UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Neubio Matos Ferreira

UTILIZAÇÃO DO SCRUM PARA ADEQUAÇÃO PARCIAL DA FÁBRICA DE SOFTWARE DA UNIFAL-MG AO CMMI NÍVEL 2

Alfenas, 03 de Dezembro de 2010.

UNIVERSIDADE FEDERAL DLE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UTILIZAÇÃO DO SCRUM PARA ADEQUAÇÃO PARCIAL DA FÁBRICA DE SOFTWARE DA UNIFAL-MG AO CMMI NÍVEL 2

Neubio Matos Ferreira

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Alfenas como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Rodrigo Martins Pagliares

Alfenas, 03 de Dezembro de 2010.

Neubio Matos Ferreira

UTILIZAÇÃO DO SCRUM PARA ADEQUAÇÃO PARCIAL DA FÁBRICA DE SOFTWARE DA UNIFAL-MG AO CMMI NÍVEL 2

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Prof. Eduardo Gomes Salgado

Universidade Federal de Alfenas

Leonardo Aparecido Ciscon

Devex Tecnologia e Sistemas

Prof. Rodrigo Martins Pagliares (Orientador)

Universidade Federal de Alfenas

Alfenas, 03 de Dezembro de 2010.

Dedico este Trabalho as grandes pessoas da minha vida que tornaram este trabalho possível, meu pai Waldir, minha mãe Luzdalva, meus irmãos Danubio e Adaliene.

AGRADECIMENTO

A minha família pela dedicação, criação, carinho, companheirismo e ensinamentos que recebi durante toda a minha vida que fizeram de mim o homem que sou hoje.

Agradeço em especial ao guerreiro de minha infância, de minha adolescência de minha vida, agradeço ao meu eterno pai Waldir de Souza Ferreira, meu GUERREIRO meu HERÓI, saudades eternas pai. Agradeço a Heroína de minha vida minha mãe Luzdalva de Matos Ferreira, que sempre esteve ao meu lado me apoiando. Agradeço também aos meus irmãos Danubio Matos Ferreira e Adaliene Versiane Ferreira.

Ao Professor, Orientador, Co-Orientador e amigo Humberto Cesar Brandão de Oliveira que me deu a oportunidade e a honra de trabalhar ao seu lado seja em Iniciação Cientifica, seja no Trabalho de Conclusão de curso, seja como Gerente de Projetos Junior da Fábrica de Software de seu laboratório. Muito obrigado Humberto.

Ao Professor Rodrigo Martins Pagliares, pela Orientação deste trabalho de Conclusão de Curso.

A todos os professores que me auxiliaram durante toda a minha graduação.

A todos os meus amigos que durante esses quatro anos e meio de faculdade, fizeram parte de minha vida, seja nos estudos, seja nas conversas, seja nos churrascos e seja nas festas. Saibam que os nomes são muitos, mas seguem alguns deles Robson, Edgar, Julio, Theodoro, Alexandre, Danilo, Jan, Guilherme Zampieri, Rômulo, Flora, Alexsander, Flavio, entre outros.

“Grandes resultados requerem grandes ambições”. (Heráclito)

“O meu espírito Guerreiro determina todos os meus passos.”

Rosângela Cunha

RESUMO

O desenvolvimento de Softwares, em muitos casos, é uma atividade difícil de ser planejada, realizada, e até mesmo imaginada. Devido a esses fatores empresas do mundo todo possuem altas taxa de projetos mal sucedidos, em que softwares são entregues fora do tempo estimado, fora do orçamento e com funcionalidades não terminadas. Na tentativa de diminuir esses índices, métodos, tais como o SCRUM, modelos e processos como o CMMI são criados e padronizados. Muitas organizações que aderem a estes métodos ou modelos criados aumentam significativamente as chances de obterem êxito em seus projetos. Neste intuito, o trabalho presente tem como principal contribuição verificar a possibilidade de se adotar o SCRUM com o objetivo de se atingir o nível 2 de maturidade do CMMI, abordando suas respectivas áreas de processo. Para tal, abordamos neste trabalho o estudo de caso de dois projetos da Fábrica de Software do Laboratório de Pesquisa e Desenvolvimento (LP&D) da UNIFAL-MG. Neste estudo de caso, todos os passos de Planejamento, especificação, implementação e entrega do produto foram realizados com os princípios e regras empregados pelo SCRUM. Espera-se que os resultados deste trabalho possam auxiliar gestores de projetos em tomadas de decisões relacionadas ao desenvolvimento de software e também como guia para empresas que queiram adequar seus processos de desenvolvimento aos processos abordados pelo CMMI nível 2.

Palavras-Chave: SCRUM, CMMI, Estudo de Caso, SCRUM Master

ABSTRACT

The software development, in many cases, is a difficult activity to be planned, performed, and even imagined. Due these factors, companies worldwide have high rate of unsuccessful projects, where software is delivered after the estimated time, off-budget and unfinished features. To attempt to minimize such rates, methods such as SCRUM, models and processes such as CMMI are created and standardized. Many organizations that support these methods or models significantly increases the chances of succeed in their designs. For this reason the present work has as its main contribution to verify the possibility of adopting Scrum with the goal of achieving Level 2 CMMI maturity, addressing their respective areas of process. We examined a case study of two projects from Software Factory at Laboratory Research and Development (LP & D) UNIFAL-MG. All steps of planning, specification, implementation and delivery of the product were carried out with the principles and rules employed by SCRUM. We intend that this study may assist project managers in making decisions related to software development and also as a guide for companies to adapt their development processes to the processes addressed by the CMMI level 2.

Keywords: SCRUM, CMMI, Case of Study, SCRUM Master.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

SEI	*Software Engineering Institute*
DoD	*Departament of Defense*
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integration*
XP	*Extreme Programming*
LP&D	Laboratório de Pesquisa e Desenvolvimento
ROI	*Return of Investiment*

SUMÁRIO

1
Introdução

Neste capítulo será apresentada uma visão geral sobre o tema que será tratado neste trabalho. Na Seção 1.1 são apresentadas a justificativa e a motivação do trabalho. Na Seção 1.2 é discutido o problema que envolve o tema proposto. Na Seção 1.3 são mostrados quais são os objetivos que este trabalho se propõe a realizar e na Seção 1.4 é mostrada a organização desta monografia.

A concepção, construção, análise, desenvolvimento e manutenção de softwares pode ser considerada uma atividade difícil de ser planejada, gerenciada, e até mesmo realizada. No intuito de se garantir o sucesso dessas atividades, muitos processos e métodos são criados e padronizados.

Por volta da década de 90, o Departamento de Defesa Americano (DoD), devido aos elevados gastos com fornecedores de software, viu a necessidade de se criar um modelo de avaliação onde era possível identificar se os fornecedores contratados eram capazes de desenvolver softwares em prazos e orçamentos combinados já que estes, na maior parte das vezes, entregavam os softwares muito além do prazo estipulado, com menos funcionalidades que o acordado, com orçamentos estourados e de baixa qualidade (CARLETON, 1992).

Neste intuito o Departamento de Defesa Americano, iniciou uma parceria de invesTimento com o Instituto de Engenharia de Software (Software Engineering Institute, 1995) que, por sua vez, com base em suas experiências de campo, criou o *Capability Maturity Model* (Systems Security Engineering, 1999). O CMM era um modelo de avaliação de risco para contratação de fornecedores de software que mais tarde se transformou no *Capability Maturity Mode Integration* (CMMI Product Team, 2006), um modelo de maturidade para melhoria de processo, destinado ao desenvolvimento de produtos e serviços. O CMMI hoje é dividido em níveis de crescimento de capacidade e maturidade que permite que organizações se avaliem tendo como referência uma escala pré-definida, podendo fixar objetivos claros para a melhoria do conhecimento e da maturidade da organização.

Hoje, mesmo com a criação destes modelos, empresas de desenvolvimento de software do mundo inteiro, optam por não adotá-los ou os utiliza de forma incorreta, perdendo muito dinheiro e credibilidade por erros em seus planejamentos, acarretando assim no não cumprimento de datas estipuladas, e até mesmo no aumento do custo dos softwares desenvolvidos.

Pesquisas realizadas pelo *Standish Group* nos relataram que em 2009, 32% dos projetos foram entregues conforme o planejado, ou seja no prazo, no orçamento e com funções e funcionalidades definidas em contrato. Este grupo nos relatou também que 44% dos projetos mostraram estar atrasados, seja sobre o orçamento e/ou com menos funcionalidades que o planejado e 24% dos projetos foram encerrados antes de sua conclusão, ou seja, nunca foram entregues (Standish Group, 2009). Estes dados podem ser vistos na tabela 1, que nos mostra também as pesquisas realizadas em anos anteriores e suas estatísticas de sucessos, fracassos, e projetos não entregues conforme planejados.

Tabela 1 – Dados das taxas de projetos de software concluídos no prazo, atrasados e cancelados nos seus respectivos anos.

	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16%	27%	26%	28%	34%	29%	35%	32%
Challenged	53%	33%	46%	49%	51%	53%	46%	44%
Failed	31%	40%	28%	23%	15%	18%	19%	24%

Fonte: Domingues (2009, p.1).

No intuito de aderir ao modelo proposto pelo CMMI, mais especificamente ao nível 2, este projeto têm como objetivo verificar a viabilidade da utilização do SCRUM na tentativa de adequar os processos de desenvolvimento da fábrica de software do LP&D aos processos abordados pelo CMMI.

1.1 Justificativa e Motivação

Atualmente as pesquisas realizadas, principalmente pelo *Standish Group* (STANDISH GROUP, 2009), têm mostrado que a análise, planejamento, ou desenvolvimento, dentre outros fatores que influenciam o desenvolvimento de projetos de software, possuem pontos de falhas. Com o objetivo de amenizar essas taxas de falhas, novos métodos, modelos de processos e experiências adquiridas no desenvolvimento de projetos podem servir de guia para que desenvolvedores ou Gerentes de Projetos, entre outros, que passam por situações semelhantes ao que este trabalho passou possam utilizar as medidas tomadas neste a fim de obterem sucesso no desenvolvimento de seus projetos.

1.2 Problematização

Atualmente é notório que as organizações estão se tornando cada vez mais dependentes das indústrias de softwares e que problemas relacionados ao processo de desenvolvimento de software ficam cada vez mais evidenciados. No intuito de suavizar este problema, uma vertente da engenharia de software vem adotando mini-processos conhecidos como métodos ágeis (FOWLER, 2001), utilizados para o gerenciamento e desenvolvimento de projetos como SCRUM (SCHWABER, 2004), Programação Extrema (XP) (BECK, 1999), *Feature Driven Development* (FDD) (NEBULON, 2010), *Crystal* (COCKBURN, 2004), *Pragmatic Programming* (HUNT, 2000), etc.

Com base em métodos ágeis existentes, esta pesquisa se propõe a resolução da seguinte questão:

É possível que uma fábrica de Software atinja o nível de maturidade do CMMI nível 2 utilizando o método ágil SCRUM?

1.3 Objetivos

1.3.1 Gerais

Implantar o SCRUM para a gestão dos projetos na fábrica de software do Laboratório de Pesquisa e Desenvolvimento (LP&D) e verificar a viabilidade de se utilizar este método ágil para obtenção do nível 2 do CMMI.

1.3.2 Específicos

- Estudar o SCRUM.
- Implantar o SCRUM na fábrica de software do LP&D para realização dos experimentos.
- Apresentar os resultados da utilização do SCRUM em projetos reais da fábrica de software.
- Estudar as áreas de processo que o CMMI aborda.
- Adequar as áreas de processo existentes na fábrica de software com as áreas de processo abordadas pelo CMMI nível 2.
- Avaliar os resultados, se possível chegar a uma conclusão, que sinalize se é possível ou não atender as exigências do CMMI nível 2 usando o SCRUM.

1.4 Organização da Monografia

No Capítulo 2 é mostrado o surgimento do Manifesto Ágil, juntamente com os doze princípios indispensáveis para o sucesso de projetos. Nele também é abordado o Desenvolvimento Ágil, que é um conjunto de técnicas que são aderidas aos projetos com o intuito de diminuir o período de desenvolvimento. O Capítulo 3 é dedicado a realizar uma descrição do método ágil SCRUM, abordando seus

papeis (SCRUM *Master, Product Owner* e Time), seus fundamentos (*Time-Box*, Impedimentos e Definição de pronto). Nele também é abordado assuntos referentes ao planejamento (*Product Backlog, Sprint Backlog e Planning Poker*) e ao andamento dos projetos (Daily SCRUM, *Sprint Review* e *Burndown Chart*). No Capítulo 4 são mostrados conceitos referentes ao CMMI como as representações (Continua e Estagiada) juntamente com seus respectivos níveis (Capacidade e Maturidade) e as áreas de processo existentes. O Capítulo 5 mostra a pesquisa-ação realizada na Fábrica de Software da UNIFAL-MG, juntamente com tudo o que foi feito durante o planejamento, a realização e a entrega dos projetos, abordando todos os passos do SCRUM. Neste capítulo também é abordado as áreas de processo do CMMI, especificamente o nível 2 e como os processos da fábrica foram adequados para atingir o CMMI 2. No Capítulo 6 são apresentadas as conclusões do trabalho.

2
Manifesto e Desenvolvimento ágil de softwares

Nesse capítulo, inicialmente são abordados alguns conceitos como o Manifesto Ágil e o Desenvolvimento Ágil. Na Seção 2.2 são feitas algumas considerações sobre o surgimento do Manifesto Ágil e seus valores, indispensáveis para o sucesso de projetos. Na Seção 2.3 são feitas algumas considerações sobre o desenvolvimento ágil de projetos, que têm como finalidade o desenvolvimento em um curto período de tempo.

