Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

MARIANA CARREIRO DA SILVA

SCRUM

QUALIDADE E AGILIDADE NO DESENVOLVIMENTO DE PROJETOS

2010

Assis SP

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

SCRUM

QUALIDADE E AGILIDADE NO DESENVOLVIMENTO DE PROJETOS

Trabalho de Conclusão de
Curso apresentado ao Instituto
Municipal de Ensino Superior de
Assis, como requisito do Curso
de Processamento de Dados.
Orientanda: Mariana Carreiro da Silva
Orientador: Luiz Ricardo Begosso

2010

Assis SP

FICHA CATALOGRÁFICA

CARREIRO DA SILVA, Mariana
SCRUM – QUALIDADE E AGILIDADE NO DESENVOLVIMENTO DE PROJETOS / Mariana Carreiro da Silva. Fundação Educacional do Município de Assis – FEMA – Assis, 2010.
50p.

Orientador: Luiz Ricardo Begosso
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1.Scrum 2. Qualidade de Software.

CDD: 001.61
Biblioteca da FEMA

SCRUM

QUALIDADE E AGILIDADE NO DESENVOLVIMENTO DE PROJETOS

MARIANA CARREIRO DA SILVA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Processamento de Dados, analisado pela seguinte comissão examinadora:

Orientador: Luiz Ricardo Begosso

Avaliador: Luiz Carlos Begosso

2010

Assis SP

DEDICATÓRIA

Com muita saudade e eterno amor dedico
este trabalho à você Carlão, sei que me olha
e me guia ai de cima amigo...
A gente ainda se encontra.

AGRADECIMENTOS

Tantos fizeram parte dessa jornada. Tantos contribuíram comigo em tanta coisa que sem eles eu jamais teria concretizado essa conquista.

Aos mestres agradeço por tanto se dedicarem a mim, não somente por terem ensinado, mas por terem me feito aprender! Obrigada aos professores amigos, pacientes e dedicados, aos quais, sem nominar terão meu eterno agradecimento.

A Minha Família que sempre me fez entender que o futuro é feito a partir da constante dedicação no presente. Foram eles que formaram os fundamentos do meu caráter e apontaram uma vida eterna. Obrigada por serem a minha referência de tantas maneiras e estarem sempre presentes na minha vida de uma forma indispensável.

Aos meus amigos, minha segunda família, que fortaleceram os laços da igualdade, num ambiente fraterno e respeitoso. Cada um com sua particularidade, cada um com seu jeito de demonstrar afinidade e irmandade. Vocês são um presente na minha vida, jamais lhes esquecerei.

À minha amiga-irmã, pela companhia constante e tão querida, sacrifício ilimitado em todos os sentidos, orações, palavras, abraços, companhia e aconchego.

Ao meu professor e orientador deste trabalho, Luiz Ricardo Begosso, pelo desprendimento ao escolher me dar apoio, pela paciência e dedicação.

Muito obrigada nunca será suficiente para demonstrar a grandeza do que recebi de vocês. Peço a Deus que os recompense à altura. E é a Ele que dirijo minha maior gratidão. Deus, mais do que me criar, deu propósito à minha vida. A você meu DEUS, obrigada por cada momento da minha vida.

“Ama-se mais o que se conquista com esforço.”

Benjamin Disraeli

RESUMO

Este trabalho traz um breve histórico sobre a Qualidade, sua origem e evolução. Trata mais detalhadamente da Qualidade de Software, área estudada principalmente após os anos 70 pela Engenharia de Software.

Tratando-se de Qualidade de Software, existem várias metodologias para auxilio, padronização, orientação e avaliação do desenvolvimento dos projetos e das equipes. Neste trabalho são apresentadas algumas das metodologias, são elas: CMM, CMMI, PSP, MPS.BR e Scrum, com foco na última citada.

Além das informações detalhadas sobre a metodologia Scrum, este trabalho traz em seu quinto capítulo um exemplo real de aplicação da metodologia. É possível observar e acompanhar todas as fases de implantação do Scrum, desde a reunião inicial até a retrospectiva.

Através deste conteúdo o leitor conseguirá adquirir conhecimento necessário para adaptar a metodologia ao seu cotidiano e assim adotá-la, obtendo resultados muito satisfatórios, como o exemplo da empresa mostrado no quinto capítulo.

Palavras Chave: Scrum, Qualidade de Software

ABSTRACT

This work brings a brief history about the quality, its origin and evolution. It is more about Software Quality, the mainly studied area after the 70's by software engineering.

In this case, there are so many methodologies for help, standardization, orientation and rating of projects and teams developments. This work presents some methodologies, like: CMM, CMMI, PSP, MPS.BR and Scrum, with focus on the last one.

In addition to the detailed information about the Scrum's methodology, this work brings in the fifth chapter a real example about application of the methodology. It is possible to observe and monitor all phases of the Scrum's implementation, since the initial meeting until the retrospective.

Through this content, the reader will be able to acquire necessary knowledge to adapt the methodology to their daily works and embrace it as well, obtaining very satisfactory results, like the company presented in the fifth chapter.

Key-words: Scrum, Software Quality

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

Independente do produto ou serviço que uma empresa oferece a preocupação com a qualidade deixou de ser um diferencial competitivo e passou a ser pré-requisito básico para participar do jogo do mercado.

