resultados	41
6 CONCLUSÃO	44
REFERÊNCIAS	45

1 INTRODUÇÃO

Atualmente, devido à alta complexidade inerente ao desenvolvimento de software, os processos definidos, como o processo em cascata, fornecem um nível de flexibilidade menos adequado para que se obtenha alta produtividade e qualidade no produto final com as atuais práticas de engenharia.

O mercado de Tecnologia da Informação (TI) está cada vez mais competitivo, onde a rapidez na entrega de sistemas com qualidade, flexibilidade e rápida respostas às mudanças torna-se cada vez mais desejada, o SCRUM surge como uma metodologia ágil para gerenciar projetos baseado em inspeção e adaptação (CRUZ, 2008).

Desenvolver um software é uma tarefa difícil, provado pelos inúmeros projetos que falharam. Um fator essencial para o sucesso de qualquer equipe de desenvolvimento de software é o quão bem os membros do time se comunicam umas com as outras, bem como com as pessoas que queriam o software desenvolvido em primeiro lugar (BLANKENSHIP, 2013).

1.1 Motivação

A metodologia Scrum é uma metodologia ágil e leve que propõe justamente o que as empresas atuais buscam: minimizar custos, tempo e principalmente garantir a qualidade do sistema entregue. Porém para gerenciar processos complexos baseados no Scrum é necessário uma ferramenta para gerenciar e acompanhar o processo de desenvolvimento. Este trabalho propõe um estudo de caso em uma empresa que já utiliza o Scrum em seu processo de desenvolvimento, porém necessita de obter mais informações sobre os resultados de das equipes de desenvolvimento.

1.2 Objetivos

Este trabalho tem por objetivo avaliar a aplicação do método ágil Scrum utilizando como ferramenta de gerenciamento de projetos o Team Foundation Server. Para isso, foi proposto um estudo de caso da utilização do método ágil

Scrum em uma fábrica de software localizada em Natal-RN, onde são utilizadas duas ferramentas para gerenciar o processo das equipes de desenvolvimento.

1.3 Organização do trabalho

Este documento foi dividido em 5 seções envolvendo uma introdução sobre a metodologia ágil Scrum, uma explicação sobre o que é o Team Foundation Server, um estudo de caso da aplicação do Scrum e as conclusões obtidas com este trabalho.

A seção 2 aborda os conceitos do Scrum, o ciclo de desenvolvimento, papéis e responsabilidades, artefatos e cerimônias.

Na seção 3 é realizada uma introdução sobre o que é Application Lifecycle Management (ALM), seus conceitos, fases e pilares.

Na seção 4 é apresentado conceitos sobre o Team Foundation Server, seu funcionamento e como é possível customizá-lo para atender os requisitos de qualquer processo de desenvolvimento.

Na seção 5 apresenta o estudo de caso realizado detalhando as customizações realizadas no Team Foundation Server para que fosse possível sua utilização na empresa.

Por fim na seção 6 é apresentado a conclusão do trabalho, com os resultados obtidos no estudo de caso.

2 SCRUM

O termo Scrum foi aplicado pela primeira vez no processo de fabricação de automóveis na Toyota por Takeuchi e Nonaka no artigo The New Product Development Game (TAKEUCHI & NONAKA, 1986), onde pequenas equipes pequenas e multidisciplinares produziam melhores resultados e associaram estas equipes à formação do Scrum, uma jogada do Rugby, esporte coletivo semelhante ao futebol americano.

Em 1996 Ken Schwaber e Jeff Sutherland definiram o Scrum como um método ágil que aceita que o desenvolvimento de software é imprevisível e formaliza a abstração, sendo aplicável a ambientes voláteis (SCHWABER, 2004). É uma abordagem empírica baseada na flexibilidade, adaptabilidade e produtividade emonde escolha das técnicas de desenvolvimento ficam a cargo do time. Segundo Cruz (2008) as principais características do Scrum são:

- Processo ágil para gerenciar e controlar o desenvolvimento de projetos
- Aumenta a comunicação e maximiza a cooperação
- Escalável, desde projetos pequenos até grandes projetos em toda organização
- Controla o caos resultante de necessidades e interesses conflitantes
- Uma ponte para outras práticas de engenharia de software como o Extreme Programming (XP)

De acordo com Schwaber (2004), o Scrum assume a premissa que é o desenvolvimento de software é muito complexo e imprevisível para ser totalmente planejado desde o início e que ao invés de realizar um planejamento detalhado de todo projeto deve-se usar o modelo empírico de controle de processos. Três pilares sustentam qualquer implementação de controle de processos empíricos, são eles: transparência, inspeção e adaptação.

O primeiro pilar é a da transparência que garante que aspectos do processo que afetam o resultado devem ser visíveis para aqueles que gerenciam os resultados. Esses aspectos não apenas devem ser transparentes, mas também o que está sendo visto deve ser conhecido. Isto é, quando alguém que inspeciona um

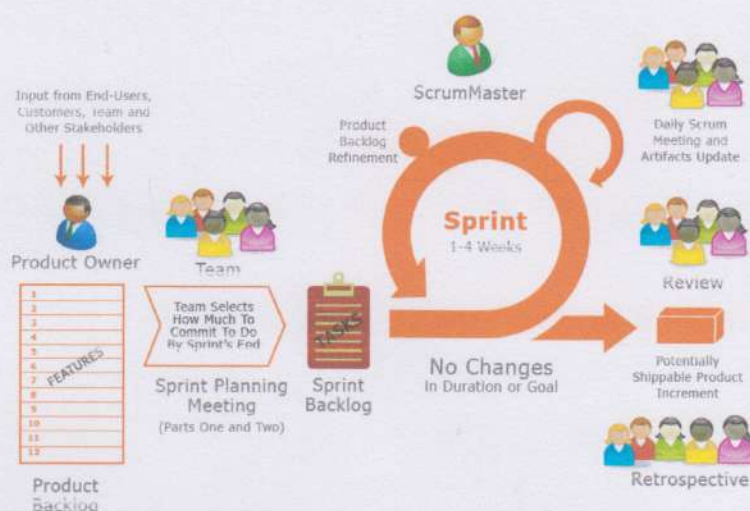
processo acredita que algo está pronto, isso deve ser equivalente à definição de pronto utilizada.

O segundo pilar é a inspeção, pois diversos aspectos do processo devem ser inspecionados com uma frequência suficiente para que variações inaceitáveis no processo possam ser detectadas. A frequência da inspeção deve levar em consideração que qualquer processo é modificado pelo próprio ato da inspeção. O problema acontece quando a frequência de inspeção necessária excede a tolerância do processo à inspeção. Os outros fatores relacionados são a habilidade e a aplicação das pessoas em inspecionar os resultados do trabalho.

A adaptação é o terceiro pilar, pois muitas vezes apartir da inspeção, é observado que um ou mais aspectos do processo estão fora dos limites aceitáveis e que conseqüentemente o produto resultante será inaceitável. Neste caso, ajustes deverão ser feitos no processo ou no software sendo desenvolvido. Esses ajustes devem ser efetuados o mais rápido possível para minimizar desvios posteriores.