2.1 Considerações Iniciais

Em fevereiro de 2001 nos Estados Unidos, um grupo de 17 pessoas se reuniu para discutir, com base em suas experiências, as melhores práticas que utilizavam para o desenvolvimento de software (BECK, 2001).

O objetivo desta reunião era oferecer ao mundo uma alternativa de desenvolvimento de software diferente daquelas altamente dirigidas por documentação. O resultado deste encontro foi a publicação de um conjunto de princípios, mundialmente conhecido como Manifesto Ágil (HIGHSMITH, 2001).

Quando se discute a respeito do Manifesto Ágil, o termo, desenvolvimento ágil, logo vem à tona, e com ele muitas dúvidas e receios sobre a sua adoção ou não são levantadas.

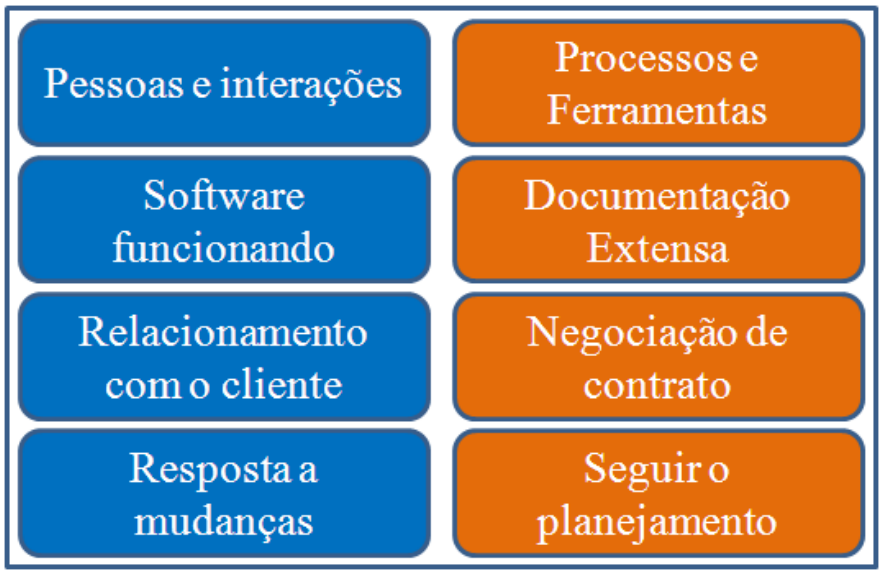
2.2 Manifesto Ágil

O Manifesto Ágil nasceu para que todas as pessoas envolvidas com o desenvolvimento de software tivessem uma alternativa, que substituísse os métodos altamente dirigidos por documentação extensa.

O manifesto ágil é uma declaração redigida por um grupo de 17 pessoas, onde constam os princípios que fundamentam o desenvolvimento ágil. Apesar de cada integrante deste encontro possuir suas próprias teorias e experiências sobre como fazer com que projetos de software tenham sucesso, eles chegaram a um consenso sobre um conjunto de princípios que sempre se repetia quando os projetos davam certo (TELES, 2010), com base neste conjunto o manifesto foi criado.

Os criadores deste manifesto são: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn,Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas (BECK, 2001).

Através da experiência deste grupo o manifesto nos mostra doze princípios indispensáveis para o sucesso de um projeto. Estes princípios podem ser resumidos em valores importantes, de acordo com a figura 1, onde pessoas e interações são mais importantes que processos e ferramentas, software funcionando é mais importante que uma documentação extensa, o relacionamento com o cliente é mais importante que a negociação do contrato e responder as mudanças é mais importante que seguir o planejamento (BECK, 2001).



Fonte: Neubio Ferreira (2010).

Figura 1 – Valores do manifesto ágil

Antes da criação deste manifesto, cada integrante relatou suas experiências e técnicas utilizadas para o sucesso de seus projetos. Foi nesta ocasião que Ken Schwaber apresentou um conjunto de práticas efetivas para o gerenciamento de projetos de softwares denominado SCRUM.

2.3 Desenvolvimento Ágil

O termo desenvolvimento ágil trata de um conjunto de técnicas de desenvolvimento que são aderidas em projetos com o principal foco no desenvolvimento do produto em um curto período de tempo.

Quando incorporamos essas técnicas em projetos, alguns aspectos devem ser levados em consideração como o planejamento, a qualidade desejada, e até mesmo o impacto que essas podem trazer à equipe. Na literatura existem muitos casos de projetos que obtiveram sucesso após aderirem ao desenvolvimento ágil (DINAKAR, 2009).

De maneira sucinta, podemos definir o desenvolvimento ágil de software como uma abordagem iterativa e incremental (evolucionária) que é executada de maneira altamente colaborativa por equipes auto-organizáveis com o objetivo de produzir soluções de alta qualidade dentro dos prazos e orçamentos acordados.

3
SCRUM

Nesse capítulo é mostrado os conceitos do método ágil SCRUM. Na Seção 3.2 abordamos os papéis existentes no SCRUM. Na Seção 3.3 é abordado os fundamentos do SCRUM (Time-Box, Impedimentos e Definição de pronto). Na Seção 3.4 é abordado alguns conceito do Sprint. Na Seção 3.5 é abordado como é feito no planejamento do Sprint e o que ocorre durante este (Daily SCRUM e Sprint Backlog). Na Seção 3.6 é mostrado o que acontece no último dia do Sprint (Sprint Review). Na Seção 3.7 é abordado como o SCRUM Master pode levantar pontos que podem ser melhorados para os próximos Sprints (Sprint Retrospective). Na Seção 3.8 é abordado uma técnica muito utilizada em que a equipe estima o grau de dificuldade de uma certa funcionalidade do sistema (Planning Poker) e na Seção 3.9 é abordado como o SCRUM Master verifica o andamento do projeto através de graficos (Burndown Chart.)

3.1 Considerações Iniciais

SCRUM é um método ágil altamente focado em objetivos, utilizado em diversas áreas, como na construção civil, na indústria automobilística, nas telecomunicações, entre outros. Em nosso caso estaremos utilizando o SCRUM para gerenciar e controlar o desenvolvimento de software utilizando práticas iterativas e incrementais.

É importante frisar que os processos do SCRUM caracterizam-se como processos empíricos e adaptativos (BASSI, 2008) e não prescritivos.

Processos empíricos são processos caóticos, com muitas incertezas onde não há a possibilidade de prever exatamente tudo o que irá acontecer (SCHWABER, 2004), já os processos prescritivos são modelos definidos com mecanismos claramente entendidos, onde temos a capacidade de estimar os tempos de execução de cada atividade.

3.2 Papéis do SCRUM

O SCRUM, como qualquer outro método ágil, é baseado em papéis e responsabilidades que possuem um propósito comum: o sucesso do projeto.

3.2.1 Product Owner

Product Owner é o nome dado ao(s) financiadores do projeto ou um importante interessado no mesmo (SCHWABER, 2004).

É ele quem define as funcionalidades do produto, concentra informações vindas de usuários, *stakeholders* ou do mercado de maneira que se obtenha uma visão única dos requisitos do sistema. Também é responsável por priorizar o *Product Backlog*, uma lista contendo as funcionalidades do sistema. Cabe ao *Product Owner* aceitar ou rejeitar os resultados dos trabalhos e alterar as prioridades dos itens do *Product Backlog* fora da iteração vigente, chamada de *Sprint*. Este papel representa o principal investidor do Projeto (HUNDERMARK, 2009) (STUART, 2010).

3.2.2 O Time

O Time é um grupo de pessoas diretamente ligadas ao trabalho a ser feito. Este papel se esforça ao máximo para que o projeto seja entregue com todas as funcionalidades necessárias.

Normalmente é formado por até sete pessoas, responsáveis por definir o objetivo do *Sprint*, além de especificar os resultados dos trabalhos e demonstrar o resultado para o *Product Owner* e outros *Stakeholders*. O Time é Multi-funcional, auto-organizável e faz aquilo que é necessário dentro das diretrizes do projeto para alcançar o objetivo do *Sprint* (HUNDERMARK, 2009) (Stuart, 2010) (SCHWABER, 2004).

3.2.3 SCRUM Master

Desempenha um papel de liderança, gerenciando os interesses do *Product Owner* mediante o Time. Numa abordagem tradicional o SCRUM *Master* seria um Gerente de Projetos, porém essa nomenclatura foi substituída para diferenciar o foco de liderança necessário para que um processo empírico funcione.

É de responsabilidade deste melhorar o ambiente de trabalho e a produtividade do Time de desenvolvimento promovendo a criatividade, o conhecimento, a comunicação e cooperação entre todas as pessoas envolvidas.

Cabe ao SCRUM *Master* proteger o Time de interferências externas, removendo impedimentos, garantindo que o processo esteja sendo respeitado, através de reuniões de acompanhamento (*Daily SCRUM, Sprint Review e Sprint Retrospective*). Este papel também pode auxiliar o *Product Owner* a maximizar o retorno sobre invesTimento, atingindo seus objetivos e promovendo práticas de engenharia, para que cada funcionalidade seja potencialmente implantável (HUNDERMARK, 2009) (STUART,2010) (SCHWABER, 2004).

3.3 Fundamentos do SCRUM

3.3.1 Time-box: Período de tempo

O *Time-Box* é um conceito importante para as práticas do SCRUM. Trata-se de um período de tempo em que são realizadas diversas atividades prescritas no processo. Um *Sprint* (iteração) de 30 dias é um *Time-Box*. O planejamento que ocorre no primeiro dia do *Sprint* ocorre num Time-*Box* de um dia. As reuniões diárias com a equipe devem demorar no máximo 15 minutos (DEEMER, 2010). Tudo que acontece dentro da metodologia SCRUM tem o espaço de tempo definido e cronometrado.

3.3.2 *Impedimentos*

Um impedimento é qualquer problema que um membro do Time está enfrentando e que impede o andamento dos trabalhos. Os impedimentos são reportados diretamente ao SCRUM *Master*, responsável por remover essas barreiras.

3.3.3 *Definição de Pronto*

Um objetivo do *Sprint* só é dado como pronto quando este é potencialmente implantável, isto é, a funcionalidade só está pronta quando possui o potencial de entrar em produção assim que o *Product Owner* decidir (SCHWABER, 2004).

Esta definição de pronto é um importante aspecto para avaliação do projeto, pois o projeto só avança quando itens são dados como prontos no *Product Backlog*.

3.4 *Sprint*

Um *Sprint* é um período de trabalho planejado e observável. Durante esse período a equipe faz tudo ao seu alcance para que os objetivos que agregam valor ao negocio sejam alcançados.

O processo iterativo é um dos conceitos básicos para qualquer metodologia de desenvolvimento de software que possui como objetivo minimizar riscos oferecendo uma rápida avaliação dos usuários a respeito do que está sendo construído.

Uma iteração é um período de tempo fixo em que o Time está trabalhando para que ao final deste período algo de valor para os usuários ou interessados no projeto seja demonstrado. Esta demonstração é importante para avaliar o andamento do projeto e também para verificar se o Time esta realmente compreendendo o que o produto deve fazer.

No SCRUM uma iteração é chamada de *Sprint*, normalmente com duração de 30 dias. Dentro deste período, o Time trabalha nos objetivos determinados para o *Sprint* (SCHWABER, 2004).

Um projeto SCRUM é composto por vários *Sprints* do mesmo tamanho. Não é aconselhável que o tamanho dos *Sprints* varie, pois um período de tempo fixo é a melhor maneira para observar resultados, principalmente para avaliar a produtividade da equipe (KTATA, 2010).

3.5 Planejamento do *Sprint*

No SCRUM, o planejamento do projeto como um todo ocorre a cada planejamento do *Sprint*. Este planejamento ocorre em duas partes, cada uma delas com *Time-Box* de 4 horas.

Na primeira parte da reunião o SCRUM *Master* reúne-se com o *Product Owner* para a criação/manutenção do *Product Backlog*.

O *Product Backlog* é um artefato do SCRUM que se assemelha com um documento ou planilha contendo todas as funcionalidades de alto nível do sistema, ordenadas por prioridades, solicitadas pelos *stakeholders* (DEEMER, 2010).

Nesta primeira parte da reunião a conversa é focada no *ROI* (Retorno sobre InvesTimento) do projeto. O trabalho de ordenar o *Product Backlog* é uma importante tarefa, pois o *Product Owner* decidirá aquilo que agrega mais valor ao software para ser entregue ao final do *Sprint* que se inicia no dia seguinte.

Os itens que aparecem no *Product Backlog* são qualquer tipo de funcionalidade ou tarefa que possua um valor para o *Product Owner* ou ao grupo de *Stakeholders* que este *Product Owner* representa.

Estes itens seguem um conceito do SCRUM chamado "incremento de funcionalidade potencialmente implantável". Este termo nos diz que quando um item do *Product Backlog* for finalizado ele estará completamente funcional e resolvendo algum problema de negócio dos *stakeholders*.