Cada vez mais os clientes/consumidores têm a consciência de seu direito de exigir um produto/serviço testado, aprovado, seguro, que funcione completamente.

De acordo com a *International Organization for Standardization* (ISO) “Qualidade é a adequação ao uso. É a conformidade às exigências”. Não é algo que pode ser definido de forma única, pois a qualidade para uma pessoa pode não ser a mesma para a outra. A definição do termo depende da necessidade e/ou satisfação de cada cliente.

Se o produto for um carro, qualidade estará relacionada a conforto, segurança, desempenho, beleza e custo. O foco deste trabalho é a qualidade de software, portanto qualidade estará relacionada ao prazo de entrega, ao atendimento das solicitações, ao funcionamento adequado do software, à usabilidade do mesmo, entre outros fatores.

Neste trabalho, serão apresentados alguns procedimentos e metodologias que ajudam a garantir a qualidade para o desenvolvimento de softwares, que reflete diretamente na satisfação do cliente.

1.1. JUSTIFICATIVAS E MOTIVAÇÃO

É extremamente complicado tratar de qualidade no desenvolvimento de softwares, mesmo existindo várias teorias e modelos a serem seguidos. A realidade de muitos desenvolvedores não permite que estes sejam postos em prática.

Existe uma pressão no desenvolvimento dos projetos que pode ser chamada de “curto prazo”. Na realidade é necessário atribuir pequenos prazos no atendimento das necessidades dos usuários, que muitas vezes não são cumpridos. A alta demanda de tarefas a serem desenvolvidas sempre chega junto com o pequeno espaço de tempo para desenvolvê-las.

Neste trabalho será apresentada uma possível solução, o modelo de qualidade de software *Scrum*, usado para gerenciamento e desenvolvimento de software tão ágil que o tempo dispensado para a organização das exigências do mesmo é facilmente compensado ao se aplicar suas metodologias.

Qualidade cada vez mais é essencial, independente do produto ou serviço. Tal assunto precisa ser sempre enriquecido com novas experiências relatadas.

Para TI, qualidade entra no histórico principalmente a partir de 1970, com a engenharia de software. Portanto, principalmente por ser um assunto recente, é necessário que sejam disponibilizados mais trabalhos acadêmicos que possam estar ao alcance de todos os envolvidos com desenvolvimento e avaliação de softwares (tanto empresários, quanto desenvolvedores e usuários).

Este modelo que será apresentado mais detalhadamente, o *Scrum*, está em constante renovação, pois cada empresa adapta as metodologias à sua realidade, e cada vez mais surgem idéias para complementar o modelo, que precisam ser disponibilizados ao meio, contribuindo desta forma com todos os interessados.

1.2. OBJETIVOS

O método *Scrum* permite que profissionais possam desenvolver seus projetos de forma ágil e com a qualidade necessária. Através das informações levantadas nessa pesquisa profissionais de Tecnologia da Informação (TI) poderão conhecer as diferenças entre alguns modelos e implantar em suas empresas e no seu dia-a-dia os modelos de qualidade de software. Poderão adquirir mais conhecimento sobre os mesmos, especialmente o *Scrum*.

Através de um projeto real de uma empresa de software, demonstraremos todas as fases de desenvolvimento do mesmo utilizando o *Scrum*. Serão apresentados de forma detalhada todos os passos para implantação do método. Desta forma será defendida a teoria de que o mesmo pode ser implantado no desenvolvimento de projetos, independente da realidade de cada empresa.

Além dos envolvidos com a parte de desenvolvimento dos softwares (desenvolvedores, gerentes, empresários, entre outros), esse trabalho trará a

possibilidade de transmitir aos usuários finais o conhecimento necessário para acompanhar o andamento do projeto solicitado.

1.3. ESTRUTURA DO TRABALHO

O Projeto final seguirá uma estrutura dividida em capítulos, conforme apresentado a seguir.

Conforme acompanhado até o momento, o primeiro capítulo é a Introdução da pesquisa, este possui três seções: justificativas e motivação, objetivos e este relatando a estrutura do trabalho.

O Segundo capitulo é dividido em duas seções, sendo a primeira um breve relato histórico da qualidade, definindo-a e mostrando que a mesma é cada vez mais importante no mercado e a segunda destinada especificamente à qualidade de Software.

O próximo capítulo, o terceiro, apresenta alguns modelos de qualidade de software: a primeira seção é destinada ao modelo CMM, este possui uma subseção que trata de sua evolução, o CMMI. Posteriormente, na seção 3.2 apresenta o modelo PSP, e por fim a última seção do terceiro capítulo traz o MPS.BR.

O quarto capítulo é dedicado ao Modelo de Qualidade de Software *Scrum*. Nele é apresentado desde o histórico deste modelo, até os resultados que grandes empresas vem tendo com a implantação do mesmo. Este capítulo é composto por várias seções, seqüencialmente: conceitos básicos, Product Owner, Scrum Master, o time, Product Backlog, reunião de planejamento do sprint, tamanho do sprint, gráfico de burndown, reunião diária, revisão e por fim retrospectiva.

Como o quarto capítulo é composto por muitas definições, foi dedicado um capítulo com o histórico de implantação do Scrum em uma empresa, mostrando detalhadamente todos os passos vivenciados pelos participantes do projeto.

Encerra-se o trabalho com um breve capítulo de conclusão da pesquisa e perspectivas futuras.