No Scrum existem três pontos de inspeção e adaptação. A Reunião Diária é utilizada para inspecionar o progresso em relação a meta da Sprint (tempo de 1 a 4 semanas onde o produto é desenvolvido) e para realizar adaptações que otimizem o trabalho a ser realizado no dia. As reuniões de Review e a Sprint Planning são utilizadas para inspecionar o progresso em relação a meta do Release para entrega e para fazer adaptações que agreguem valor a próxima Sprint, por último a Retrospectiva, utilizada para avaliar a Sprint anterior e definir as adaptações necessárias para maximizar o valor da próxima Sprint. A figura abaixo mostra o fluxo principal do Scrum.

Figura 1 - Ciclo do Scrum



Fonte: N-Axis Software (2013)

2.1 Papeis e Responsabilidades

O scrum é um framework interativo e incremental cujas atividades são assumidas por três papéis: o Dono do Produto, o Scrum Master e o Time (SCHWABER, 2004).

2.2.1 Dono do produto / Product Owner

O Dono do Produto é o responsável pela visão de negócios e por representar os interesses das pessoas que apostaram no projeto e no produto resultante, é ele quem define e prioriza o Product Backlog (MARTINS, 2007).

Suas principais responsabilidades são:

- Definir as funcionalidades do produto
- Priorizar o Product Backlog
- Aceitar ou rejeitar o resultado dos trabalhos
- Maximizar o Retorno do Investimento (ROI) do produto
- Ajustar os recursos e priorizar as tarefas a cada sprint
- Decidir sobre a data do termino

2.2.2 Mestre Scrum / Scrum Master

O Scrum Master apesar de ocupar uma posição similar à de um gerente de projeto, porém não é, mas sim um líder, facilitador e mediador. Está sempre em contato com o Product Owner (SANCHEZ, 2007).

Suas principais responsabilidades são:

- Garantir que o processo do Scrum seja seguido por todos os envolvidos no processo;
- Remover os obstáculos da equipe de desenvolvimento;
- Manter as informações sobre o progresso do time durante a Sprint atualizadas e visíveis para todos;
- Facilita a comunicação entre o time e o Dono do Produto
- Auxiliar o Dono do Produto como maximizar o ROI
- Assegurar que a equipe de desenvolvimento funcione plenamente e seja produtiva

De acordo com Devin (2013), devido ao fato do Scrum Master possuir todas essas responsabilidades citadas acima, pode-se chama-lo também de coordenador de projeto.

2.2.3 Equipe de Desenvolvimento / Scrum Team

A equipe de desenvolvimento são os profissionais que realizam o trabalho, transformando os itens do Product Backlog em incrementos do produto em uma versão usável do produto ao final de cada sprint.

Segundo Yoshima (2007), as equipes são auto gerenciadas, auto organizadas e multifuncionais, ou seja, cada membro da equipe deve ter o conhecimento técnico sobre todo o processo de desenvolvimento do produto. De tal maneira, em projetos de desenvolvimento de software, a equipe de projeto deve ser composta de pessoas capazes de analisar a solução, codificá-la e testá-la sem necessitar de outros times ou outras pessoas.

Possui entre 4 e 9 integrantes de forma a otimizar a comunicação e a produtividade, porém o recomendado são 7 pessoas (MARTINS, 2007).

É importante ressaltar que a responsabilidade pela qualidade do produto é do time, ele deve verificar se suas práticas de desenvolvimento são compatíveis com as necessidades de qualidade e disponibilização da aplicação. No momento em que o time encontra necessidades de refatorar seu código ou construir novos testes ele deve notificar tais necessidades ao Product Owner.

2.3 Artefatos

De acordo com Schwaber (2004), o Scrum introduz artefatos usados ao longo do seu fluxo de desenvolvimento que ajudam a monitorar o processo e o desenvolvimento do produto, bem como, apoiam o acompanhamento do progresso do mesmo.

2.3.1 Product Backlog

De acordo com Ken e Colin (2009), o Product Backlog é o artefato mais fundamental no Scrum, pois é ele que indica o que a equipe irá desenvolver. Trata-se de uma lista priorizada pelo Product Owner que incluem os requisitos funcionais e não funcionais do produto que está sendo desenvolvido no projeto.

O Product Backlog é dinâmico; mudando contante para identificar o que o produto necessita para ser mais apropriado, competitivo e útil. O Product Backlog nunca está completo, ele evolui tanto quanto o produto e o ambiente no qual ele será utilizado evoluem e existirá enquanto o produto também existir (SCHWABER; SUTHERLAND, 2011).

O Product Backlog é, geralmente, ordenado por valor, risco, prioridade e necessidade. Os itens no topo da lista determinam as atividades de desenvolvimento mais imediatas, devem ser mais claros e mais detalhados que os itens de ordem mais baixa. Estimativas mais precisas são feitas com base na maior clareza e maior detalhe; Quanto menor a ordem na lista, menor os detalhes.

A constante preparação e o cuidado dado ao Product Backlog é a ação de adicionar detalhes, estimativas e ordem aos itens. Este é um processo contínuo em

que o Product Owner e a Equipe de Desenvolvimento colaboram nos detalhes dos itens.

De acordo com Schwaber e Sutherland (2011) preparar o Product Backlog é uma atividade de tempo parcial, durante a Sprint, entre o Product Owner e a Equipe de Desenvolvimento. Geralmente a Equipe de Desenvolvimento tem o domínio do conhecimento para realizar a preparação por si própria.

A Equipe de Desenvolvimento é responsável por todas as estimativas. O Product Owner deve influenciar o Time, ajudando no entendimento e nas decisões conflituosas de troca, mas as pessoas que irão realizar o trabalho fazem a estimativa final.

2.3.2 Sprint Backlog

O Sprint Backlog é um conjunto de itens do Product Backlog selecionados para a Sprint, juntamente com o plano de entrega do incremento do produto e atingir o objetivo da Sprint. O Sprint Backlog é a previsão da Equipe de Desenvolvimento sobre qual funcionalidade estará no próximo incremento e do trabalho necessário para entregar essa funcionalidade (SCHWABER; SUTHERLAND, 2011).

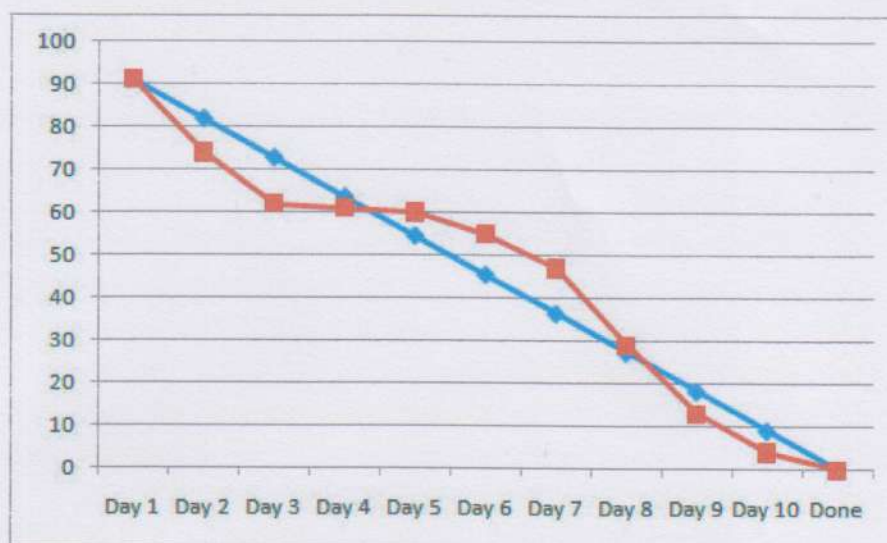
A cada nova Sprint é realizada uma seleção dos itens do Product Backlog de acordo com que a equipe aderiu para a Sprint (MARÇAL, 2009).