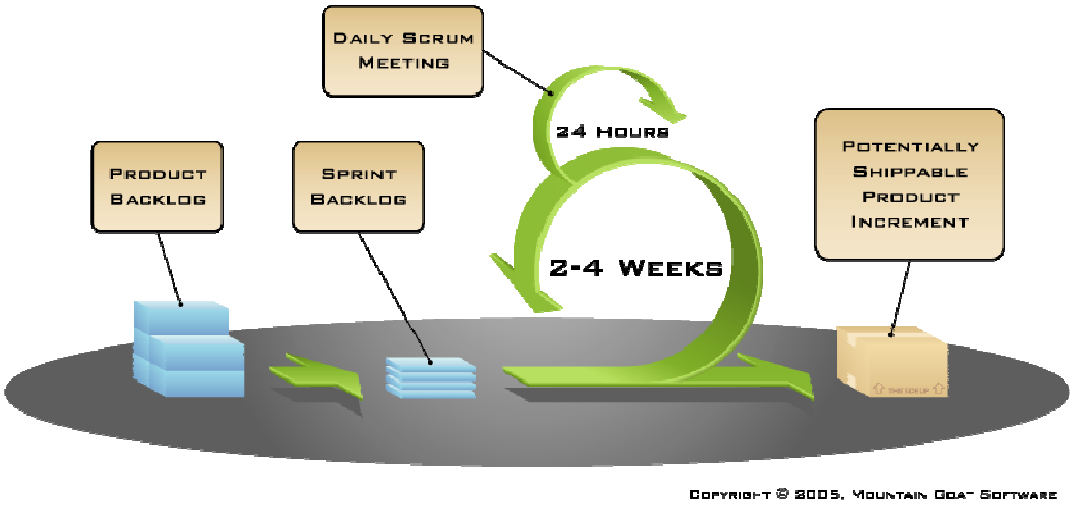
O Planejamento dos *Sprints* é feito através de conjuntos de funcionalidades potencialmente implantáveis, sendo que ao final do *Sprint* algo funcional sempre esteja sendo entregue (SCHWABER, 2004).

Na segunda parte do dia do planejamento, o SCRUM *Master* se reúne com o Time e com o *Product Owner* para planejar o *Sprint* baseado no *Product Backlog* criado ou atualizado. O trabalho nessa segunda parte é estabelecer e firmar os objetivos do *Sprint*, selecionando quais itens prioritários do *Product Backlog* será implantado completamente até o fim dos próximos 28 dias, conforme pode ser visto na figura 2.

Após selecionar itens potencialmente implantáveis que serão resolvidos no *Sprint*, o Time quebra cada item selecionado em tarefas menores que podem ser cumpridas, desejavelmente, em um dia. Esta quebra de cada item é chamado *Sprint Backlog*, e o Time define que cada uma dessas tarefas esclarece tudo que é necessário ser feito para implementar os itens do *Product Backlog* selecionados para esse *Sprint*.

Após definido o *Sprint*, o SCRUM *Master* realiza reuniões diárias com o Time, como pode ser visto na figura 2.

Após o término do *Sprint*, o SCRUM *Master* juntamente com o Time realizam o *Sprint Review*, e mais tarde realizam junto ao *Product Owner* o *Sprint Retrospective* demonstrando a ele uma funcionalidade potencialmente implantável. Ambos, *Sprint Review* e *Sprint Retrospective* serão discutidos adiante neste trabalho.



Fonte: THREEWILL (2005).

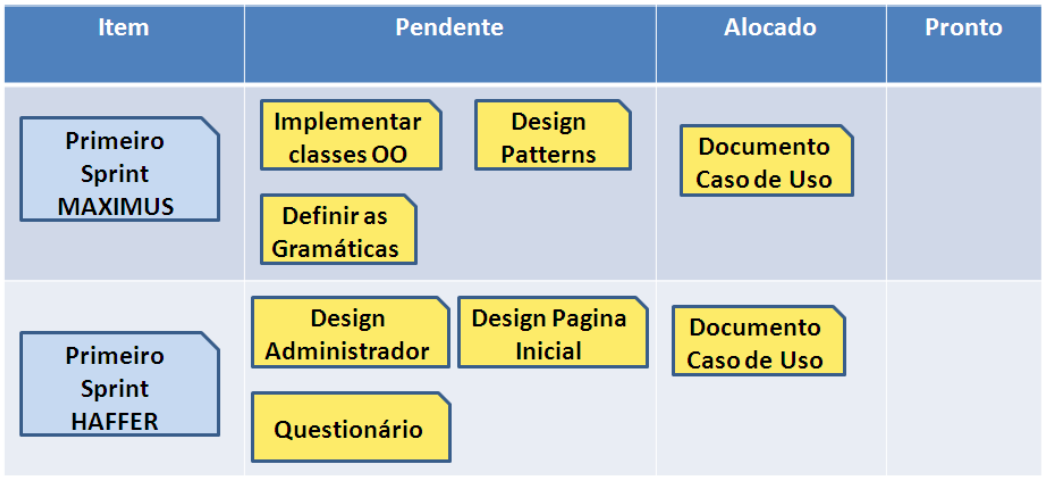
Figura 2 – Ciclo do SCRUM

3.5.1 Sprint Backlog

Em todas as empresas onde o SCRUM é adotado nos deparamos com um quadro similar ao da figura 3. Este quadro é preenchido após a definição do *Sprint Backlog*, onde o item do *Product Backlog* é colocado na coluna item e a quebra desses itens em tarefas menores é colocada na coluna Pendente. Normalmente tanto o item do *Product Backlog* quanto suas quebras em tarefas são escritos em post-its.

Uma característica importante é que cabe ao membro do Time a seleção da tarefa que irá executar; assim, este desloca o post-it da tarefa escolhida da coluna Pendente para a coluna Alocado, e assim por diante. Isso faz com que o Time tenha um maior compromeTimento e um maior controle sobre o próprio trabalho.

Lembrando que tal item só pode ser colocado na coluna Pronto se obedecer ao critério de potencialmente implantável.



Fonte: Neubio Ferreira (2010).

Figura 3 – Quadro SCRUM

3.5.2 Daily SCRUM – Reunião Diária

Um evento importante que ocorre durante o *Sprint* é a reunião Diária. Esta reunião é feita com o SCRUM *Master*, o Time e qualquer pessoa interessada no projeto. As reuniões diárias ocorrem num *Time-Box* de no máximo 15 minutos (SCHWABER, 2004) (RUBART, 2009). Esse tempo limitado serve para que o Time se concentre no que realmente é importante, e que a reunião não dure 2 ou 3 horas comprometendo a produtividade do Time.

Nessas reuniões cada membro do Time dá suas impressões a respeito do projeto respondendo a três perguntas importantes (SCHWABER, 2004):

1. O que eu fiz desde a última reunião diária?
2. O que eu pretendo fazer até amanhã?
3. Têm alguma coisa impedindo o meu trabalho?

Essas reuniões são importantes, pois o *SCRUM Master* verifica o andamento do trabalho, observando problemas, verificando a continuidade do processo, resolvendo mal-entendidos e principalmente liderando pessoas.

3.6 *Sprint Review*

É um importante ponto de inspeção da metodologia SCRUM. Esta reunião ocorre no último dia do *Sprint* (KTATA, 2010) e representa o momento e, que o Time juntamente com o SCRUM *Master* demonstram as funcionalidades potencialmente implantáveis desenvolvidas durante o *Sprint* para o *Product Owner* (SCHWABER, 2004).

3.7 *Sprint Retrospective*

É uma reunião entre o SCRUM *Master* e o Time onde duas perguntas são feitas:

1. O que foi bom durante o *Sprint*?
2. O que pode ser melhorado?

O objetivo desta reunião é a transparência interna do Time. O SCRUM *Master* deve avaliar os pontos apresentados e prover recursos necessários para que as mudanças ocorram.

A adaptação contínua é um fundamento importante para controlar projetos críticos e esses pontos de melhorias devem ser valorizados.

3.8 *Story Points*

Story Points são unidades de medida utilizada por equipes ágeis para estimarem a complexidade funcional do sistema. Não existe uma fórmula definida para a definição do tamanho do sistema. Em vez disso utilizamos o Story Points que é uma mistura de quantidade de esforço envolvido no desenvolvimento, na complexidade, no risco inerente a ela, entre outros.

Para estimar itens do sistema utilizando Story Points primeiro é analisado os itens referentes ao sistema que o Time julgue mais fácil de ser realizado. Este

item recebe um Story Point e todos os outros itens do sistema são avaliados em relação a este *Story Point*.

Normalmente a velocidade da equipe é dada por estes Story Points, pois caso uma equipe desenvolva cinco Story Points dentro de cinco dias, é visível que esta equipe realizará dez Story Points em dez dias.

Neste projeto utilizamos a técnica do *Planning Poker* para estimar o tamanho funcional e a complexidade técnica dos itens do sistem.

É importante ressaltar que uma estimativa nunca acertará o nível de precisão de cem por cento, pois uma estimativa não é uma previsão certa do futuro.

3.8.1 Planning Poker

É uma técnica utilizada pelo Time para avalar o tamanho funcional e a complexidade técnica de um determinado item do *Product Backlog*. Esta técnica é uma das maneiras de se obter estimativas mais próximas da opinião do grupo como um todo.

Cada membro do Time recebe um conjunto de cartas contendo normalmente os números 1,2,3,5,8 da seqüência de Fibonacci. Cada integrante da equipe pontua a complexidade de cada item do *Product Backolog* utilizando para isso uma carta com um número da seqüência (GRENNING, 2002). Todos os integrantes devem mostrar a carta escolhida ao mesmo tempo, conforme figura 4, de forma que a opinião de um membro da equipe não interfira na opinião dos outros.

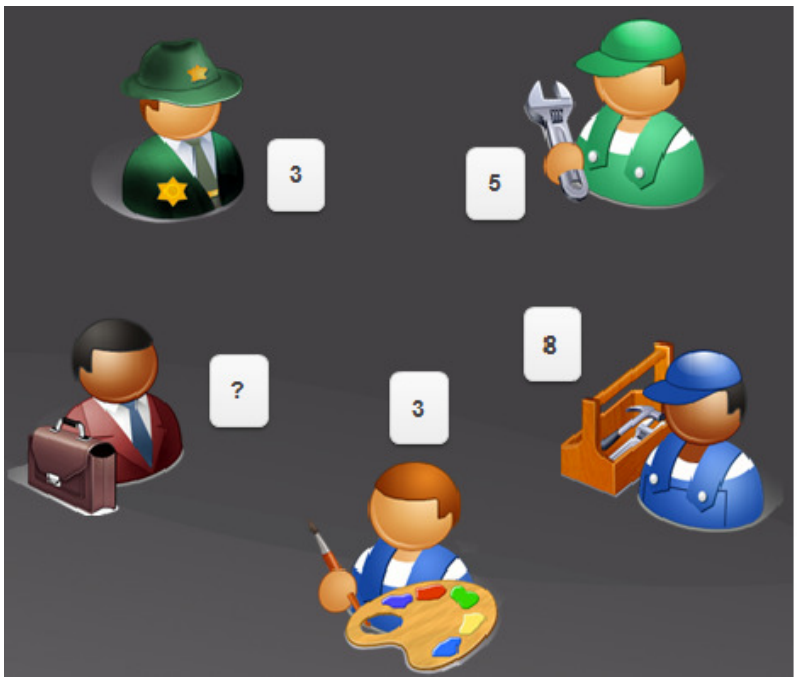


Figura 4 – *Planning Poker*

Caso em algum item do *Product Backlog* a equipe apresente valores de cartas diferentes, é perguntado aos membros porque, na opinião deles, tal item merece referida estimativa. Isto é feito com todos os membros do Time. Depois de discutir a estimativa, o item é avaliado novamente. Isso acontece até que o Time entre em consenso.

Caso existam dúvidas remanescentes acerca de algum item do *Product Backlog*, recomenda-se a participação na reunião de alguns *stakeholders* ou do *Product Owner* para maiores esclarecimentos sobre o item. Modelos preliminares como protótipos de tela em papel, documento de arquitetura ou levantamentos gerados no tempo de concepção, podem auxiliar nesta tarefa de estimar os itens através do *Planning Poker*. Vale a pena ressaltar que o *Product Owner* ou demais *stakeholders* somente participam no esclarecimento dos itens do *Product Backlog*. Não é recomendado que eles participem ativamente das estimativas de forma a não coagir os membros do Time a usarem sempre números baixos nas suas estimativas.

3.9 *Burndown Chart*

O Burndown Chart é um gráfico onde o SCRUM *Master* verifica o andamento do projeto (SCHWABER, 2004) conforme pode ser visto pela figura 5.

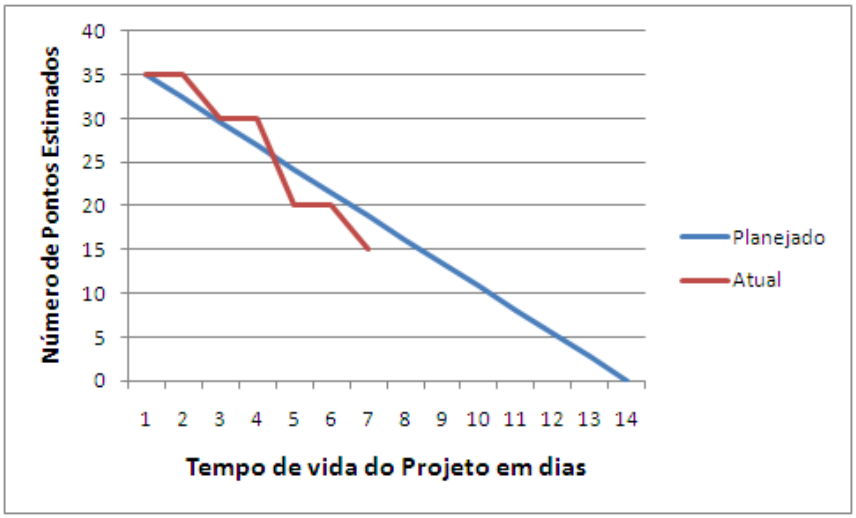


Figura 5 – Burndown Chart

Neste gráfico podemos perceber que o eixo X é a linha de tempo do *Sprint* e o eixo Y é o número de pontos estimados através do *Planning Poker*, que devem ser cumpridos durante o *Sprint*. A linha azul representa o ponto de referência para o SCRUM *Master*, ou seja, ela marca a previsibilidade de conclusão do *Sprint*. A linha vermelha representa o andamento do projeto. Caso a linha vermelha esteja abaixo da linha azul, temos a informação de que a velocidade do Time está acima do planejado; caso a linha vermelha esteja acima da linha azul, isso implica que a velocidade do Time está abaixo do esperado e a tendência é que o prazo para este *Sprint* não seja cumprido.