2. QUALIDADE DE SOFTWARE

Embora o controle de qualidade e o uso de padrões como ISO sejam algo que tenha atraído bastante atenção nas últimas décadas, historicamente o assunto é muito antigo.

As próximas seções apresentarão os conceitos de Qualidade e Qualidade de Software.

2.1 QUALIDADE

Pode-se definir como um provável inicio da história da qualidade os relatos existentes há mais de quatro mil anos dos egípcios; eles estabeleceram um padrão de medida de comprimento: o cúbito (tratava-se da medida do comprimento do braço do faraó reinante), ou seja, se trocasse o faraó, a medida deveria ser atualizada. O cúbito deveria ser utilizado como uma medida padrão, para cálculos, construções, etc. Os responsáveis pelas construções tinham de conferir as medidas a cada lua cheia, caso a mesma tivesse divergência, o responsável pela falha precisava ser punido [Juran e Gryna, 1988]. O resultado de tal precisão são as brilhantes pirâmides egípcias.

Posterior aos Egípcios prosseguiu o histórico de qualidade com os templos da Grécia e ainda mais, as catedrais medievais da Roma (século XVI). Em todas essas realizações foram utilizados instrumentos para precisão nas medidas criadas pelos povos nas respectivas épocas.

Um grande marco na história da qualidade foi, com certeza, a revolução industrial. Esse período também é associado a profundas mudanças econômicas e sociais, como o início da automação e o surgimento do consumo de massa. Durante essa fase, a criação de diversas indústrias levou rapidamente à concorrência entre elas, o que, por sua vez, desencadeou um processo de melhoria contínua que perdura até hoje.

Segundo Koscianski (2007), na década de 1920 surgiu o controle estatístico de produção. Nas fábricas que produziam grande quantidade de itens tornou-se impossível garantir a qualidade individual de cada peça, ao contrário do que se fazia

(e ainda se faz) no trabalho artesanal. Dessa forma foi preciso desenvolver mecanismos diferentes e a resposta veio da estatística. Um dos primeiros trabalhos associados ao assunto é o livro publicado por Walter Shewhart em 1931, *Economic Control of Quality of Manufactured Product*. Shewhart, dos Bell Laboratories, teria introduzido os diagramas de controle (*control charts* ou *Shewhart chart*).

Na década de 1940 o Japão destacou-se como um importante pólo no assunto e contribuiu com diversas novas ferramentas: o método de Taguchi para projeto experimental, e posteriormente (década de 1950) a metodologia 5S ou, ainda, os diagramas de causa e efeito de Ishikawa, também conhecidos como diagramas espinha de peixe.

Ainda na década de 1940, surgiram vários organismos ligados à qualidade; por exemplo, a ASQC (*American Society for Quality Control*), a ABNT (Associação Brasileira de Normas Técnicas) e, ainda, a ISO (*International Organization for Standardization*). A Segunda Guerra Mundial também contribuiu com o processo, quando as técnicas de manufatura foram aprimoradas para fabricação de material bélico.

Segundo Joseph Juan (1997*):*

> Um segundo efeito residual dos cursos de treinamento da Segunda Guerra Mundial foi a criação da Sociedade Americana de Controle de Qualidade (ASQC, na sigla em inglês). A história começou com um congresso de duas semanas, no qual pessoas que trabalhavam com controle de qualidade em sua empresa puderam trocar experiências. A partir de então, eles criaram sociedades de controle da qualidade em sua região e a união dessas sociedades deu origem à ASQC.

Provavelmente a primeira vez em que se utilizou o termo “Engenharia de Software” foi em uma conferência com esse nome, realizada em 1968, na Alemanha. A conferência foi realizada por uma entidade que, a rigor, não possuía nenhuma ligação com a área: o Comitê de Ciência da OTAN (Organização do Tratado do Atlântico Norte). E então, por volta de 1970 inicia-se a discussão sobre qualidade de Software, tratada pela Engenharia de software.

2.2 QUALIDADE DE SOFTWARE

Entende-se por Qualidade de Software o conjunto de características a serem satisfeitas em um determinado grau, de modo que o software satisfaça às necessidades de seus usuários.

O relatório da conferência da OTAN (1968) e outros documentos da década de 1970 provam que os problemas daquela época são os mesmo enfrentados atualmente. Entre eles: cronogramas não observados; projetos com tantas dificuldades que são abandonados; módulos que não operam corretamente quando combinados; programas que não fazem exatamente o que era esperado; programas tão difíceis de usar que são descartados e programas que simplesmente param de funcionar.

Alguns clientes continuam insatisfeitos com seus softwares devido aos “antigos-atuais” problemas relatados. Para ajudar a combater esses problemas, surgem os modelos de Qualidade de Software. O próximo capítulo tem o objetivo de discutir com maiores detalhes o CMM (*Capability Maturity Model*)/CMMI (*Capability Maturity Model Integration*), o PSP (*Personal Software Process*) e o MPS.BR. Em seguira, o trabalho tratará mais detalhadamente o método Scrum.

O Scrum é um método ágil de desenvolvimento de software aplicável a ambientes voláteis. Este método aplica maior ênfase ao gerenciamento de projetos em relação aos demais, além de reunir atividades como monitoramento, feedback, reuniões rápidas e diárias com toda equipe, visando identificação e correção de possíveis falhas.