Somente a Equipe de Desenvolvimento pode alterar o Sprint Backlog durante ele é altamente visível, uma imagem em tempo real do trabalho que a Equipe de Desenvolvimento planeja completar durante a Sprint, e pertence exclusivamente aos mesmos.

2.3.3 Sprint Burndown

O Sprint Burndown é um artefato do Scrum que representa graficamente o trabalho restante para finalizar a Sprint. Durante as reuniões diárias o Scrum Master coleta as informações sobre a conclusão dos itens do Sprint Backlog gerando um gráfico do progresso da sprint.

Figura 2 - Gráfico do Sprint Burndown



Fonte: CertSchool.com (2013)

O gráfico da figura 2, possui no eixo X o número de dias da Sprint e no eixo Y a estimativa em horas do time para completar todos os itens do Sprint Backlog. Este número de horas vai caindo a medida que alguns itens são concluídos (linha irregular do gráfico). Uma linha é traçada do maior valor de X até o maior valor de Y que serve como guia da produtividade do time (SCHWABER; SUTHERLAND, 2011).

2.4 Reuniões / Cerimônias

O Scrum possui diversas reuniões realizadas em diferentes etapas do projeto, o que facilita o gerenciamento e controle do mesmo. Em todas as reuniões é definido um Time Box que é o tempo máximo que cada reunião pode durar.

2.4.1 Estimativa

A reunião de estimativa acontece antes do planejamento da Sprint, nesse encontro participam o Product Owner, Scrum Master e o Time.

O Product Owner apresenta ao time quais itens do Product Backlog que ele deseja que sejam estimados e explica os detalhes de cada um para que a equipe possa realizar o método de estimativa, no Scrum o método mais utilizado é o Planning Poker. Enquanto todos os integrantes não estiverem de acordo com o

resultado estimado para o item, a estimativa não é finalizada, se houver necessidade o item é explicado e discutido novamente pelos participantes.

Nessa reunião o Scrum Master deve apenas assegurar que a mesma está sendo realizada de acordo com as regras do Scrum e em momento algum deve influenciar a equipe na estimativa dos itens (FELIX, 2010).

2.4.2 Sprint Planning

A reunião do Sprint Planning duas atividades são realizadas antes do início da sprint, são elas: Sprint Planning 1 e Sprint Planning 2.

2.4.2.1 Sprint Planning 1

A Sprint Planning 1 é a principal reunião da Sprint, nela participam o Product Owner, Scrum Master e o Time, porém além desses participantes podem ser convidados outros profissionais que estejam interessados no resultado do projeto.

De acordo com Vailati (2010), o foco dessa reunião é que além do Product Owner apresentar as funcionalidades que serão desenvolvidas deverão ser esclarecidas as dúvidas levantadas pelo time, de maneira que o time consiga absorver todo o conhecimento necessário para desenvolver as funcionalidades apresentadas.

Nessa reunião o time deverá escolher quais itens do Product Backlog serão desenvolvidos na Sprint de acordo com a prioridade definida pelo Product Owner. Todo time deve chegar num acordo de que terão condições de entregar todos os itens aceitos.

2.4.2.1 Sprint Planning 2

No Sprint Planning 2 o Time se reúne para quebrar as histórias aceitas na Sprint Planning 1 em tarefas menores, conhecidas como Tasks, que possam ser realizadas em um dia ou menos, pois com isso além de identificar possíveis riscos também facilitará o gerenciamento e visualização no momento da Reunião Diária (BURGOS, 2010)

Após esta reunião é elaborado o artefato Sprint Backlog que contém a especificação de todos os itens aceitos no Sprint Planning 1 e suas respectivas Tasks, bem como o esforço necessário para realizar cada história.

2.4.3 Reunião diária

A Reunião diária é onde o Time e o Scrum Master se encontram diariamente como uma forma de melhorar a comunicação entre os participantes. Essa reunião tem duração de 15 minutos podendo se estender por mais 15. Esse encontro serve para identificar e remover impedimentos, e atualizar o artefato Sprint Burndown. De acordo com CRUZ (2008), nessa reunião cada membro do time responde a três perguntas:

1. O que você fez desde a última Reunião Diária?
2. O que você vai fazer hoje?
3. Existe algum obstáculo que impeça de realizar as tarefas?

Essa reunião é uma parte importante do processo de inspeção e adaptação do Scrum.

2.4.4 Review

Ocorre no final de cada Sprint, nesse momento o Time exibe o incremento do produto, potencialmente utilizável que foi construído de acordo com o Sprint Backlog, ao Product Owner, que é responsável por aprovar, ou não, se o incremento é válido e atende todos os requisitos especificados na história. Nessa reunião participam o Product Owner, o Scrum Master, Time e todos os interessados no projeto. O ideal é que todos os itens do Sprint Backlog sejam aprovados pelo PO. As histórias que não foram aceitas ou não foram apresentadas serão priorizadas novamente na próxima sprint.

2.4.5 Retrospectiva

Após a Review o Time e o Scrum Master se reúnem para revisar o modelo de trabalho utilizado, debater sobre os pontos positivos e negativos da Sprint e o que

pode ser melhorado e quais ações necessárias para colocar as melhorias em prática. Geralmente a Retrospectiva tem de três a quatro horas de duração (AGUIAR, 2008).

Nesta seção foram apresentados os conceitos do Scrum, o papel de cada pessoa no processo, quais artefatos são gerados pelo Scrum e as reuniões necessárias para que a comunicação seja um ponto forte entre os participantes do processo.

Na próxima seção será apresentado os conceitos de Application Lifecycle Management (ALM) ou Gerenciamento do Ciclo de Vida da aplicação. Estes conceitos são essenciais para se entender o que é o Team Foundation Server suas principais características.

3 APPLICATION LIFECYCLE MANAGEMENT (ALM)

A atividade de construção de aplicações é executada através do emprego intenso do trabalho humano, consequentemente exige das empresas somas de investimento financeiro. Parte deste investimento é consumido por atrasos de cronograma, retrabalho, testes inadequados e outros. Estima-se que em 2002, gastaram cerca 20 bilhões de dólares só com testes inadequados de software feito por desenvolvedores de software (CONDÉ, 2009 apud NIST, 2002).

Segundo Condé (2009)

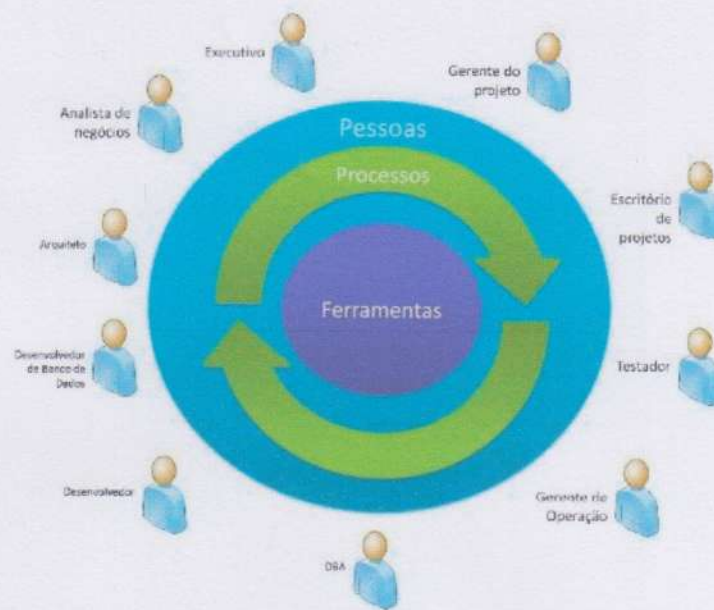
Podemos definir ALM como todo o processo que guia a vida útil de uma aplicação desde a sua concepção, passando pela construção, operação e evolução. O ALM não apenas observa qual é o método de construção, mas preocupa também em como a empresa está gastando o seu dinheiro no gerenciamento daquele ativo corporativo.