O *Burndown Chart* é um complemento visual do *Product Backlog* e ele é uma ferramenta de extrema importância para o SCRUM *Master* acompanhar o andamento do projeto, tomando as decisões necessárias para que o projeto seja cumprido no prazo determinado (RUBART, 2009).

4
Capability Maturity Model Integration - CMMI

Neste capítulo é mostrado alguns conceitos do CMM como sua criação e sua finalidade. Na Seção 4.2 é abordado as representações que o CMMI possui (Continua e Estagiada) juntamente com seus respectivos níveis (Capacidade e Maturidade). Na Seção 4.3 é abordado as áreas de processo existentes no CMMI juntamente com as áreas de processo das representações Continua e por Estágios.

4.1 Considerações Iniciais

CMMI é um modelo de referência criado em 1997 por Watts Humphrey, do Instituto de Engenharia de Software (SEI) da Universidade de Carnegie Mellon. Este modelo contém um conjunto de práticas que abrangem o ciclo de vida do produto desde a concepção até a entrega, garantindo a maturidade na capacidade de gerenciar e melhorar a qualidade a serem aplicadas em processos de desenvolvimento de software (CMMI Product Team, 2006).

O CMMI não é uma metodologia, pois ele mostra o que deve ser feito, mas não diz nada a respeito de como deve ser feito.

O CMMI é uma evolução do CMM (*Capability Maturity Model*) que surgiu em 1987 como modelos de avaliação de risco de contratação de fornecedores de software para as forças armadas Norte Americanas. A idéia era que os produtos fossem entregues com qualidade e em prazos estipulados.

A diferença entre o CMMI e o CMM é que a partir dos anos 90 surgiram diversos CMM's (Systems Security Engineering, 1995) com foco em desenvolvimento de sistemas, engenharia de software, aquisição de produtos, entre outros. Apesar destes modelos serem úteis para as organizações, eles se tornaram um problema, pois cada modelo deveria ser avaliado separadamente e isso gerava

mais custos para a organização. No intuito de resolver este problema, unificando todos esses modelos em um só, foi criado o CMMI.

O CMMI é utilizado por organizações com o intuito de aprimorar seus processos de desenvolvimento, e na atual estrutura que este abrange, podemos afirmar que pode ser utilizado também na construção civil, na administração, no marketing, entre outros.

Afim de abordar melhorias e avaliação de processos o CMMI utiliza duas representações distintas.

4.2 Representações CMMI

O CMMI possui duas representações: contínua e por estágios. Estas representações permitem que organizações possam utilizar diversos caminhos para a melhoria de seus processos.

4.2.1 Representação Contínua

Este tipo de representação permite que organizações melhorem diferentes processos, em diferentes momentos, ou seja, uma organização pode focar na melhoria de um processo problemático, ou pode trabalhar em áreas que estejam ligadas a objetivos estratégicos da organização.

Este tipo de representação também permite que organizações melhorem diferentes processos ao longo do tempo, pois algumas áreas de processos são muito dependentes uma das outras (CMMI Product Team, 2006).

A representação contínua utiliza a ordem de melhoria que melhor atende os negócios das organizações, e esta representação é caracterizada por níveis de capacidade.

4.2.1.1 Níveis de Capacidade

Na representação contínua existem seis níveis de capacidade, e cada um deles descreve o comportamento dos processos nas organizações.

No nível 0, denominado Incompleto, os processos não são executados ou são executados parcialmente (SOTILLE, 2006).

No nível 1, denominado Executado (Definido), os processos são executados, ou seja, as metas especificas das áreas de processo são satisfeitas, viabilizando o trabalho necessário para produzir os produtos de trabalho (SOTILLE, 2006).

No nível 2, denominado Gerenciado/Gerido, os processos são executados e possuem infra-estrutura adequada para apoiar o processo planejado e executado de acordo com uma política. Neste nível os processos são monitorados, controlados, revisados e avaliados (SOTILLE, 2006).

No nível 3, denominado Definido, os processos são executados, e são gerenciados mais proativamente, baseando-se na compreensão de como as atividades de processo relacionam-se e nas medidas detalhadas do processo, seus produtos de trabalho e serviços (SOTILLE, 2006).

No nível 4, denominado Quantitativamente Gerenciado/Gerido Quantitativamente, os processos são executados e controlados por meio de técnicas estatísticas e quantitativas (SOTILLE, 2006).

No nível 5, denominado Otimizado, os processos são executados e são melhorados com base no entendimento das causas comuns de variação inerentes ao processo. O foco deste processo é a melhoria contínua do desempenho de processo tanto por meio de melhorias incrementais quanto de inovação (SOTILLE, 2006).

4.2.2 Representação por Estágios

Este tipo de representação permite que as organizações melhorem um processo de cada vez, nomeando cada melhoria de processo de estágio. Atingir um estágio, significa que uma estrutura de processo adequada foi estabelecida, e esta estrutura servirá de base para que uma nova estrutura seja estabelecida (CMMI Product Team, 2006).

Assim neste tipo de representação uma organização deve melhorar uma área de processo que servirá de base para outra área de processo, e assim os processos dentro da organização são melhorados continuamente. Este tipo de representação é caracterizado por Níveis de Maturidade.

4.2.2.1 Níveis de Maturidade

Na representação por estágios existem cinco níveis de maturidade e cada um deles descreve o comportamento dos processos nas organizações.

No nível 1, denominado de Inicial, os processos são caóticos e não fornecem as organizações ambiente estável, muitas vezes o sucesso depende da habilidade de pessoas e não do uso de processos comprovados. Apesar deste caos, organizações neste nível produzem serviços e produtos que funcionam (CHRISSIS, 2004).

No nível 2, denominado Gerenciado/Gerido, os processos são planejados e organizados, de acordo com uma política adotada pela organização. Neste nível, mesmo em períodos de *stress*, as políticas existentes são mantidas (CHRISSIS, 2004).

No nível 3, denominado Definido, os processos são bem caracterizados e entendidos pois são descritos em padrões, procedimentos, ferramentas e métodos (CHRISSIS, 2004).

No nível 4, denominado Quantitativamente Gerenciado/Gerido Quantitativamente, as organizações estabelecem objetivos quantitativos para que os processos tenham qualidade e desempenho elevados. Neste nível os processos são entendidos em termos estatísticos e gerenciados ao longo da vida dos processos (CHRISSIS, 2004).

No nível 5, denominado otimização, os processos sofrem melhorias incrementais e inovadoras. Os processos das organizações neste nível são continuamente revisados, para garantir a melhoria dos processos (CHRISSIS, 2004).

4.3 Áreas de Processo

Dentro de cada nível, tanto de maturidade quanto de capacidade, existem áreas de processos. Cada área de processo contém um conjunto de práticas relacionadas que quando implementadas satisfazem um conjunto de metas consideradas importantes, realizando melhorias significativas naquela área (CMMI Product Team, 2006).

O modelo CMMI possui em seu total vinte e duas áreas de processos, que podem ser visualizadas na Tabela 2.

Tabela 2 – Áreas de processo com sua respectiva Categoria ou Nível de Maturidade

Área de Processo	*Categoria*	*Nível de Maturidade*
Análise e Resolução de Causas	Suporte	5
Gestão de Configuração	Suporte	2
Análise e Tomada de Decisões	Suporte	3
Gestão Integrada de Projeto +IPPD	Gestão de Projeto	3
Medição e Análise	Suporte	2
Implantação de Inovações na Organização	Gestão de Processo	5
Definição dos Processos da Organização +IPPD	Gestão de Processo	3
Foco nos Processos da Organização	Gestão de Processo	3
Desempenho dos Processos da Organização	Gestão de Processo	4
Treinamento na Organização	Gestão de Processo	3
Integração de Produto	Engenharia	3
Monitoramento e Controle de Projeto	Gestão de Projeto	2
Planejamento de Projeto	Gestão de Projeto	2
Garantia da Qualidade de Processo e Produto	Suporte	2
Gestão Quantitativa de Projeto	Gestão de Projeto	4
Desenvolvimento de Requisitos	Engenharia	3
Gestão de Requisitos	Engenharia	2
Gestão de Riscos	Gestão de Projeto	3
Gestão de Contrato com Fornecedores	Gestão de Projeto	2
Solução Técnica	Engenharia	3
Validação	Engenharia	3
Verificação	Engenharia	3

Fonte: CMMI Product Team (2006, p.44).

4.3.1 Áreas de Processo da Representação Contínua

Na representação contínua as áreas de processo são divididas em quatro categorias, sendo elas:

Gestão de Processos: Possui atividades relacionadas aos projetos como a definição, planejamento, implantação, implementação, monitoramento, controle, validação, medição e a melhoria de processos (CMMI Product Team, 2006).

Gestão de Projetos: Possui atividades relacionadas ao planejamento, monitoramento e controle do projeto.

Engenharia: Possui atividades relacionadas ao desenvolvimento e manutenção das diversas disciplinas da engenharia de software.

Suporte: Possui atividades relacionadas ao apoio desenvolvimento e manutenção do produto.

Ao selecionarmos algumas áreas de processos devemos selecionar o nível de capacidade que se deseja para que os processos associados às áreas de processo amadureçam. Por exemplo, uma organização pode decidir esforçar-se para alcançar o nível de capacidade 2 em uma área de processo e nível de capacidade 4 em outra. Na medida que a organização alcança determinado nível de capacidade, ela redireciona seu foco para o próximo nível dessa área de processo ou decide tratar outras áreas de processo (CMMI Product Team, 2006).

Na tabela 2, são mostradas as áreas de processo, e o nível de capacidade correspondente.

4.3.2 Áreas de Processo da Representação por Estágios

Neste tipo de representação existe um caminho pré-definido onde a melhoria dos processos é feita a partir do nível de maturidade 1 e segue em direção ao nível de maturidade 5.

Neste tipo de representação as áreas de processo estão agrupadas por níveis de maturidade, assim, para atingirmos um nível de maturidade todas as áreas de

processo deste nível devem ter sido implementadas. Na tabela 2 são mostradas as áreas de processo e o nível de maturidade correspondente.

Apesar das áreas de processo serem agrupadas, tanto na representação Contínua quanto na representação por Estágio, elas acabam interagindo entre si, e afetando umas as outras independente do grupo no qual pertencem.

5
Estudo de Caso

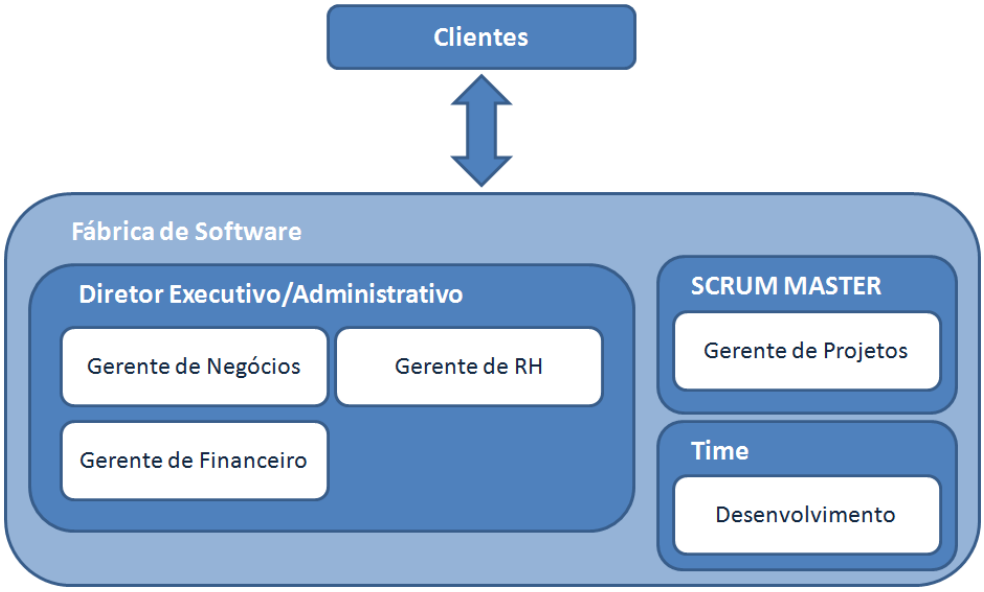
Neste capítulo abordamos o estudo de caso feito na Fábrica de Software da UNIFAL-MG, e alguns conceitos estruturais da fábrica. Na Seção 5.2 é abordado os projetos utilizados neste estudo de caso juntamente com os seus planejamentos e reuniões (Daily SCRUM, Sprint Review e Sprint Retrospective). Na Seção 5.3 é abordado o Product Backlog dos projetos e as estimativas feitas pela equipe para cada requisito deste. Na Seção 5.4 é abordado os Bunrdown Charts dos Sprints dos projetos e também a estimativa de término dos projetos com base no primeiro Sprint. Na Seção 5.5 é abordado conceitos sobre o CMMI nível 2 e qual foi a representação deste utilizada na fábrica de Software e na Seção 5.6 é abordado como atingimos as áreas de processo do CMMI nível 2 utilizando o SCRUM.