3. MODELOS DE QUALIDADE DE SOFTWARE

Os modelos de qualidade de software foram criados para organizar, avaliar, auxiliar e prover melhorias no desenvolvimento dos softwares.

Em seguida traremos mais informações sobre o CMM (*Capability Maturity Model*)/CMMI (*Capability Maturity Model Integration*), o PSP (*Personal Software Process*), o MPS.BR e o *Scrum*.

3.1 CMM

O *Capability Maturity Model* (CMM), como o próprio nome diz, é um Modelo para avaliação da maturidade dos processos de software de uma organização. Este identifica as principais práticas que são necessárias para aumentar a maturidade desses processos.

O CMM prevê cinco níveis de maturidade: inicial, repetível, definido, gerenciado e otimizado.

Estágio	Foco	Principais Áreas do Processo	Resultado
5 Otimizado	Processo que será constantemente melhorado	- Prevenção de defeitos - Gestão de alterações tecnológicas - Gestão de alterações do processo	Produtividade e Qualidade
4 Gerenciado	Processo e Produto Medido	- Gestão quantitativa do processo - Gestão da qualidade do software	
3 Definido	Processo definido e institucionalizado	- Organização do processo - Definição do processo - Formação - Gestão integrada com o software - Engenharia do software - Coordenação inter-grupos - Revisões (testes)	
2 Repetível	Processo dependente de Indivíduos	- Gestão de requisitos - Planejamento de projetos - Acompanhamento e inspeção de projetos - Gestão da subcontratação - Gestão de configurações - Verificação da qualidade do software	Risco
1 Inicial	Processo Caótico		

Tabela 1 – Estrutura do Modelo CMM

Cada um dos níveis da Tabela 1 pode ser detalhado da seguinte forma:

“1º Estágio – Inicial: neste nível o processo de software é caracterizado como “ad hoc” e até mesmo ocasionalmente caótico. Neste, poucos processos são definidos e o sucesso depende de esforço individual dos membros da equipe.

2º Estágio - Repetível: Os processos básicos de gestão de projeto são estabelecidos para acompanhar custo, cronograma e funcionalidade. Existe a necessária disciplina do processo para repetir sucessos anteriores em projetos com aplicações similares.

3º Estágio - Definido: neste terceiro nível, o processo de software para as atividades de gestão e engenharia é documentado, padronizado e integrado em um processo de software padrão para a organização. Todos os projetos utilizam uma versão aprovada do processo de software padrão para desenvolver e manter software.

4º Estágio - Gerenciado: É neste nível que as medidas detalhadas do processo de software e da qualidade do produto são realizadas. O processo e os produtos de software são quantitativamente compreendidos e controlados.

5º Estágio - Otimizado: A melhoria contínua do processo é propiciada pelo feedback quantitativo do processo e pelas idéias e tecnologias inovadoras.” Fernandes (Análise de Sistema Orientada ao Sucesso, 2005).

A especificação do modelo compreende um instrumental para a averiguação do nível atual da organização e expõe os pontos de ação, ou áreas chave de processo, que servem como pontos de destaque para a promoção a um nível mais elevado de maturidade.

A adoção do CMM se baseia em um processo gradual de aumento da maturidade da capacidade de desenvolvimento de software de uma organização. A maturidade desse processo pode ser traduzida como a probabilidade de entregar sistemas de software dentro do prazo, utilizando os recursos planejados e atendendo aos requerimentos para o qual foi projetado dentro dos padrões de qualidade desejados.

Este modelo de maturidade fornece às organizações de software um guia de como obter controle em seus processos para desenvolver e manter software e como evoluir em direção a uma cultura de engenharia de software e excelência de gestão. Porém ele não garante que os produtos de software serão concretizados com sucesso ou que todos os problemas de desenvolvimento de software serão

resolvidos. Ele identifica práticas para um processo de software maduro e as características de um processo de software efetivo, mas a organização madura trata de todas as questões essenciais para um projeto bem sucedido, incluindo pessoas e tecnologia, tanto quanto processos.

3.1.2 CMMI

O CMMI organiza as práticas, que já são consideradas efetivas, em uma estrutura que visa auxiliar a organização a estabelecer prioridades para melhoria e também fornece um guia para a implementação dessas melhorias.

Assim como o CMM, o CMMI está dividido em cinco estágios:

1 Realização – Estágio inicial;

2 Gerenciado – Gerenciamento de requisitos, planejamento de projeto, monitoramento e controle de projeto, gerenciamento de fornecedores, medição e análise, garantia da qualidade do processo e do produto, gerenciamento de configuração;

3 Definido – Desenvolvimento de requisitos, solução técnica, integração do produto, verificação e validação, foco no processo organizacional, definição do processo organizacional, treinamento organizacional, gerenciamento de riscos, gerenciamento integrado do projeto, análise da decisão e resolução;

4 Quantitativamente – Gerenciamento quantitativo do projeto, performance do processo organizacional;

5 Otimização – Análise causal e resolução, inovação organizacional e implantação.

A Metodologia CMMI divide cada estágio em áreas de processo e para cada uma delas são definidos dois conjuntos de metas: as específicas e as genéricas.