O ALM é diferente do Software Development Lifecycle (SDLC) que segundo Condé (2009) é um processo focado no desenho, criação e manutenção de aplicações. O ALM é um guia que acompanha toda a vida da aplicação, tendo o SDLC como uma das fases do ciclo de vida (CONDÉ, 2009)

3.1 Pilares

O ALM é estruturado em cima de três pilares que se complementam, são eles: Pessoas, Processos e Ferramentas. Quando unidos fornecem os recursos necessários para que as organizações possam gerenciar o ciclo de vida de suas aplicações, como apresentado na figura 3.

Figura 3 - Pilares do ALM



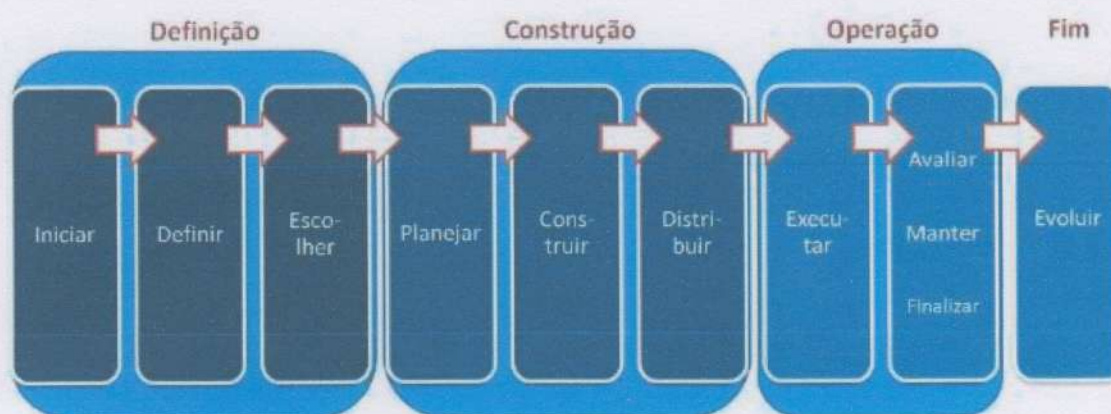
Fonte: Condé (2009)

- **Pessoas:** Segundo Heringer (2012) as pessoas são o elo entre ferramentas e processos. Cada pessoa possui um papel específico que quando bem treinadas e capacitadas formam uma cola que une adequadamente as ferramentas e os processos do ALM (CONDÉ, 2009).
- **Processos:** É o conjunto de boas práticas, artefatos, guias e manuais que conduzem a construção e manutenção de uma aplicação. Estes processos incluem o levantamento de requisitos, construção e monitoramento em ambiente de produção.
- **Ferramentas:** São os meios/equipamentos/tecnologias que automatizam e facilitam a condução dos processos pelas pessoas.

3.2 Fases

Segundo Condé (2009) o ALM possui 4 fases: Definição, Construção, Operação e Final. Na figura 4 pode-se observar as fases e suas respectivas etapas.

Figura 4 - Fases do ALM



Fonte: Condé (2009)

Na fase Definição, procura identificar quais as necessidades e motivações que uma empresa tem (CONDÉ, 2009). Nessa fase serão alocados os recursos iniciais (processos, ferramentas e pessoas), definição das estratégias, métodos, ferramentas e tecnologias.

Na fase Construção é onde ocorre a execução do plano definido na fase anterior. Usam-se disciplinas de gerenciamento de projeto para conduzir o plano. Neste ponto pode ser utilizado qualquer metodologia ou modelo de maturidade para gerenciar o andamento da construção da aplicação.

A fase de Operação se dá no momento em que a aplicação está construída e será distribuída. Tal como a fase de construção, também existem no mercado diversos modelos/frameworks para operacionalizar a infraestrutura necessária nessa fase, como, ITIL, COBIT e ISO 20.000 (CONDÉ, 2009).

A fase Final, segundo Condé (2009) é o momento em que a empresa pode evoluir a aplicação, substituindo por uma versão mais nova e adicionar recursos no ambiente de produção.

3.3 Disciplinas

O uso de disciplinas permite identificar as entradas e resultados esperados em cada etapa do ALM, bem como quem são os envolvidos (CONDÉ, 2009).

3.3.1 Gerência de requisitos

Segundo Condé (2009), requisitos são a expressão do que a aplicação deve fazer e o que é esperado dela em momento de execução. Um requisito pode ser entendido como a descrição do comportamento de um sistema.

O gerenciamento de requisitos é a prática de documentar e manter a rastreabilidade dos requisitos ao longo do ciclo de vida da aplicação.

3.3.2 Gerência de configuração

Durante a construção de um software são gerados diversos artefatos, como código-fonte, documentos e planilhas. A disciplina de gerência de configuração é responsável por manter e gerenciar estes artefatos, além da rastreabilidade e versionamento dos mesmos.

3.3.3 Montagem e Integração

Todo software é composto por diversos módulos, componentes e controles. A disciplina de Montagem e Integração consiste na prática de unir tudo isso em um único pacote para ser testado e distribuído, este processo não deve ser visto apenas como uma tarefa de compilação, mas sim, com a integração de diversas partes (CONDÉ, 2009)

3.3.4 Engenharia de Distribuição

A disciplina de engenharia de distribuição é responsável por garantir a consistência das diversas versões da aplicação (CONDÉ, 2009). Esta disciplina trabalha em conjunto com a Gerência de Configuração para garantir que os artefatos estejam devidamente rastreáveis.

3.3.5 Gerenciamento de Defeitos

Todo software por mais que seja testado, apresenta erros, a disciplina de Gerenciamento de Defeitos consiste em coletar a ocorrência destes erros, tratar como serão corrigidas e documentar o ocorrido para evitar erros semelhantes novamente.

3.3.6 Teste unitário, Integrado e Regressão

Esta disciplina visa detectar defeitos ainda na fase de construção do software, reduzindo assim o custo de manutenção.

3.3.7 Análise de código

A disciplina de análise de código é responsável em identificar se o código foi documentado, se a formatação está correta e se está de acordo com os padrões da empresa.

3.3.8 Teste de Sistema

A disciplina de teste de sistema envolve o teste da aplicação quando completa. Estes testes procuram identificar se a aplicação construída está de acordo com os requisitos especificados.

Na próxima seção serão apresentadas as principais funcionalidades do Team Foundation Server, onde cada uma delas corresponde a uma disciplina do ALM.

4 TEAM FOUNDATION SERVER

Em outubro de 2005 a Microsoft lançou o Visual Studio Team System juntamente o Visual Studio 2005. O Team System foi construído a partir de várias tecnologias usadas internamente para desenvolver os mais complexos softwares da empresa, envolvendo não apenas desenvolvedores, mas todos os membros da equipe como administradores de banco de dados, testadores e gerentes de projeto (GOUSSET).