5.1 Considerações Iniciais

A pesquisa-ação é um tipo de pesquisa no qual os pesquisadores fazem parte efetivamente do trabalho estudado, contribuindo com idéias realizando ações na tomada de decisão e revelando acontecimentos que determinam e envolvem diversos momentos da realidade estudada (FRANCO, 2005).

A pesquisa-ação que este trabalho aborda, foi realizado na Fábrica de Software do LP&D da Universidade Federal de Alfenas. Atualmente a fábrica é composta por oito pessoas, sendo sete graduandos e um doutorando.

A estrutura da fábrica pode ser vista de acordo com a figura 6.



Fonte: Neubio Ferreira (2010).

Figura 6 – Estrutura da Fábrica de Software

Os cargos da fábrica são divididos da seguinte forma:

Diretor Executivo/Administrativo possui a função de administrar e prover recursos para que a fábrica funcione adequadamente, e realizar a gerência de RH. Cabe ao Diretor e ao SCRUM *Master* levantar junto aos Clientes requisitos necessários para o desenvolvimento dos projetos.

SCRUM *Master* têm como função planejar as atividades dos projetos a serem cumpridas, gerenciar os interesses do *Product Owner* mediante ao Time, melhorar a produtividade da equipe, garantir que os processos estejam sendo realizados, garantir que o produto seja entregue conforme combinado em contrato, entre outros.

O Time possui a função de entregar o projeto com todas as funcionalidades necessárias.

O autor deste projeto participou efetivamente dos projetos atuando no papel de SCRUM Master.

5.2 Projetos

A pesquisa-ação, descrita neste trabalho, foi feito utilizando dois projetos.

O primeiro projeto intitulado MAXIMUS refere-se a um projeto que têm como objetivo implementar um software robô inteligente, que irá submeter ordens de compra e venda de ações, de acordo com regras de mercado encontradas por ele. A equipe designada para este projeto é composta de um Gerente de projetos e dois desenvolvedores.

O segundo projeto intitulado HAFFNER trata-se de um software para gestão de RH, que será capaz, através de métodos de agrupamento otimizados, e com base em estudos sócio-métricos, indicar dentro de um conjunto de pessoas, possíveis grupos que possuem a maior tendência de serem harmônicos para trabalharem em conjunto. A equipe designada para este projeto consta de um gerente de projetos, e um desenvolvedor.

É importante ressaltar que cada projeto possui equipes diferentes, sendo o gerente de projetos o mesmo nos dois projetos.

5.2.1 Planejamento

Antes de iniciarmos o planejamento dos projetos, o SCRUM *Master* juntamente com o Diretor Administrativo/Executivo reuniu com os *Product Owners* (Clientes) dos respectivos projetos para realizar um estudo sobre a viabilidade do projeto baseado na visão apresentada.

Ao validar a viabilidade de desenvolvimento dos projetos, o planejamento dos mesmos foi dividido em três momentos.

No primeiro momento o SCRUM *Master* e o *Product Owner* de cada projeto construíram o *Product Backlog*. Em seguida os respectivos *Product Owners* definiram os requisitos a serem incorporados no *Product Backlog*. Além de definidos, os requisitos foram priorizados com base no valor de negócio que eles representam.

Após esta definição de prioridades eles estabeleceram que o *Sprint* teria um *Time-box* de 14 dias úteis.

No segundo momento o SCRUM *Master* reuniu com o Time para que fosse feito uma estimativa de esforço de cada item do *Product Backlog* . Neste momento o Time utilizou a técnica *Planning Poker*.

No terceiro momento o SCRUM *Master* reuniu todos os envolvidos no projeto (*Product Owner* e Time) para definirem os itens do *Product Backlog* que fariam parte do primeiro *Sprint*. É importante ressaltar que esta reunião ocorreu duas vezes sendo uma reunião para cada projeto.

Através desta reunião foi decidido que para o primeiro *Sprint* o projeto MAXIMUS trabalharia com um esforço de 19 pontos, e o projeto HAFFNER trabalharia com um esforço de 24 pontos. Esta pontuação foi estabelecida a partir de um consenso entre os membros do Time já que para o primeiro Sprint, não havia dados históricos sobre a real velocidade do Time.

Após isso o Time se reuniu e quebraram os itens do Primeiro *Sprint* em tarefas menores, criando assim o *Sprint Backlog*.

O *Sprint Backlog* contém todas as tarefas a serem feitas para que os itens definidos para o primeiro *Sprint* sejam implementados, e no final algum produto potencialmente implantável seja entregue.

5.2.2 Reuniões Diárias

A fim de verificar o andamento dos projetos, o SCRUM *Master* realizou nos dois projetos, reuniões diárias com um *Time-Box* de 15 minutos no máximo.

Estas reuniões eram realizadas com todos os membros do Time, e estes respondiam a três perguntas:

1. O que eu fiz desde a última reunião diária?
2. O que eu pretendo fazer ate amanhã?
3. Têm alguma coisa impedindo o meu trabalho?

Com as informações extraídas destas perguntas e ao traçar o Burndown Chart de cada projeto, o SCRUM *Master* tinha o poder de visualizar o andamento do projeto, e de dizer se o projeto estava atrasado, adiantado, ou se estava

ocorrendo conforme o planejado. Além disso, as respostas da terceira pergunta permitiram que o SCRUM *Master* pudesse sanar os problemas levantados o quanto antes.

As reuniões diárias do projeto MAXIMUS podem ser vistas no Anexo I juntamente com seu respectivo *Sprint Backlog*.

As reuniões diárias do projeto HAFFNER podem ser vistas no Anexo II juntamente com seu respectivo *Sprint Backlog*.

5.2.3 Sprint Review

No final de todo *Sprint*, o SCRUM *Master* juntamente com o Time, em reunião com o *Product Owner* e demais *stakeholders*, apresentavam o produto, potencialmente implantável, produzido durante todo o *Sprint*.

5.2.3.1 Projeto MAXIMUS

Após a reunião do primeiro *Sprint Review* deste projeto, foram solicitadas mudanças estruturais na árvore de decisão, desenvolvida durante o *Sprint*. Estas alterações foram incorporadas no segundo *Sprint* como revisão de código.

Após a reunião do segundo *Sprint Review* o *Product Owner* não solicitou nenhuma mudança no *Product Backlog*.

5.2.3.2 Projeto HAFFNER

Após a reunião do primeiro *Sprint Review* deste projeto, foram solicitados novos requisitos pelo *Product Owner*, que entraram com prioridade máxima no segundo *Sprint*.

Após a reunião do segundo *Sprint Review* o *Product Owner* não solicitou nenhuma mudança no *Product Backlog*.

5.2.4 Sprint Retrospective

No final de cada *Sprint* o SCRUM *Master* se reunia com o Time e esse respondia a duas perguntas.

1. O que foi bom durante o Sprint?
2. O que pode ser melhorado?

Com estas informações o SCRUM *Master* avaliava os pontos que poderiam ser melhorados e fornecia recursos necessários para que as mudanças ocorressem.

5.2.4.1 Projeto MAXIMUS

Durante o primeiro Sprint *Retrospective* deste projeto, foi perguntado ao Time o que foi bom durante o *Sprint*, e esta respondeu que o planejamento das atividades foi essencial para o término das tarefas do primeiro *Sprint*. Foi perguntado também o que poderia ser melhorado para os próximos *sprints* e esta respondeu que seria melhor quebrar os itens do *Product Backlog* que eram feitas durante o *Sprint* em tarefas menores no Sprint *Backlog*.

Durante o segundo Sprint *Retrospective* foram feitas as mesmas perguntas do primeiro *Sprint Retrospective* e o Time respondeu que a quebra dos itens do *Product Backlog* em tarefas menores, compondo assim o *Sprint Backlog*, foi essencial para essencial para o sucesso deste *Sprint*. Na segunda pergunta o Time respondeu que mais reuniões com o *Product Owner* seria o ideal.

5.2.4.2 Projeto HAFFNER

Durante o primeiro *Sprint Retrospective* deste projeto, foi perguntado ao Time o que foi bom durante o *Sprint*, e este respondeu que a definição do *Sprint* com *Time-Box* fixo juntamente com os itens do *Product Backlog* destinados ao primeiro *Sprint* foram essenciais para aumentar a visão do sistema a ser desenvolvido como um todo. Foi perguntado também ao Time o que poderia ser melhorado para os próximos S*prints* e este respondeu que mais reuniões com o *Product Owner* seria o ideal.

Durante o segundo *Sprint Retrospective* foram feitas as mesmas perguntas do primeiro *Sprint Retrospective* e o Time respondeu que as reuniões com o *Product Owner* foram de fundamental importância nesta fase que o projeto se encontrava. Na segunda pergunta o Time respondeu que a princípio estava tudo dentro do planejado e não havia pontos a serem melhorados.

5.3 Product Backlog

O *Product Backlog* é o artefato construído pelo *Product Owner* juntamente com o SCRUM *Master*. O *Product Backlog* de cada projeto contém estimativas feitas pelo Time através da técnica do *Planning Poker*.

5.3.1 Projeto MAXIMUS

Na reunião de Planejamento do *Sprint*, O SCRUM *Master,* juntamente com o *Product Owner,* definiram o *Product Backlog* do projeto MAXIMUS (vide tabela 3). Nesta tabela, conhecida como *Product Backlog*, é apresentado em qual *Sprint* cada item foi realizado, e contém também o esforço medido pelo Time através da técnica do *Planning Poker*.

Tabela 3 – *Product Backl*og do Projeto MAXIMUS

ID	Item	*Sprint*	Esforço
1	Documentos	1	3
2	Definir Gramática	1	5
3	Definição de operadores	1	3
4	Gerar regras iniciais aleatórias	1	3
5	Operadores para gerar alterações em árvores	1	5
6	Documentos	2	3
7	Taxa de Corretagem	2	5
8	Taxa de Emolumento	2	5
9	Taxa de Imposto de Renda	2	5
10	Taxa CBLC	2	5
11	Carteira de Dinheiro	2	1
12	Carteira de Ações	2	2
13	Operações realizadas (Extrato)	2	3
14	Comparativo Lucro e Drawndown (Desvalorização máxima da Carteira)	2	3
15	Comparativo Lucro Drawndown e quantidade de Operações	3	3
16	Método Simulated Annealing	3	5
17	Testes Simulated Annealing	3	2
18	Método Hill Climbing	3	2
19	Testes Método Hill Climbing	3	2

ID	Item	*Sprint*	Esforço
20	Acréscimo de Indicadores	3	5
21	Revisão dos Operadores de Mutação	4	5

5.3.2 Projeto HAFFNER

Na reunião de Planejamento do *Sprint*, O SCRUM *Master,* juntamente com o *Product Owner*, definiram o *Product Backlog* do projeto HAFFNER (vide tabela 4). Nesta tabela, conhecida como Product Backlog, é apresentado em qual *Sprint* cada item foi realizado, e contém também o esforço medido pelo Time através da técnica do *Planning Poker.*

Tabela 4 – *Product Backlog* do Projeto HAFFNER

ID	Item	*Sprint*	Esforço
1	Documentação	1	3
2	Design Pagina Inicial	1	1
3	Design Administrador	1	1
4	CRUD Empresa	1	3
5	Design RH Login	1	1
6	CRUD Característica	1	2
7	CRUD Pessoas	1	3
8	Agrupamento Harmônico	1	5
9	Tela de Resposta	1	3
10	Design Equipe Login	1	1
11	Questionário	1	1
12	Atualizações das telas junto ao *Product Owner*	2	5
13	Cadastro de competências	2	2
14	Configuração de Agrupamento Cargo	2	3
15	Configuração de Agrupamento habilidade	2	5
16	Configuração de Agrupamento pessoal	2	3
17	Configuração de Agrupamento para cadastrar outras habilidades	2	3
18	Estudos sobre técnicas de otimização combinatoria	3	5
19	Método Hill Climbing separar equipes	3	3

ID	Item	*Sprint*	Esforço
20	Teste Método Hill Climbing separar equipes	3	2
21	Método Simulated Annealing separar equipes	3	5
22	Teste Método Simulated Annealing separar equipes	3	3
23	Algoritmo Genético separa equipes	3	5
24	Teste Algoritmo Genético separa equipes	4	3
25	Enviar email pelo sistema	4	5
26	Funcionar com os parametros	4	5
27	Modificar Visual	4	5

5.4 *Burndown Chart* dos *Sprints*

Durante o andamento de cada projeto foi traçado o Burndown Chart de cada *Sprint*. Com ele podemos perceber se o projeto está sendo desenvolvido conforme o planejamento, se está atrasado ou até mesmo adiantado.

5.4.1 Projeto MAXIMUS

O andamento de todo o primeiro *Sprint* do projeto MAXIMUS, pode ser visualizado de acordo com a figura 7. Nesta figura percebemos que durante aproximadamente 13 dias, o projeto estava atrasado se comparado com o planejado, mas no último dia o Time conseguiu entregar as funcionalidades propostas dentro do prazo estimado. Podemos perceber também que o Time trabalhou no primeiro *Sprint* com um esforço de 19 pontos, acordado pelo Time.