A essas metas, a definição do modelo recomenda práticas genéricas divididas em um conjunto de características comuns que por sua vez se divide em quatro categorias. São elas:

- **Comprometimento com a execução** – Agrupa práticas relacionadas à definição de políticas e responsabilidades, descrevendo ações para assegurar que o processo se estabeleça e seja duradouro;
- **Habilitação para execução** – Agrupa práticas contendo pré-condições para o projeto, de forma a permitir a implementação adequada do processo;
- **Direcionamento a implementação** – Agrupa práticas relacionadas ao gerenciamento do desempenho do processo;
- **Verificação da implementação** – Agrupa práticas para revisão junto à alta gerência e avaliação objetiva da conformidade com processos, procedimentos e padrões.

É necessário à Empresa focar seus esforços na definição das metas específicas/genéricas para a realização do trabalho.

As metas específicas, na maioria das vezes, estão focadas no negócio da empresa e buscam alinhar a metodologia CMMI às necessidades próprias; por sua vez as metas comuns focam em aspectos inerentes a qualquer empresa e devem ser considerados para a correta implementação da metodologia, de forma a garantir a maximização dos resultados.

As categorias acima descritas deverão ser consideradas em qualquer estágio com o qual a empresa se identifique dentro da metodologia exposta. Elas buscam direcionar as ações de forma a garantir que o ciclo de evolução seja completado, possibilitando a implementação de uma evolução contínua dos processos e do produto como um todo.

A metodologia CMMI foi criada pela SEI (Software Engineering Institute) para ser um guia destinado a melhorar os processos organizacionais e a habilidade desses em gerenciar o desenvolvimento, a aquisição e a manutenção de produtos e serviços.

3.2 PSP

O PSP (*Personal Software Process*) foi desenvolvido por Watts Humphrey, com o objetivo de auxiliar desenvolvedores e pequenas equipes de desenvolvimento em suas necessidades de melhoria.

O autor utilizou como base o modelo SW-CMM (*Capability Maturity Model for Software*), baseado no princípio da melhoria do processo. O que diferencia um modelo do outro é que o SW-CMM é focado na melhoria da capacidade organizacional, enquanto que o foco do PSP é o indivíduo, auxiliando-o a promover melhorias em seu processo de desenvolvimento.

Dessa forma, o profissional produzirá com mais qualidade, dentro de prazos realisticamente estipulados e com custos menores.

Segundo Rebelo (2010), o PSP vem ao encontro da necessidade dos desenvolvedores de cumprir as exigências do mercado, que pede profissionais que saibam planejar o seu trabalho e realizá-lo conforme o planejado. Ele tem como principais objetivos melhorar o planejamento e o acompanhamento de cronogramas, criar um comprometimento pessoal com a qualidade e um envolvimento constante do desenvolvedor na melhoria contínua do processo.

O modelo de qualidade está estruturado em formulários que funcionam como um guia para a coleta e a totalização dos dados. Existem formulários específicos para cada parte do projeto: planejamento, desenvolvimento e relatórios de acompanhamento. Os formulários auxiliam o desenvolvedor a gerenciar de maneira mais efetiva o seu tempo e suas atividades e, também, evidenciam ao desenvolvedor onde o mesmo está cometendo mais erros e se o seu planejamento foi preciso.

3.2.1 OS NÍVEIS DE MATURIDADE DO PSP

O PSP foi criado para ser utilizado de acordo com sete níveis de maturidade, nos quais, gradativamente são introduzidas novas práticas de Engenharia de Software. Estes níveis estão agrupados dois a dois, com base no seu foco.

Nível	Nome	Atividades
PSP0 PSP0.1	Medição Pessoal	Registro de tempo Registro de defeitos Padrão de tipos de defeitos Padrão de codificação Medida de tamanho Proposta de melhoramento do processo
PSP 1 PSP 1.1	Planejamento Pessoal	Estimativa de tamanho Relatório de testes Planejamento de tarefas Cronogramas
PSP 2 PSP 2.1	Qualidade Pessoal	Revisões de código Revisões de projeto Padrões de projeto
PSP 3	Processo Cíclico Pessoal	Desenvolvimento cíclico

Figura 1 – Níveis do PSP

Segundo Rebelo (2010), cada nível pode ser explicado da seguinte forma:

- PSP0 e PSP0.1 - *The Baseline Process*

O primeiro grupo está ligado às melhorias individuais e é denominado *Baseline Process*. Ele compreende os níveis PSP0 e PSP0.1, que constituem a base para se obter dados sobre os processos atuais. Esses níveis pretendem proporcionar a melhoria individual através de métricas e relatórios padronizados. Tempo, defeito e tamanho são as métricas básicas com as quais o PSP trabalha. As duas primeiras são introduzidas no nível PSP0 e a terceira no PSP0.1. As demais, que são combinações derivadas destas, serão introduzidas nos níveis posteriores.

- PSP1 e PSP1.1 - *The Personal Planning Process*

Este segundo grupo, o *Personal Planning Process*, procura ensinar o desenvolvedor a planejar melhor seus projetos. Nele estão compreendidos os níveis PSP1 e PSP1.1. Estes níveis estão focados no auxílio ao desenvolvedor, para que este

melhore seu perfil de planejamento pessoal. O PSP1 dedica-se a estabelecer uma forma de estimar o tamanho da atividade, além de se preocupar em introduzir um formato padrão para registrar os dados de teste. Já o PSP1.1 vem acompanhado de um formulário para planejar as tarefas e outro para prever e acompanhar o cronograma de execução. Seu foco é efetuar o planejamento propriamente dito. Para identificar antecipadamente possíveis atrasos, o PSP se preocupa em auxiliar o desenvolvedor a acompanhar de forma mais precisa o progresso de seu projeto. Para isso, utiliza o conceito de valor agregado, do original em inglês: *earned value*.