Após algumas pesquisas com consumidores em 2010 a Microsoft decidiu alterar o nome do produto para Visual Studio Team Foundation Server (TFS).

O Team Foundation Server é a plataforma de colaboração na essência da solução de gerenciamento do ciclo de vida de aplicativos (ALM) da Microsoft (MICROSOFT, 2012), ele fornece as funcionalidades básicas para colaboração de equipes durante o desenvolvimento do software. São elas:

- Gerenciamento de Projetos
- Gerenciamento de casos de teste
- Controle de versão
- Rastreamento de item de trabalho (WIT)
- Build automatizado
- Relatórios

Figura 5 - Funcionalidades do TFS



Fonte: Microsoft (2013)

O TFS é um produto desenvolvido especialmente para times de engenharia de software com desenvolvedores, testadores, arquitetos, gerentes de projeto, analistas de negócio e qualquer um interessado em contribuir para o desenvolvimento do projeto (BLANKENSHIP, 2013).

4.1 Arquitetura

Uma das vantagens do TFS é a sua alta escalabilidade pois ele possui duas camadas distintas que podem ser divididas fisicamente entre um ou mais servidores, são elas:

- Camada de Aplicação: Consiste em um conjunto de web services com os quais as aplicações clientes se comunicam e uma interface Web que permite a iteração do usuário sem a necessidade de instalar um cliente como o Visual Studio.
- Camada de Dados: É utilizado o SQL Server como repositório de dados da camada de aplicação, juntamente com o SQL Reporting Services para geração de relatórios, além do SQL Analysis Services como ferramenta de mineração de dados.

Team Foundation Server fornece um conjunto de recursos para auxiliar o planejamento, desenvolvimento, gerenciamento do escopo, comunicação, elaboração de relatórios e análise contínua do processo. Estes recursos incluem:

- Team Project – Armazena e organiza todos os dados utilizados na construção do projeto, código-fonte, requisitos, planos de teste e relatórios
- Modelo do processo (Process Template) – É um conjunto de arquivos XML que define o processo utilizado pelo Team Project, que inclui os tipos de item de trabalho, campos e relatórios.
- Rastreamento de itens de trabalho (Work Item Tracking) – Itens de trabalho representam requisitos, bugs e casos de teste no TFS. O rastreamento destes itens é a chave para gerenciar o projeto.
- Relatórios – Os relatórios são vitais para se gerenciar qualquer projeto, o TFS inclui vários relatórios dependendo do modelo do processo adotado e é

possível criar relatórios customizados com informações sobre cobertura de código por testes, índice de bugs registrados, erros de compilação ou qualquer outra informação armazenada no banco de dados da ferramenta.

4.2 Item de Trabalho (Work Item)

A partícula elementar do TFS é o Work Item. Segundo Blankenship (2013) o Work Item é o bloco básico da construção da capacidade de gerenciamento de projeto no TFS. Existem muitos tipos de itens de trabalho, eles podem ser definidos como uma Tarefa, História, Requisito, Bug ou Caso de Teste. Qualquer artefato que precise ser documentado em um projeto é um Work Item (GOUSSET, 2010).

Uma História se difere de um Bug pelo seu Work Item Type, estes são definidos no modelo do processo utilizado na criação do Team Project.

O Work item armazena diversos tipos de informação, porém algumas delas são pré-definidas, são elas:

- Campos (Fields) – Os Fields contém as informações sobre um Work Item, alguns são compartilhados entre qualquer tipo de Work Item, são os System Fields. Alguns exemplos de System Fields são: Title, ID, Assigned To (para quem aquele Work Item está associado, pode ser um time, um desenvolvedor ou um stackholder). Outros Fields também podem ser criados de acordo com a necessidade do processo.
- Estado (State) – Define o estado atual do Work Item num fluxo de estados pre-definidos. Por exemplo, um Bug pode estar Aberto ou Resolvido.
- Formulário (Form) – O Form é como o Work Item será apresentado ao aplicativo cliente, por exemplo o Visual Studio.
- Transições (Transition) – As transições definem o fluxo de estados permitidos aquele Work Item.
- Link – Os links são ligações entre os Work items. Estas ligações também podem ter tipos de ligações.
- Histórico (History) – Todo Work Item possui um histórico de todas as alterações realizadas em seus Fields e seus estados.

Além dos campos citados acima existem outros dois muito importantes, denominados Area e Iteration. O campo Area pode ser utilizado definir uma categoria lógica para o item de trabalho, porém é comum utiliza-lo para separar a equipe alocada para desenvolver o Work Item. Já o campo Iteration é usado para o planejamento, ele indica em uma linha de tempo, quando aquele Work Item será trabalhado, por exemplo Versão 1.0\Release 1, Versão 1.0\Release 2.

Na próxima seção será apresentado o estudo de caso realizado, detalhando quais customizações foram realizadas no TFS para que fosse possível a utilização do processo de desenvolvimento da fábrica de software na ferramenta.

5 ESTUDO DE CASO

Este trabalho avalia os resultados obtidos na aplicação da metodologia ágil Scrum, com o Team Foundation Server em uma fábrica de software em Natal-RN. O estudo de caso foi realizado, com base nos dados dos projetos entre Abril e Julho de 2013.

A fábrica de software objeto de estudo deste trabalho, atua desde 1999 desenvolvendo softwares para os mais diversos setores, porém o foco principal da empresa é o setor varejista.

5.1 O Processo

Atualmente a empresa possui duas equipes que utilizam o Scrum como metodologia de desenvolvimento. Cada equipe é composta por cinco desenvolvedores, sendo que um deles também possui o papel de Scrum Master e um Product Owner.

Cada projeto é composto de duas Sprints com 14 dias de duração. No início da Sprint, cada equipe se reúne separadamente para estimar as histórias especificadas e priorizadas pelo Product Owner, utilizando o método conhecido como Planning Poker.

Após a estimativa as equipes se reúnem com o Product Owner para negociar as histórias que serão aceitas de acordo com a prioridade e respeitando a capacidade de cada equipe.

Uma vez definido os itens que farão parte da Sprint, cada equipe é responsável por analisar as histórias e subdividi-las em tarefas menores que possam ser concluídas em no máximo um dia. Feito isso a Sprint é dada como iniciada.

No 14º dia da Sprint é realizada a reunião de Review onde cada equipe irá apresentar ao Product Owner e demais interessados as tarefas concluídas e cabe ao PO decidir se atendem aos critérios previamente especificados.

Após a Review os membros das equipes se reúnem novamente para a retrospectiva onde debatem sobre o que foi positivo naquela Sprint, o que foi ruim e o que poderia melhorar. Terminada esta reunião é iniciado o ciclo da próxima Sprint.

5.2 Motivação

Os projetos desenvolvidos na fábrica de software por utilizarem linguagens de programação e ferramentas de desenvolvimento diferentes, eram gerenciados em ferramentas distintas, dificultando o gerenciamento e a avaliação dos resultados do setor de desenvolvimento da empresa como um todo. Com isso foi constatado que seria necessário unificar o processo de gerenciamento das equipes em uma única ferramenta, para diminuir o tempo gasto com gerencia de projetos, facilitar a visualização do andamento das Sprints e consequentemente diminuir o custo da criação e manutenção dos softwares da empresa.