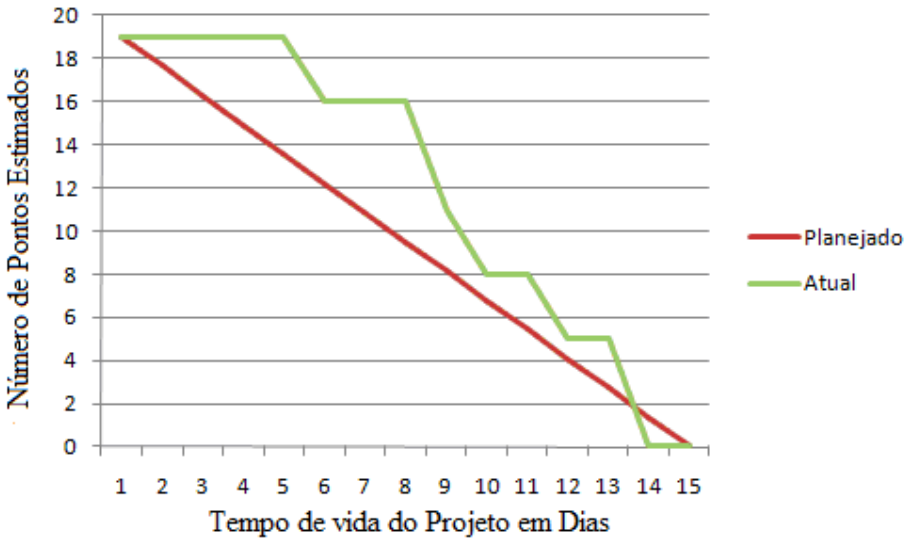


Figura 7 – Burndown Chart Primeiro Sprint

A pontuação do primeiro *Sprint* serviu de base para o planejamento do segundo *sprint*, que pode ser visto de acordo com a figura 8. Neste *Sprint* o SCRUM *Master* junto com o Time definiram que iriam trabalhar com um esforço de 24 pontos.

Durante este *Sprint* um fato alarmante aconteceu. O Time, no Segundo Sprint demonstrou não ter total conhecimento sobre todos os itens do Product Backlog, pois durante o *Planning Poker,* estimou quatro itens do *Product Backlog* com um esforço de 5 pontos cada, totalizando um esforço 20 pontos, sendo que estes itens foram cumpridos bem antes do que o estipulado. Durante as reuniões diárias o SCRUM *Master,* juntamente com o *Product Owner,* percebendo que os itens alocados para o segundo *Sprint* iriam ser finalizados antes do término do *Sprint*, decidiu antecipar funcionalidades que estavam planejadas para o terceiro *Sprint*, alocando assim itens do *Product Backlog* que somaram 9 pontos de esforço, para serem cumpridos no segundo *Sprint*, por esta razão a figura 8 sofreu um pico do oitavo dia ao nono dia.

Mesmo com este acréscimo de funcionalidades as funcionalidades foram entregues no prazo planejado.

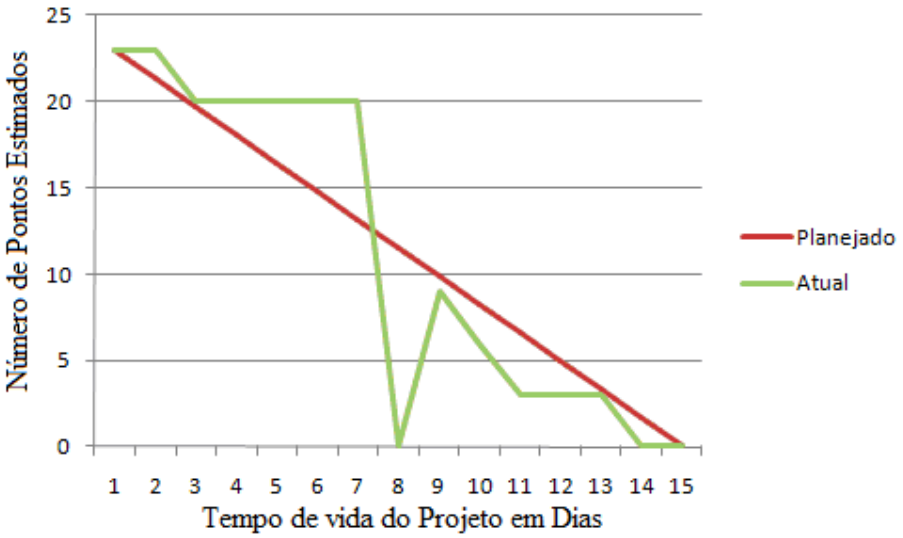


Figura 8 – Burndown Chart Segundo Sprint

Após o fato ocorrido no segundo *Sprint* o SCRUM *Master* reuniu o Time e definiram em consenso que o terceiro *Sprint* seria trabalhado com uma estimativa de 19 pontos, como feito no primeiro *Sprint*.

O Terceiro *Sprint* ainda não foi finalizado, como visto na figura 9.

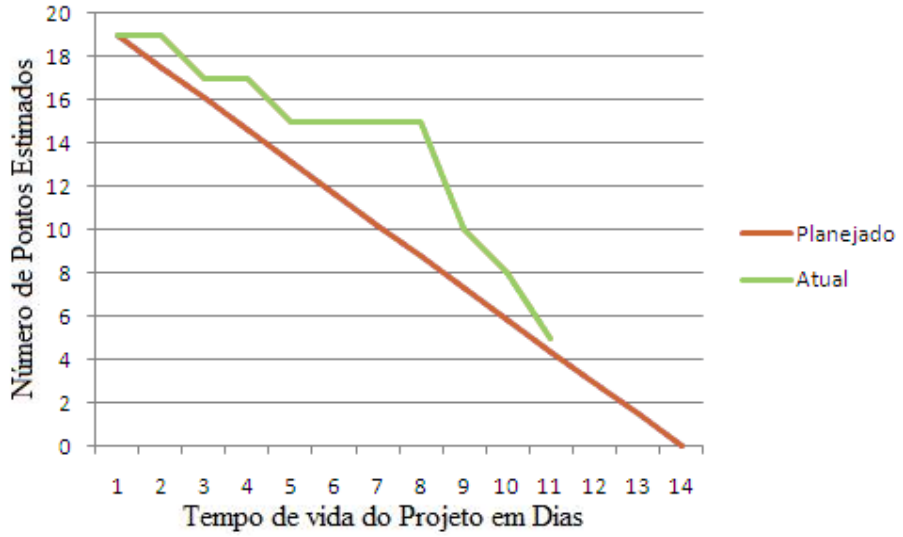


Figura 9 – Burndown Chart Terceiro Sprint

5.4.1.1 Estimativa de término do Projeto MAXIMUS com base no primeiro *Sprint*

Com base neste primeiro *Sprint*, figura 7, o SCRUM *Master* estimou que o projeto MAXIMUS, teria uma duração de quatro *Sprints*. Se em todos os *Sprints* trabalhássemos com um esforço de dezenove pontos e o *Product Owner* não solicitasse nenhuma mudança do *Product Backlog*, no final de quatro *Sprints* atingiríamos um total de setenta e seis pontos, sendo que todo o *Product Backlog* possui hoje um total de setenta e cinco pontos de esforço. A realização do projeto em quatro *Sprints* é a mais indicada devido ao total de pontos de esforço do Product Backlog e dos pontos de esforço estipulados para cada *Sprint*.

5.4.2 Projeto HAFFNER

O andamento de todo o primeiro *Sprint* do projeto HAFFNER, pode ser visualizado de acordo com a figura 10. Nesta figura percebemos que durante todo o projeto o Time cumpriu suas atividades em um prazo menor, se comparado ao planejado. Podemos perceber também que a equipe trabalhou no primeiro *Sprint* com um esforço de vinte e quatro pontos, acordado pelo Time.

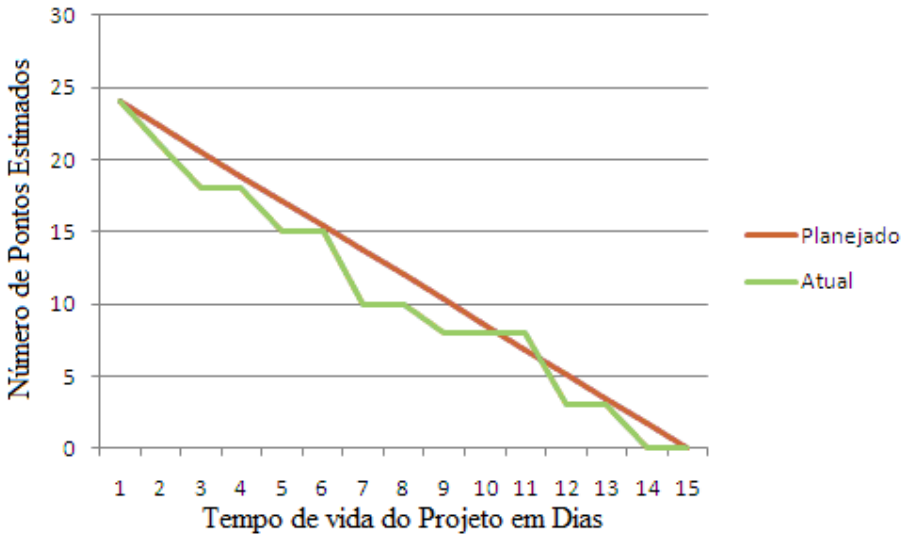


Figura 10 – Burndown Chart Primeiro *Sprint*

A pontuação do primeiro *Sprint* serviu de base para o planejamento do segundo *sprint*, que pode ser visto de acordo com a figura 11. Neste *sprint* o SCRUM M*aster* junto com a equipe definiram que iriam trabalhar no segundo *Sprint* com um esforço de vinte e um pontos. O SCRUM M*aster* decidiu trabalhar neste *Sprint* com um esforço menor que a do primeiro *Sprint*, pois neste *Sprint* a equipe iria reunir uma quantidade de vezes maior do que a do primeiro *Sprint* com o *Product Owner*, afim de melhorarem as telas definidas no *Product Backlog*. Por esta razão o esforço para o segundo *Sprint* sofreu uma redução.

Durante os três primeiros dias do segundo *Sprint* o projeto não andou conforme o planejado, devido às reuniões da equipe com o *Product Owner*, para a definição das telas. Passado a definição destas a equipe conseguiu dar prosseguimento no projeto, entregando as funcionalidades potencialmente implantáveis no prazo planejado.

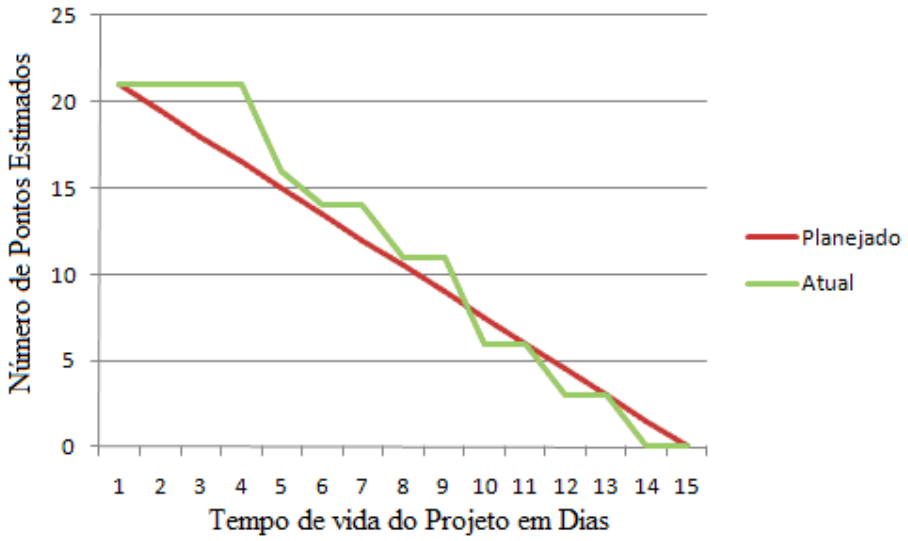


Figura 11 – Burndown Chart Segundo *Sprint*

Após o segundo *Sprint*, o SCRUM *Master* juntamente com o Time definiram que para o terceiro *Sprint* o esforço seria maior do que o *Sprint* passado, pois neste as reuniões com o *Product Owner* seriam menos freqüentes, assim o esforço estipulado foi de vinte e três pontos (Vide figura 12).

Durante os 9 primeiros dias o projeto não andou conforme o planejado, no 11° dia o Time conseguiu realizar as tarefas deixando o projeto no prazo

estipulado. Durante o 12° dia o projeto saiu do planejamento, mas no último dia as funcionalidades propostas foram entregues com sucesso.

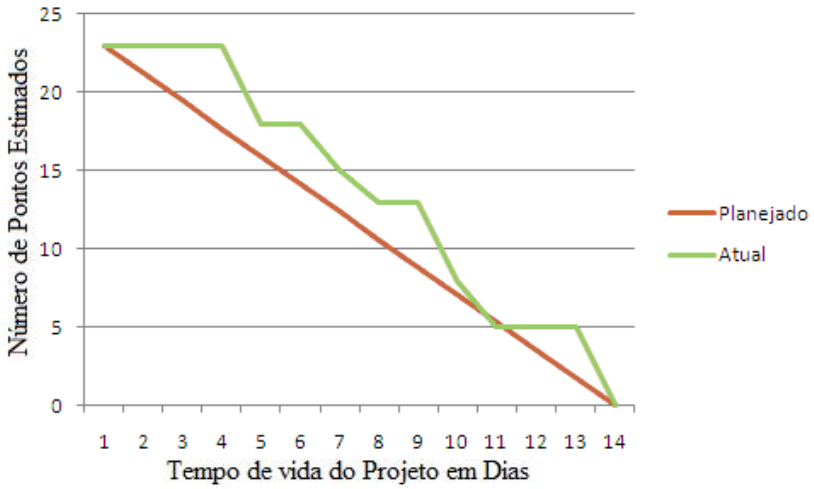


Figura 12 – Burndown Chart Terceiro *Sprint*

5.4.2.1 Estimativa de término do Projeto HAFFNER com base no primeiro *Sprint*

Com base neste primeiro *Sprint*, figura 10, o SCRUM *Master* estimou que o projeto HAFFNER, teria uma duração de quatro *Sprints*. Se em todos os *Sprints* trabalhássemos com um esforço de vinte e quatro pontos e o *Product Owner* não solicitar nenhuma mudança do *Product Backlog*, no final de quatro *Sprints* atingiríamos um total de noventa e seis pontos, sendo que todo o *Product Backlog* possui hoje um total de oitenta e seis pontos de esforço.