- PSP2 e PSP2.1 - *Personal Quality Management*

Focado nas questões de qualidade de desenvolvimento pessoal, o *Personal Quality Management*, é o terceiro grupo de níveis: PSP2 e PSP2.1. Esses dois níveis tratam do gerenciamento da qualidade pessoal. No PSP2, são introduzidas as revisões de código e projeto ao processo do desenvolvedor. Tanto na revisão de código quanto na revisão de projeto, deve-se estabelecer objetivos e seguir um processo de revisão definido, coletando métricas, de modo a poder avaliar o processo utilizado. As revisões de projeto no PSP são auxiliadas pelo formulário *Design Review Checklist*. Ele orienta a revisar projetos de desenvolvimento de programas e deve ser alterado dependendo da linguagem de programação utilizada. Nas revisões de código, o formulário utilizado é o *Code Review Checklist*, no qual o desenvolvedor deve coletar diferentes dados para as linguagens de programação utilizadas (caso utilize mais de uma).

O nível PSP2.1 auxilia o desenvolvedor a reduzir o número de defeitos da fase de projeto do programa, melhorando sua qualidade. Para orientar o desenvolvedor, são utilizados quatro templates, que são baseados em métodos orientados a objeto. Caso a empresa não utilize estes métodos, o desenvolvedor poderá fazer as adaptações necessárias.

- PSP3 - *Cyclic Personal Process*

Por último, o *Cyclic Personal Process* é oferecido com o objetivo de escalonar projetos maiores, e seu nível é o PSP3. Ele é o último nível do PSP e auxilia o desenvolvedor a lidar com programas maiores. Para isso, divide-se o projeto em

partes menores e aplica-se o processo PSP2.1 para cada uma delas. Esses pequenos projetos ficam mais gerenciáveis para o desenvolvedor do que o todo.

Através da utilização do modelo PSP, o profissional organiza suas atividades de maneira fácil e objetiva, gerenciando suas tarefas de forma organizada e com planejamentos precisos, atingindo metas e entregando produtos no prazo estipulado.

3.3 MPS.BR

O MPS.BR (Melhoria de Processos do Software Brasileiro), é um movimento para a melhoria e também um modelo de qualidade de processo voltada para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil.

A Melhoria de Processos do Software Brasileiro tem como base o CMMI, as normas ISO/IEC 12207 e ISO/IEC 15504 e a realidade do mercado brasileiro.

A coordenação do Programa MPS.BR conta com duas estruturas de apoio para o desenvolvimento de suas atividades, o Fórum de Credenciamento e Controle (FCC) e a Equipe Técnica do Modelo (ETM). Através destas estruturas, o MPS.BR obtém a participação de representantes de Universidades, Instituições Governamentais, Centros de Pesquisa e de organizações privadas, os quais contribuem com suas visões complementares que agregam qualidade ao empreendimento.

No Brasil, uma das principais vantagens do modelo é seu custo reduzido de certificação em relação as normas estrangeiras, sendo ideal para micro, pequenas e médias empresas.

Segundo Mesquita (2008), um dos objetivos do projeto é replicar o modelo na América Latina, incluindo o Chile, Argentina, Costa Rica, Peru e Uruguai.

O projeto tem apoio do Ministério da Ciência e Tecnologia, do FINEP e do Banco Interamericano de Desenvolvimento. No Brasil o projeto é desenvolvido pelo Softex, pelo governo e por universidades.

O programa mobilizador MPS.BR está dividido em três (3) componentes: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-

MPS). Cada componente é descrito por meio de Guias e/ou de Documentos do MPS.BR.