5.3 A Implantação do Team Foundation Server

Inicialmente foi realizado um estudo sobre as ferramentas de ALM disponíveis no mercado como Rational da IBM, Star Team da Borland, HP ALM da Hewlett-Packard, Redmine e o Team Foundation Server da Microsoft. O Redmine e TFS se destacaram por já serem utilizados na empresa e por não necessitarem de nenhum custo adicional para a adoção como ferramenta única de gestão.

O TFS foi escolhido devido a facilidade de customização, integração com qualquer ferramenta de desenvolvimento e por já possuir um BI integrado, facilitando a criação de relatórios mais complexos.

Com a ferramenta de gerenciamento escolhida, foi realizado um estudo sobre as customizações necessárias para adequar o modelo do processo Scrum disponível no TFS ao processo da empresa.

Chegou-se à conclusão que seria necessário a criação de quatro tipos de itens de trabalho adicionais: Spike, Sprint, Projeto e ReviewItem.

- Spike é um item de trabalho cujo objetivo é documentar um estudo realizado durante a Sprint. Este item de trabalho serve apenas para registrar o resultado obtido desde estudo sem gerar qualquer alteração

no código-fonte. Um exemplo de um Spike pode ser o estudo do manual de uma DLL de terceiros.

- O Sprint é o item de trabalho onde é armazenado todas as informações sobre uma Sprint, como data inicial e final, itens estimados, aceitos e entregues e atas das reuniões realizadas.
- O Projeto é utilizado para documentar as atas relacionadas à um projeto, por exemplo escopo positivo e negativo, esforço disponível, atas de abertura e fechamento.
- O ReviewItem é um item de trabalho criado com um intuito de ser uma cópia somente leitura de cada item apresentado na Review.

Para cada item de trabalho criado foi necessário definir os campos para armazenar as informações, o tipo de cada um e como essa informação seria apresentada nos relatórios (Dimension, Detail ou Measure). A tabela 1 mostra a definição dos campos de cada item de trabalho.

Tabela 1 - Novos campos

Item de trabalho	Campo	Tipo	Relatório
Spike	TempoTotal	Double	Measure
	TempoSP1	Double	Measure
Sprint	TempoSP2	Double	Measure
	TempoReview	Double	Measure
	TempoEstimativa	Double	Measure
	TempoRetrospectiva	Double	Measure
	PontosSP1	Double	Measure
	PontosSP2	Double	Measure
	PontosEntregues	Double	Measure
	BugsInicio	Integer	Measure
	BugsFim	Integer	Measure
	InicioSprint	DateTime	Dimension
	FimSprint	DateTime	Dimension
	InicioEstimativa	DateTime	Dimension
	FimEstimativa	DateTime	Dimension
	InicioSP1	DateTime	Dimension
	FimSP1	DateTime	Dimension
	InicioSP2	DateTime	Dimension
	FimSP2	DateTime	Dimension
	InicioReview	DateTime	Dimension
	FimReview	DateTime	Dimension

	InicioRetrospectiva	DateTime	Dimension
	FimRetrospectiva	DateTime	Dimension
	AtaSP1	Html	
	AtaSP2	Html	
	AtaReview	Html	
	AtaRetrospectiva	Html	
Projeto	Inicio	DateTime	Dimension
	Fim	DateTime	Dimension
	AtaAberutra	Html	
	AtaFechamento	Html	
	AceitesAbertura	Html	
	AceitesFechamento	Html	

Fonte: Próprio autor (2013)

Na etapa posterior foi customizado os itens de trabalho pré-existent do modelo do processo Scrum já disponibilizado pelo TFS e adicionados os campos conforme a tabela 2.

Tabela 2 - Campos adicionados 2

Item de trabalho	Campo	Tipo	Relatório
Product Backlog Item e Bug	TempoTotal	Double	Measure
	TempoTeste	Double	Measure
	Cliente	String	Dimension
	Sistema	String	Dimension
	Retrabalho	String	Dimension
	AposSP1	String	Dimension
	Modulo	String	Dimension
	MySuite	String	Dimension
	Prioridade	String	Dimension
Bug	Tipo	String	Dimension
Task	TempoTotal	Double	Measure
	Tipo	String	Dimension

Fonte: Próprio autor (2013)

Devido à restrição imposta pelo modelo de dados do TFS, não é possível utilizar campos do tipo lógico, por isso alguns campos como Retrabalho e AposSP1 recebem apenas dois valores Sim ou Não. A figura 6 e 7 mostram os novos campos adicionados nos itens de trabalho Bug e Product Backlog Item.

Figura 6 - Product Backlog Item

New Product Backlog Item 1: Field 'Title' cannot be empty.

Copy template URL

Tags: Add...

Iteration: DevProjeto 2013/Mês 11/Sprint 2

STATUS	DETAILS
Assigned To:	Effort:
Status: New	Business Value:
Reason: New backlog item	Backlog Priority: 50
Area: Dev/PeD	Prioridade: 4 - Normal
	Retrabalho: Não
	Pós-Sp1: Não
	Data Reunião:
	Cliente: Módulo
	Programa: MySuite

ESFORÇO

Desenvolvimento (h):

Teste (h):

DESCRIPTION STORYBOARDS TEST CASES TASKS

ACCEPTANCE CRITERIA LINKS ATTACHMENTS HISTORY

Save Save and Close Cancel

Fonte: Próprio autor (2013)

Figura 7 - Bug

New Bug 2: Field 'Title' cannot be empty.

Copy template URL

Tags: Add...

Iteration: DevProjeto 2013/Mês 11/Sprint 2

STATUS	DETAILS
Assigned To:	Effort:
Status: New	Backlog Priority: 50
Reason: New defect reported	Prioridade: 4 - Normal
Area: Dev/PeD	Severity: 3 - Medium
	Tipos:
	Pós-SP1: Não
	Data Reunião:
	Cliente: Módulo
	Programa: MySuite

ESFORÇO

Desenvolvimento (h):

Teste (h):

TEST DETAILS

Found in Build:

Integrated in Build:

STEPS TO REPRODUCE SYSTEM TEST CASES TASKS

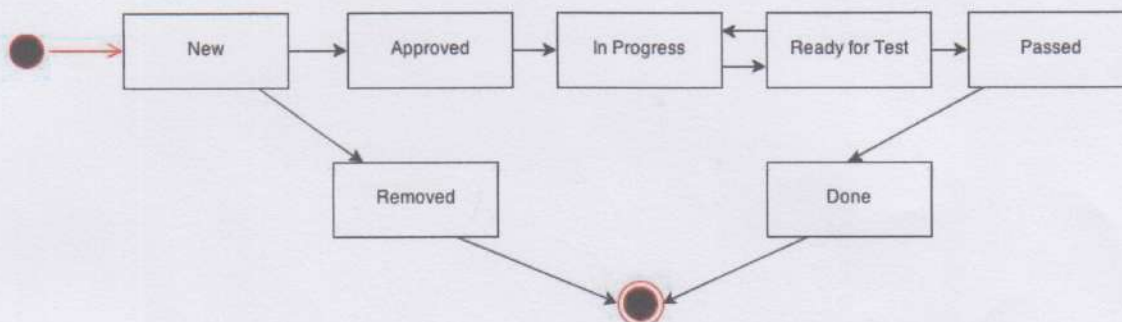
ACCEPTANCE CRITERIA ATTACHMENTS HISTORY LINKS

Save Save and Close Cancel

Fonte: Próprio autor (2013)

Também foi customizado em cada item de trabalho o fluxo de alteração entre os possíveis estados dos mesmos. O Product Backlog Item e Bug foram os que tiveram mais alterações, como mostrado na figura 8.