A possibilidade de trabalharmos com três *Sprints* não é possível, pois assim em um *Sprint* teríamos que trabalhar com um esforço de trinta e oito pontos, o que não é aconselhável devido a produtividade da equipe ser de vinte e quatro pontos por *Sprint*.

5.5 CMMI nível 2

Com o intuito de melhorar os processos da fábrica de software e em conseqüência melhorar a qualidade dos softwares produzidos nela, os representantes da fábrica de software se reuniram, e após um estudo detalhado, decidiram que a fábrica de software do LP&D, utilizaria o método ágil SCRUM para gerenciar os seus projetos e também tentar obter a certificação do CMMI nível 2.

Após esta reunião, foi estipulado um prazo de duas semanas, para que todos os projetos do LP&D passasem a ser gerenciados seguindo o método ágil SCRUM. Este prazo foi cumprido e após isso foi iniciado um estudo, com o objetivo de identificar as áreas de processo necessárias a serem cumpridas para que a fábrica de software obtenha a certificação CMMI nível 2.

A fábrica de software optou por utilizar a representação por estágios pois, com esta representação é possível traçar um caminho evolutivo pré-definido para a melhoria dos processo, e também com esta representação podemos avançar estágio por estágio afim de melhorar todo o processo de construção de software.

Não optamos na escolha da representação continua, pois ela é apropriada para organizações que querem melhorar uma área de processo especifica, e no caso da fábrica de software todos os processos devem ser melhorados.

5.6 Utilização do SCRUM para atingir áreas de processo do CMMI nível 2

Com o objetivo de tornar os processos da fábrica de software condizentes com o que o CMMI emprega, utilizamos o SCRUM na tentativa de atingir as sete áreas de processo.

A área de processo Gestão de requisitos contém um conjunto de atividades que engloba receber as solicitações de alteração de requisitos, registrar novos requisitos, analisar impacto de mudança de requisitos, notificar os envolvidos e Coletar métricas (HAZAN, 2003).

Esta área de processo começa a ser atingida a partir da primeira reunião do projeto onde o SCRUM *Master* se reúne com o *Product Owner* para construírem o *Product Backlog* do sistema. O *Product Backlog* contém todos os requisitos que devem conter no produto.

As mudanças de requisitos podem ocorrer a qualquer momento, ficando a cargo do *Product Owner*, pois a todo instante novos requisitos podem surgir, mas o SCRUM *Master* deve evitar que muitas mudanças ocorram. Caso haja alguma mudança nos requisitos de um *Sprint*, o SCRUM *Master* retira deste as funcionalidades que estavam planejadas, que possuem o mesmo esforço que a nova funcionalidade requerida. Feito isso as funcionalidades retiradas são alocadas para o próximo *Sprint*, assim um outro planejamento para o projeto é realizado, modificando o *Product Backlog*.

A área de processo planejamento de Projeto contém um conjunto de atividades que envolvem elaboração do plano de projeto. O planejamento inclui a estimativa de atributos de produtos de trabalho e de tarefas, a determinação de recursos necessários, a negociação de compromissos, a elaboração de um cronograma, e a identificação e análise de riscos do projeto.

Esta área de processo é atingida utilizando o SCRUM quando é realizado o planejamento de todo o projeto. Este planejamento é feito durante as reuniões com o *Product Owner*, onde é verificada a viabilidade do projeto e a definição do *Product Backlog*. Também a atingimos quando realizamos reuniões com toda a equipe definindo o *Sprint Backlog*.

A área de processo monitoramento e controle de projeto têm o objetivo de fornecer informações com o intuito de tornar visível o progresso do projeto, pois caso o desempenho do projeto seja insatisfatório, medidas corretivas podem ser implementadas , fazendo com que o projeto tenha sucesso.

Esta área de processo contém mecanismos que controlam e monitoram as atividades realizadas nos projetos. Este controle e monitoramento são feitos utilizando reuniões diárias, onde a equipe de desenvolvimento juntamente com o SCRUM M*aster* apontam o que foi feito no final do dia, e o SCRUM *Master* aponta os avanços e os problemas. O Burndown Chart também é utilizado, pois com ele o SCRUM *Master* visualiza o andamento do projeto junto a velocidade da equipe e ao

final do *Sprint*, através do *Sprint Retrospective*, no qual a equipe aponta melhorias que poderiam ser postas em práticas a partir do próximo *Sprint*.

A área de processo gestão de contrato com fornecedores têm o objetivo de gerenciar a aquisição de produtos e/ou serviços de fornecedores. Esta área de processo contém um conjunto de atividades que envolvem seleção de fornecedores, estabelecimento e manutenção de contratos com fornecedores, execução do contrato com fornecedores, transferência dos produtos adquiridos para o projeto, entre outros.

Na fábrica de software abordamos esta área de processo utilizando contrato de escopo fechado. Este tipo de contrato se caracteriza por fixar os prazos, os custos, e o escopo, dando a garantia para o cliente de que o escopo solicitado será entregue a um custo fixo. Caso o *Product Owner* queira realizar mudanças no escopo, ele deverá negociar essas mudanças sofrendo as sanções de acordo com o que foi assinado em contrato.

A área de processo medição e análise têm o objetivo fornecer informações a respeito dos projetos. Esta área contém um conjunto de atividades que envolve: a especificação de técnicas de análise e mecanismos para coleta e armazenamento de dados, formas de relato e de feedback, fornecimento de resultados objetivos que possam ser utilizados na tomada de decisões bem fundamentadas e na implementação de ações corretivas apropriadas. Estas atividades possibilitam planejar e estimar de forma objetiva, acompanhar o desempenho em relação aos objetivos traçados e identificar e tratar questões críticas.

No SCRUM esta área de processo é atingida através das reuniões diárias e também através do *Sprint Retrospective*.

Nas reuniões diárias o SCRUM *Master* possui todas as informações referentes ao que foi planejado de ser realizado durante o *Sprint*. Durante o andamento do projeto este possui visão total do que já foi realizado, do que falta a ser realizado e o tempo disponível para que tais requisitos sejam cumpridos. Através disto o SCRUM *Master* possui informações suficientes para relatar as partes interessadas, informações a respeito do andamento do projeto e até mesmo possibilitar a este tomar medidas corretivas caso o projeto não esteja correndo conforme o planejado.

Durante o *Sprint Retrospective* a equipe relata ao SCRUM *Master* o que foi bom e o que não foi durante o *Sprint*. Nesta reunião também são levantados pontos de melhorias que podem ser realizadas para os próximos *Sprints*.

A área de processo Garantia da Qualidade de Processo e Produto têm o objetivo de mostrar tanto para a equipe quanto para a gerência os processos e produtos associados ao trabalho. Esta área contém um conjunto de atividades que envolvem avaliação objetiva dos processos executados, produto de trabalho e serviços em relação às descrições de processo, padrões e procedimentos aplicáveis, identificação e documentação das não conformidades, fornecimento de feedback à equipe do projeto e aos gerentes sobre os resultados das atividades de garantia da qualidade.

Esta área de processo é parcialmente atingida pelo SCRUM através das reuniões diárias e pelo *Sprint Retrospective*, pois com eles garantimos o feedback do andamento do projeto de forma geral a equipe do projeto e ao gerente, mas não podemos dizer que o SCRUM abrange esta área de processo porque ele a prevê por meio da experiência adquirida, não existindo um relatório formal que avalie os processos.

A área de processo Gestão de Configuração tem o objetivo de estabelecer e manter a integridade dos produtos. Esta área contém um conjunto de atividades que envolve controle de mudanças nos itens de configuração, manutenção da integridade de baselines, disponibilização do status e dos dados atuais de configuração para desenvolvedores, usuários finais e clientes. Os produtos de trabalho aqui incluem os produtos que são entregues ao cliente, produtos de trabalho internos selecionados, produtos adquiridos, ferramentas e outros itens que são utilizados para criar e descrever esses produtos de trabalho.

Esta are não é atingida, pois o SCRUM não prevê o controle de versões, mas para abordar esta área de processo estamos estudando a viabilidade de se usar o Subversion (PILATO, 2004) mais conhecido como SVN. Esta aplicação cliente/servidor permite a edição colaborativa de usuários, o compartilhamento de dados e a configuração de mudanças.

6
Conclusões

Neste capítulo são apresentadas as conclusões que foram obtidas a partir dos experimentos realizados nesta monografia.

6.1 Considerações Iniciais

Com este trabalho podemos concluir que após a implantação do método ágil SCRUM no desenvolvimento dos projetos da fábrica de software do LP&D, obtivemos resultados satisfatórios, no planejamento, desenvolvimento e principalmente nas entregas dos sistemas.

Os Times juntamente com o SCRUM *Master* têm cumprido todos os prazos planejados. Isso se deve principalmente às reuniões diárias juntamente com a criação/manutenção do Burndown Chart, que fornece ao SCRUM *Master* uma visão bem clara de como está o andamento do projeto.

Em relação ao nível de maturidade 2 do CMMI, notamos que o SCRUM não aborda todas, mas a maioria das áreas de processos necessárias a obtenção da certificação CMMI 2.

A única área de processo não abordada pelo CMMI nível 2 é a gerência de configuração. Mesmo não abordando esta área, não podemos afirmar que a utilização do SCRUM para obtenção desta certificação não é viável, pois podemos abordá-la utilizando o Subversion.

No momento a fábrica de Software não esta funcionando de acordo com o que o CMMI nível 2 aborda, pois ainda não implantamos uma ferramenta de gestão de configuração. Atualmente, estamos testando o Subversion, umas das possíveis ferramentas presentes no mercado.

6.2 Trabalhos Futuros

Trabalhos futuros derivados deste trabalho podem ser realizados como a verificação de se adequar as áreas de processo da fábrica de software com as áreas de processo abordadas pelo CMMI nível 3. Este trabalho futuro seria de fundamental importância para que a fábrica de software continue evoluindo cada vez mais, e em longo prazo ter todos os seus processos adequados aos processos abordados desde o CMMI nível 2 ao CMMI nível 5.

7 Referências Bibliográficas

Bassi, D. L. F. Colli, E. *Experiências com desenvolvimento ágil*. Dissertação Universidade de São Paulo. Março de 2008.

Beck, K. et all. *The Agile Manifesto*,2001. Disponível em: <http://agilemanifesto.org/>. Acessado em: 23 Ago. 2010.

Beck, K. *Extreme Programming Explained*. Publisher: First Edition September 29, 1999.

Carleton, A. D. et all. *Software Measurement for DoD Systems: Recommendations for Initial Core Measures*. September 1992.

Chrissis, M.B., Konrad, M., Shrum, S. *CMMI Guidelines for Process Integration and Product Impovement*, SEI Series in Software Engineering, Ed. Publisher Addison Wesley, 2004.

CMMI Product Team. *CMMI for Development, version 1.2*. Carnegie Mellon, Software Engineering Institute. Pittsburgh, PA 15213-3890. August 2006.

Cockburn, A. *Crystal clear a human-powered methodology for small teams*. Publisher Addison-Wesley Professional 2004.

Deemer, P. Benefield, G. Larman, C. Vodde B. *The SCRUM Primer*, version 1.2, 2010

Dinakar, K. *Agile development: Overcoming a short-term focus in implementing best practices*, ACM Special Interest on Programming Languages an Applications, 2009, Orlando, Florida, USA, Pages: 579-588.

Dominguez, J. *The curious case of the chaos report 2009*. Disponível em: <http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html>. Acessado em: 31 Out. 2010.

Franco, M. A. S. *Pedagogia da pesquisa-ação*. Educação e Pesquisa, São Paulo, 2005.

Fowler, M. *The New Methodology*. 2001. Disponível em: <http://www.martinfowler.com/articles/newMethodology.h>. Acessado em: 26 Ago. 2010.

Grenning, J. *Planning Poker or How to avoid analysis paralysis while release planning*, April 2002.

Hazan, C. Leite, J. C. S. P. *Indicadores para a Gerência de Requisitos*. Anais do WER03 - Workshop em Engenharia de Requisitos, pp. 285-301, Piracicaba, SP, 2003

Highsmith, J. *History The Agile Manifesto*, 2001. Disponível em: <http://www.agilemanifesto.org/history.html>. Acessado em: 15 Ago. 2010

Hundermark, P. *Do Better SCRUM*, Cape Town Second edition, November 2009

Hunt, A. Thomas, D. *The Pragmatic Programmer*. Publisher Addison-Wesley, 2000

Lévesque, G. Ktata, O. *Designing and Implementing a Measurement Program for SCRUM Teams: What do agile developers really need and want?* ACM International Conference Proceeding Series, Proceedings of the Third C* Conference on Computer Science and Software Engineering, May 2010. Montréal, Quebec, Canada, Pages: 101-107

Nebulon Pty. Ldt. Agile *Software Development using Feature Driven Development (FDD)*. Disponível em: <http://www.nebulon.com/fdd/index.html>. Acessado em: 25 Ago. 2010

Pilato,C., COLLINS-SUSSMAN, M., FITZPATRICK, W. Brian. Version Control with Subversion.2004

Rubart, J. Freykamp, F. *Supporting Daily SCRUM Meetings with Change Structure*, Proceedings of the 20th ACM conference on Hypertext and hypermedia, June 2009. Torino, Italy, Pages: 57-62

Schwaber, K. *Agile Project Management with SCRUM*, Published by Microsoft Redmond, WA, 2004

Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Publisher Addison-Wesley, 1995.