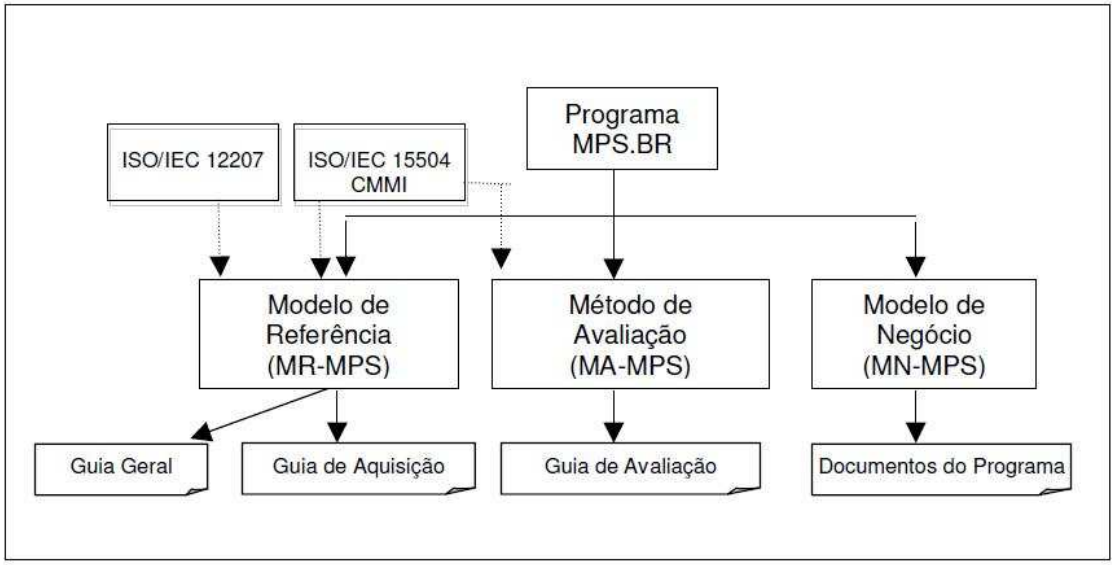

Figura 2 – Componentes do Programa MPS.BR

O Modelo de Referência MR-MPS contém os requisitos que os processos das unidades organizacionais devem atender para estar em conformidade com o MR-MPS.

O Guia de Avaliação contém o processo e o método de avaliação MA-MPS, os requisitos para os avaliadores líderes, avaliadores adjuntos e Instituições Avaliadoras (IA). O processo e o método de avaliação MA-MPS está em conformidade com a norma ISO/IEC 15504-2 [ISO/IEC 15504-2, 2003].

O Modelo de Negócio MN-MPS descreve regras de negócio para implementação do MR-MPS pelas Instituições Implementadoras (II), avaliação seguindo o MA-MPS pelas Instituições Avaliadoras (IA), organização de grupos de empresas para implementação do MR-MPS e avaliação MA-MPS pelas Instituições Organizadoras de Grupos de Empresas (IOGE), certificação de consultores de aquisição e programas anuais de treinamento por meio de cursos, provas e workshops MPS.BR.

3.3.1 NÍVEIS DE MATURIDADE DO MPS.BR

Os níveis de maturidade estabelecem patamares de evolução de processos,caracterizando estágios de melhoria da implementação de processos na organização.

O MR-MPS define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado).

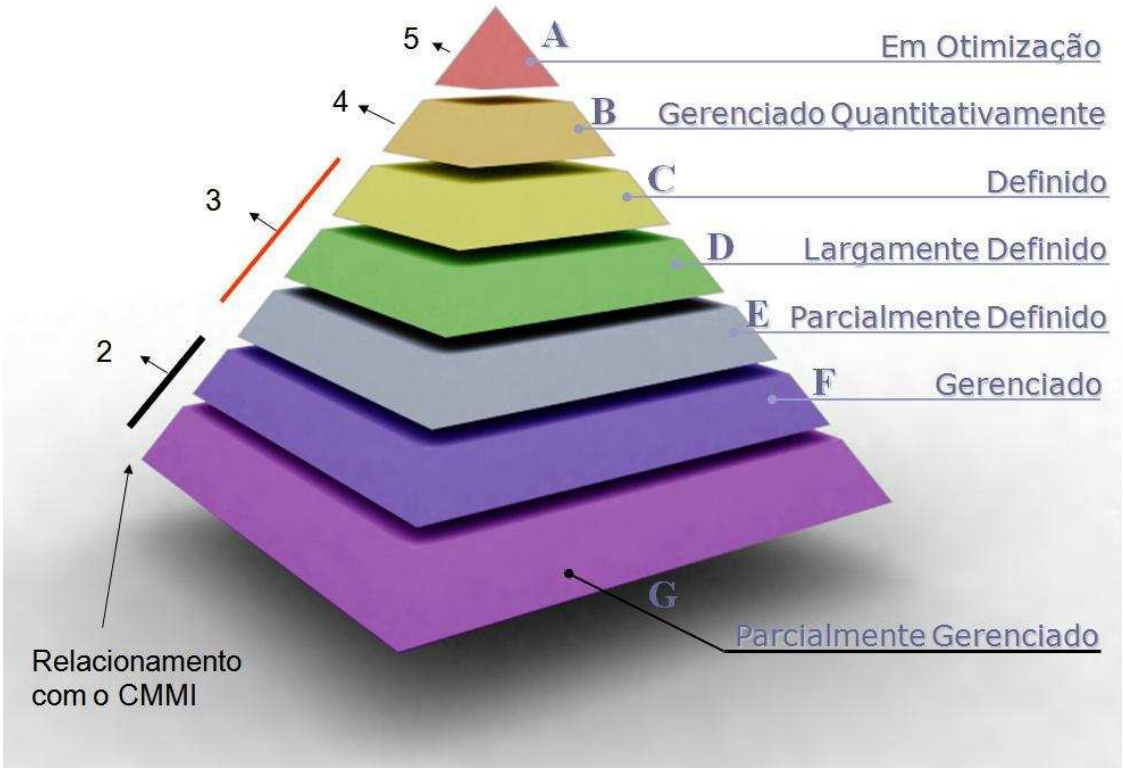

Figura 3 – Níveis da Maturidade do MPS.BR

O progresso e o alcance de um determinado nível de maturidade MPS se obtém quando são atendidos os propósitos e todos os resultados esperados dos respectivos processos e dos atributos de processo estabelecidos para aquele nível.

A divisão em estágios, embora baseada nos níveis de maturidade do CMMI-SE/SW tem uma graduação diferente, com o objetivo de possibilitar uma implementação e avaliação mais adequada às micros, pequenas e médias empresas.