Figura 8 - Fluxo de estados



Fonte: Próprio autor (2013)

No TFS um item de trabalho pode ser alterado enquanto não for definido um estado final, para o processo da empresa isso não poderia ocorrer, uma vez que quando um item não é aceito ou não apresentado na Review, este é movido para a próxima Sprint, porém deveria ser possível visualizar através do item de trabalho da Sprint o estado do item não apresentado no momento da Review. Para resolver este problema foi criado o ReviewItem, um tipo de item de trabalho que contém as mesmas informações do Product Backlog Item e Bug.

Definiu-se que ao término da reunião de Review, o TFS deveria criar uma cópia de todos os itens de trabalho que a equipe se comprometeu a entregar e armazenar estas informações em um ReviewItem, que não pode ser alterado em nenhum momento. Para este processo de cópia dos itens de trabalho, foi desenvolvido uma extensão do TFS utilizando sua API e o sistema de eventos.

Uma vez que todas as customizações necessárias na ferramenta foram realizadas a etapa seguinte foi criar os relatórios e gráficos para acompanhar o resultado as Sprints.

5.4 Resultados

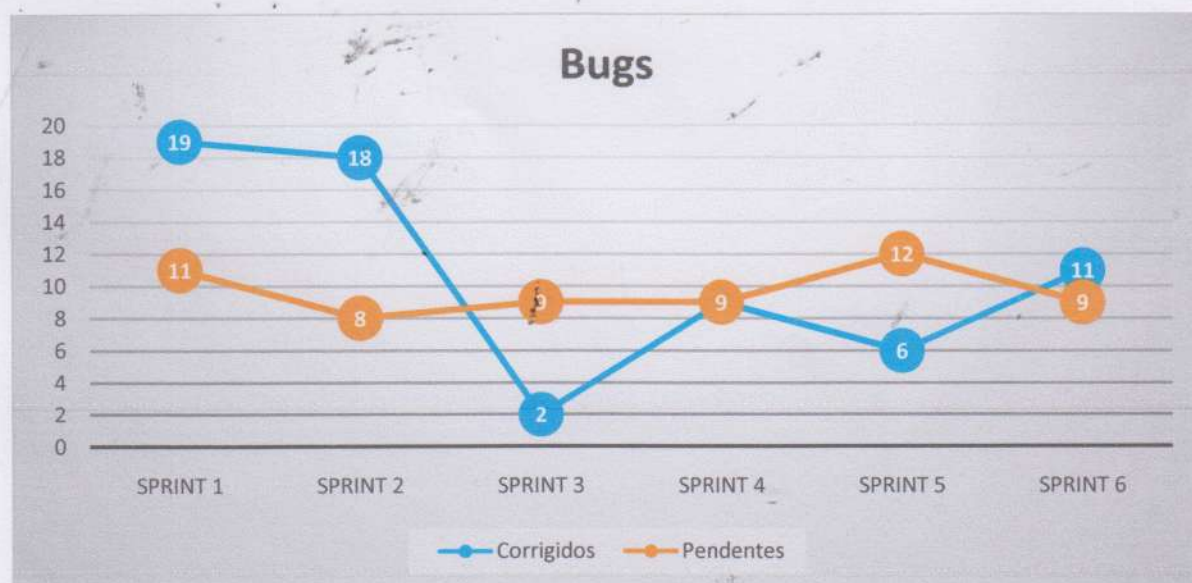
Nas Sprints analisadas por este trabalho, após a implantação do TFS, pode se observar que a tendência é melhorar a produtividade (horas/ponto) conforme mostra o gráfico 1. Quanto menor o tempo gasto para produzir um ponto, melhor a produtividade da equipe.

Gráfico 1 – Produtividade (horas/ponto)



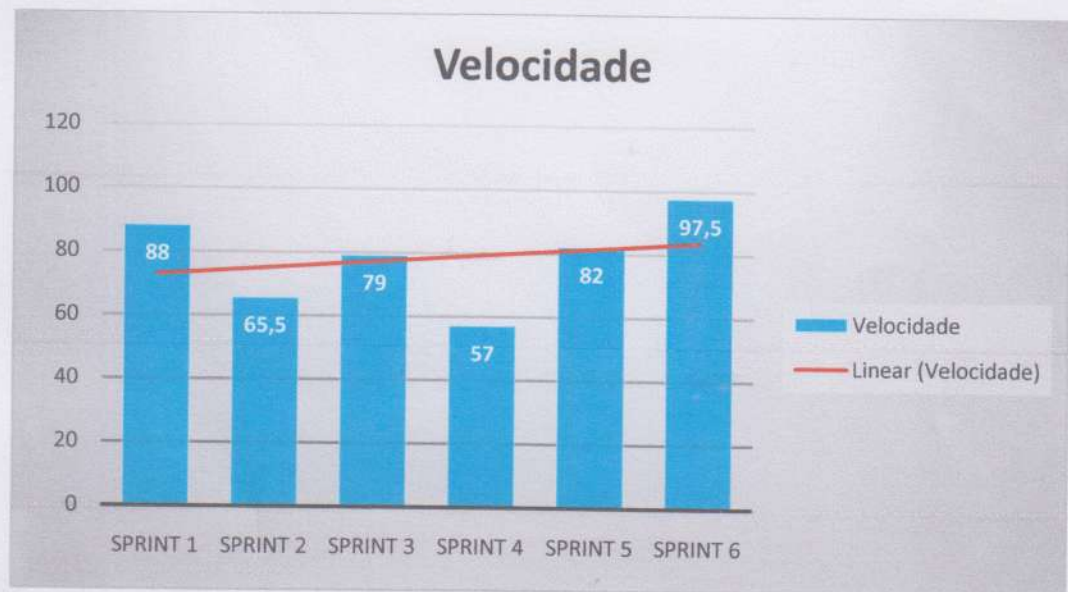
No gráfico 2 pode-se observar que muitos bugs foram corrigidos nas primeiras sprints, porém a quantidade de bugs em aberto se manteve na média. Identificando um grande número de bugs cadastrados no decorrer das sprints.