Sotille, M. A. O *gerenciamento de Projetos de Software e a implementação do CMMI*. Pmtech capacitação em projetos, 2006.

Stuart, J. et all. *Construx Software, 10 Keys to Successful SCRUM Adoption ,version 1.2*, April 2010.

Systems Security Engineering. *Capability Maturity Model Appraisal Method*, version 2.0. Apr 1999.

Teles, V. M. *Manifesto Agil*, 2010. Disponível em: <http://improveit.com.br/xp/manifesto_agil/>. Acessado em: 23 Ago. 2010.

The Standish Group. *New Chaos numbes show Startling Results*. Boston, Massachusetts, 23 de abril de 2009. Disponível em: <http://standishgroup.com/newsroom/chaos_2009.php>. Acessado em: 02 Nov. 2010.

THREEWILL *Choose to Succeed. Our Process*. 2005. Disponível em: <http://www.threewill.com/company/home/ourprocess/>. Acessado em: 24 Nov. 2010.

8 Apêndices e Anexos

8.1 Anexo I

8.1.1 Projeto MAXIMUS

8.1.1.1 Primeiro *Sprint Backlog*

ID	Item	ID Product Backlog
1	Definir as gramáticas	2
2	Implementar Classes OO	2
3	Design Patterns	2
4	Documento Caso de Uso	1
5	Diagrama de Classe	1
6	Método de gerar regras iniciais	4
7	Definição Operadores para alterar as arvores	5
8	Implementação para gerar alteração em arvores	5

8.1.1.2 Reuniões Diárias Primeiro *Sprint Backlog*

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
26/08	Início	Definição da Gramática	Não
30/08	Definição da Gramática (Parcial)	Definição da Gramática	Não

Dia	**O que já foi feito?**	**O que eu pretendo fazer ate amanhã?**	**Têm alguma coisa impedindo meu trabalho?**
31/08	Definição da Gramática (Parcial)	Definição da Gramática	Falta de Material Bibliográfico (Indicadores da Bolsa de Valores)
01/09	Definição da Gramática	Documento caso de Uso	Não
02/09	Documento caso de Uso	Diagrama de Classe	Material livro Koza
03/09	Diagrama de classe	Estudo de Operadores Definir operadores para gerar alteração em árvores (alteração aleatória)	Não
08/09	Estudo de Operadores Definir operadores para gerar alteração em árvores (alteração aleatória)	Implementar Classes OO Design Patterns	Não
09/09	Implementar Classes OO (Parcial) Design Patterns (Parcial)	Implementar Classes OO Design Patterns	Não
10/09	Implementar Classes OO Design Patterns	Método para gerar regras iniciais Estudo de operadores	Não
13/09	Métodos para gerar regras iniciais	Definir operadores Implementação para gerar alteração em arvore	Não

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
14/09	Definir operadores (Parcial)	Definir operadores Implementação para gerar alteração em arvore	Não
15/09	Definir operadores para alterar as arvores	Implementação para gerar alteração em arvore	Não
16/09	Implementação para gerar alteração em arvore (parcial)	Implementação para gerar alteração em arvore	Não
17/09	Implementação para gerar alteração em arvore testes	Reunião Sprint Review	Não

8.1.1.3 Segundo *Sprint Backlog*

ID	Item	ID Product Backlog
1	Documento Requisitos	6
2	Implementar Camada Objeto relacional	7,8,9,10
3	Criação, manipulação BD	7,8,9,10
4	Implementação taxas	7,8,9,10
5	Modelagem Carteira de Dinheiro	11
6	Modelagem Carteira de Ações	12
7	Implementação Carteira de Dinheiro	11
8	Implementação Carteira de Ações	12

ID	Item	ID Product Backlog
9	Operações realizadas	13
10	Comparativo Lucro e Drawndown	14

8.1.1.4 Reuniões Diárias Segundo *Sprint Backlog*

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
29/09	Revisão do Código em andamento	Revisão de Código	Não
30/09	Revisão do Código	Documento de Requisitos e Modelagem do BD.	Não
04/10	Documento de requisitos e Modelagem do BD	Implementação das taxas Corretagem Implementação das taxas Emolumento Implementação das taxas de Imposto de renda Implementação das taxas CBLC	Não
05/10	Implementação das taxas (Parcial)	Implementação das taxas	Não
06/10	Implementação de todas as taxas	Implementação objeto relacional Modelagem Carteira dinheiro Modelagem Carteira de ação	Não

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
07/10	Modelagem das carteira dinheiro & carteira de ação	Implementação objeto relacional	Não
19/10	Implementação objeto relacional (Parcial)	Implementação objeto relacional	Não
20/10	Implementação objeto relacional	Implementação carteira dinheiro Implementação carteira de ação	Não
21/10	Implementação carteira dinheiro (Parcial) Implementação carteira de ação (Parcial)	Definição da função objetivo. Testes no Agente	Não
22/10	Implementação carteira dinheiro Implementação carteira de ação	Operações realizadas Extrato	Não
25/10	Operações realizadas (Extrato)	Comparativo Lucro e Drawndown	Não
26/10	Comparativo lucro e Drawndown (Parcial)	Comparativo lucro e Drawndown	Não
27/10	Comparativo lucro e Drawndown (Parcial)	Comparativo lucro e Drawndown	Não
28/10	Comparativo lucro e Drawndown	Sprint Restrospective	Não

8.1.1.5 Terceiro *Sprint Backlog*

ID	Item	ID Product Backlog
1	Comparativo Lucro Drawndown e quantidade de Operações	15
2	Método Simulated Annealing	16
3	Testes Simulated Annealing	17
4	Método Hill Climbing	18
5	Testes Método Hill Climbing	19
6	Acréscimo de Indicadores	20

8.1.1.6 Reuniões Diárias Terceiro *Sprint Backlog*

O terceiro *Sprint* esta em andamento.

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
03/11	Método Hill Climbing (Parcial)	Método Hill Climbing	Não
04/11	Método Hill Climbing(Parcial)	Método Hill Climbing	Não
05/11	Método Hill Climbing	Testes Método Hill Climbing	Não
08/11	Testes Método Hill Climbing (Parcial)	Testes Método Hill Climbing	Não
09/11	Testes Método Hill Climbing	Método Simulated Annealing	Não
10/11	Método Simulated Annealing (Parcial)	Método Simulated Annealing	Não

Dia	**O que já foi feito?**	**O que eu pretendo fazer ate amanhã?**	**Têm alguma coisa impedindo meu trabalho?**
11/11	Método Simulated Annealing (Parcial)	Método Simulated Annealing	Não
12/11	Método Simulated Annealing (Parcial)	Método Simulated Annealing	Não
15/11	Método Simulated Annealing	Testes Método Simulated Annealing	Não
16/11	Testes Método Simulated Annealing	Comparativo Lucro Drawndown e quantidade de Operações	Não
17/11	Comparativo Lucro Drawndown e quantidade de Operações	Acréscimo de Indicadores	Não
18/11	Acréscimo de Indicadores (PArcial)	Acréscimo de Indicadores	Não

8.2 Anexo II

8.2.1 Projeto HAFFNER

8.2.1.1 Primeiro *Sprint Backlog*

ID	Item	ID Product Backlog
1	Documentação	1
2	Design Pagina Inicial	2
3	Design Administrador	3
4	CRUD Empresa	4
5	Design RH Login	5
6	CRUD Característica	6
7	CRUD Pessoas	7
8	Agrupamento Harmônico	8
9	Tela de Resposta	9
10	Design Equipe Login	10
11	Questionário	11

8.2.1.2 Reuniões Diárias Primeiro *Sprint Backlog*

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
24/08	Documentação	Pagina Inicial Administrador RH login	Não

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
25/08	Pagina Inicial Administrador RH login	CRUD empresa	Não
26/08	CRUD empresa (Parcial) Foi acrescentado Crud de Login que não estava no escopo	CRUD empresa	Não
27/08	CRUD empresa	CRUD pessoa	Não
30/08	CRUD Pessoa (Parcial)	CRUD pessoa CRUD Característica	Estudo de Tecnologia
31/08	CRUD Pessoa CRUD Característica	Questionário	Não
01/09	Questionário (Parcial)	Questionário Equipe Login	Não
02/09	Questionário Equipe Login	Agrupamento Harmônico	Material livro Koza
03/09	Agrupamento Harmônico (Parcial)	Agrupamento Harmônico Tela de Resposta	Não
06/09	Agrupamento Harmônico (Parcial)	Agrupamento Harmônico	Não
07/09	Agrupamento Harmônico	Tela de Resposta	Não
08/09	Tela de Resposta (Parcial)	Tela de Resposta	Não
09/09	Tela de Resposta (Parcial	Tela de Resposta	Não

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
10/09	Tela de Resposta	Reunião Sprint Review	Não

8.2.1.3 Segundo *Sprint Backlog*

ID	Item	ID Product Backlog
1	Design Pagina Inicial	12
2	Design Administrador	12
3	Design RH Login	12
4	Tela de Resposta	12
5	Design Equipe Login	12
6	Cadastro de competências	13
7	Configuração de Agrupamento Cargo	14
8	Configuração de Agrupamento habilidade	15
9	Configuração de Agrupamento pessoal	16
10	Configuração de Agrupamento para cadastrar outras habilidades	17
11	Design Pagina Inicial	12

8.2.1.4 Reuniões Diárias Segundo *Sprint Backlog*

Dia	**O que já foi feito?**	**O que eu pretendo fazer ate amanhã?**	**Têm alguma coisa impedindo meu trabalho?**
20/09	Design Pagina Inicial	Design Pagina Administrador	Não
Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
21/09	Design Pagina Administrador	Design RH Login	Não
22/09	Design RH Login	Design Tela de Resposta	Não
23/09	Design Tela de Resposta	Design Equipe Login	Não
24/09	Design Equipe Login	Cadastro de competências	Não
27/09	Cadastro de competências	Configuração de Agrupamento Cargo	Não
28/09	Configuração de Agrupamento Cargo (Parcial)	Configuração de Agrupamento Cargo	Não
29/09	Configuração de Agrupamento Cargo	Configuração de Agrupamento habilidade	Não
30/09	Configuração de Agrupamento habilidade (Parcial)	Configuração de Agrupamento habilidade	Não
01/10	Configuração de Agrupamento habilidade	Configuração de Agrupamento pessoal	Não

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
04/10	Configuração de Agrupamento pessoal (Parcial)	Configuração de Agrupamento pessoal	Não
05/10	Configuração de Agrupamento pessoal	Configuração de Agrupamento para cadastrar outras habilidades	Não
06/10	Configuração de Agrupamento para cadastrar outras habilidades (Parcial)	Configuração de Agrupamento para cadastrar outras habilidades	Não
07/10	Configuração de Agrupamento para cadastrar outras habilidades	Sprint Restrospective	Não

8.2.1.5 Terceiro *Sprint Backlog*

ID	Item	ID Product Backlog
1	Estudos sobre técnicas de otimização combinatória	18
2	Método Hill Climbing separar equipes	19
3	Teste Método Hill Climbing separar equipes	20
4	Método Simulated Annealing separar equipes	21
5	Teste Método Simulated Annealing separar equipes	22

ID	Item	ID Product Backlog
6	Algoritmo Genético separa equipes	23

8.2.1.6 Reuniões Diárias Terceiro *Sprint Backlog*

O terceiro *Sprint* esta em andamento.

Dia	O que já foi feito?	O que eu pretendo fazer ate amanhã?	Têm alguma coisa impedindo meu trabalho?
15/10	Estudos sobre técnicas de otimização combinatória (Parcial)	Estudos sobre técnicas de otimização combinatória	Não
18/10	Estudos sobre técnicas de otimização combinatória (Parcial)	Estudos sobre técnicas de otimização combinatória	Não
19/10	Estudos sobre técnicas de otimização combinatória (Parcial)	Estudos sobre técnicas de otimização combinatória	Não
20/10	Estudos sobre técnicas de otimização combinatória (Parcial)	Estudos sobre técnicas de otimização combinatória	Não
21/10	Estudos sobre técnicas de otimização combinatória	Método Hill Climbing separar equipes	Não
22/10	Método Hill Climbing separar equipes (Parecial)	Método Hill Climbing separar equipes	Não

Dia	**O que já foi feito?**	**O que eu pretendo fazer ate amanhã?**	**Têm alguma coisa impedindo meu trabalho?**
25/10	Método Hill Climbing	Testes Hill Climbing	Não
26/10	Testes Método Hill Climbing separar equipes	Método Simulated Annealing separar equipes	Não
29/10	Método Simulated Annealing separar equipes (Parcial)	Método Simulated Annealing separar equipes	Não
30/10	Método Simulated Annealing separar equipes	Testes Método Simulated Annealing separar equipes	Não
01/11	Testes Método Simulated Annealing separar equipes	Algoritmo Genético separa equipes	Não
02/11	Algoritmo Genético separa equipes (Parcial)	Algoritmo Genético separa equipes	Não
03/11	Algoritmo Genético separa equipes (Parcial)	Algoritmo Genético separa equipes	Não
04/11	Algoritmo Genético separa equipes	Sprint Retrospective	Não