Gráfico 2 - Bugs corrigidos e pendentes



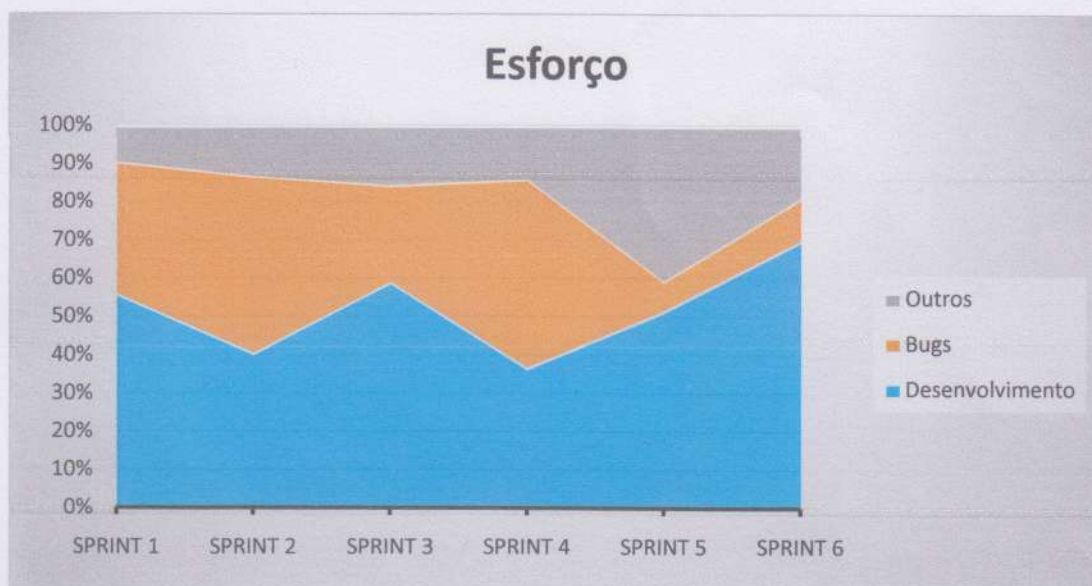
A velocidade é um fator muito importante para o planejamento das sprints futuras. Esta medida mostra quantos pontos o time é capaz de produzir em uma sprint. Analisando o gráfico 3, a equipe entregou em média 80 pontos. Nesse caso o objetivo das próximas sprints é entregar mais de 80 pontos para que haja um ganho na produtividade da equipe.

Gráfico 3 – Velocidade da equipe



O gráfico de esforço demonstra onde a equipe está utilizando o tempo da sprint. Segundo o processo da empresa, o aceitável é que 60% do esforço da sprint seja com desenvolvimento, 20% com bugs e 20% para outras atividades como reuniões, spikes e impedimentos. No gráfico 4 nota-se que em duas sprints (2 e 4) o esforço com desenvolvimento ficou bem abaixo do aceitável, mostrando que a equipe tem agregado pouco valor ao produto.

Gráfico 4 - Esforço durante as sprints

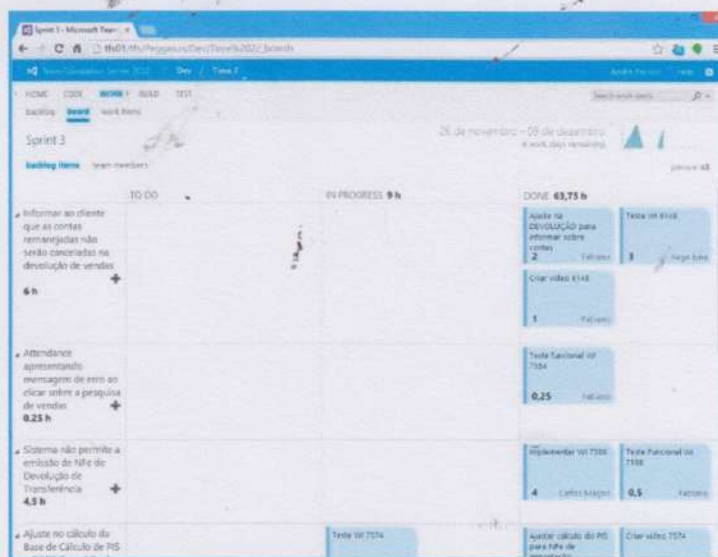


Fonte: Próprio autor (2013)

Após a implantação do TFS a empresa reduziu em 25% a infraestrutura de servidores necessários para a operação de desenvolvimento da empresa, diminuindo o custo com energia e pessoal para a manutenção dos equipamentos. Esta redução foi possível devido a eliminação do ambiente necessário para a utilização do RedMine, que era responsável por parte da gerência de projetos das equipes.

O quadro do Scrum passou a ser virtual, utilizando o próprio TFS como mostrado na figura 9.

Figura 9 - Quadro de tarefas



Fonte: Próprio autor (2013)

6 CONCLUSÃO

O objetivo deste trabalho foi realizar um estudo de caso da implantação do Scrum juntamente com o Team Foundation Server. Para isso foi necessário customizar a ferramenta adicionando novos campos aos Itens de Trabalho, customização do fluxo de estados dos itens de trabalho e a criação de um plugin utilizando a API do TFS para automatizar alguns fluxos do processo da empresa.

Um ponto que até então não pode ser completamente atendido, foi a criação do relatório para fechamento dos projetos de forma automatizada. Um relatório que exiba de forma simples todos os dados, gráficos e métricas do projeto recém fechado. Estes dados são necessários para atender os requisitos do MPS.Br nível G o qual a empresa é certificada e por enquanto está sendo gerada de forma manual através do Excel.

Atualmente a fábrica de software onde foi realizado o estudo de caso, continua utilizando o processo descrito neste trabalho e está planejando estender o uso da ferramenta à outros setores da empresa, para facilitar a comunicação entre eles.

REFERÊNCIAS

BLANKENSHIP, E.; WOODWARD, M., HOLLIDAY, G. **Professional Team Foundation Server 2012**. Indianapolis, Wrox, 2013.

CONDÉ, Luciano. **Introdução ao Application Lifecycle Management**. 2009. Disponível em: <<http://msdn.microsoft.com/pt-br/library/ee156630.aspx>>. Acesso em: 08 ago. 2013.

CRUZ, Leandro Rodrigo Saad. **Desenvolvimento de Software com Scrum**. 2008. Disponível em: <<http://www.cimtech.com.br/blog/wp-content/uploads/2010/06/Desenvolvimento-de-Software-com-Scrum.pdf>>. Acessado em: 08 ago. 2013.

DEVIN. **Modelo de Desenvolvimento Ágil Scrum**. Disponível em: <<http://www.devin.com.br/modelo-scrum/>>. Acessado em: 09 ago. 2013.

GOUSSET, M. et al. **Professional Application Lifecycle Management with Visual Studio 2010**. Wrox, 2010.

HERINGER, Cristiano. **ALM focado em resultados**. 2012. Disponível em: <<http://www.slideshare.net/CristianoHeringer/alm-focado-em-resultados>>. Acessado em 08 ago 2013.

HUNDHAUSEN, Richard. **Professional Scrum Development with Microsoft Visual Studio 2012**. Microsoft Press, 2012

MARÇAL, Ana Sofia Cysneiros. **SCRUMMI: Um processo de gestão ágil baseado no SCRUM e aderente ao CMMI**. Dissertação (Mestrado em Informática Aplicada) – Fundação Edson Queiroz. Universidade de Fortaleza. Fortaleza, 2009.

MARTINS, José Carlos Cordeiro. **Técnicas para gerenciamento de projetos de software**. 1.ed. Rio de Janeiro: Brasport, 2007.

MICROSOFT. **Team Foundation Server**. Disponível em: <<http://msdn.microsoft.com/pt-br/vstudio/ff637362.aspx>>. Acesso em: 08 ago. 2013.

ROSSBERG, Joachim; OLAUSSON, Mathias. **Pro Application Lifecycle Management with Visual Studio 2012**. 2. ed. New York, Apress, 2012.

SCHWABER, Ken. **Agile Project Management with Scrum**, Redmond, Microsoft, 2004.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum**, Scrum.org, 2011.

YOSHIMA, Rodrigo. **Gerenciamento de Projetos com Scrum**. 2007. Disponível em: <<http://www.aspercom.com.br/ead/mod/resource/view.php?id=245>>. Acesso em: 09 ago. 2013.