3.3.2 OS PROCESSOS DO MR-MPS

Os processos são agrupados, por uma questão de organização, de acordo com a sua natureza, ou seja, o seu objetivo principal no ciclo de vida de software. Esse agrupamento resultou em três (3) classes de processos, que são:

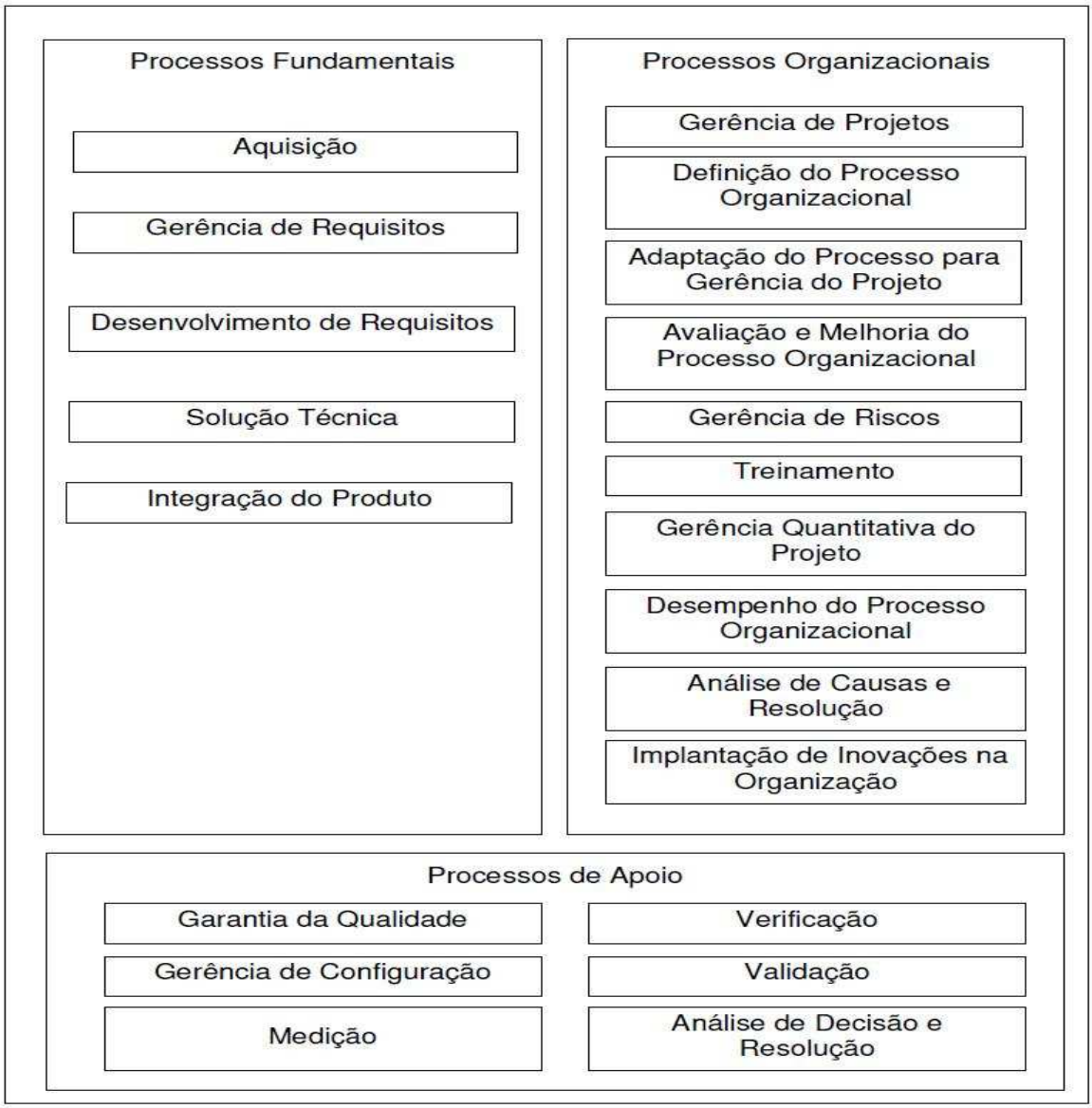

Figura 4 - Os Processos do MR-MPS

- Processos fundamentais - atendem o início e a execução do desenvolvimento, operação ou manutenção dos produtos de software e serviços correlatos durante o ciclo de vida de software;

- Processos de apoio - auxiliam outro processo e contribuem para o sucesso e qualidade do projeto de software;

- Processos organizacionais - uma organização pode empregar estes processos em nível corporativo para estabelecer, implementar e melhorar um processo do ciclo de vida.

3.3.2.1 CAPACIDADE DO PROCESSO

A capacidade do processo é representada por um conjunto de atributos de processo descrito em termos de resultados esperados. A capacidade do processo expressa o grau de refinamento e institucionalização com que o processo é executado na organização.

O atendimento aos atributos do processo (AP), através do atendimento aos resultados esperados dos atributos do processo (RAP) é requerido para todos os processos no nível correspondente ao nível de maturidade, embora eles não sejam detalhados dentro de cada processo.

Os níveis são acumulativos, ou seja, se a organização está no nível F, esta possui o nível de capacidade do nível F que inclui os atributos de processo dos níveis G e F para todos os processos relacionados no nível de maturidade F (que também inclui os processos de nível G). Isto significa que, ao passar do nível G para o nível F,os processos do nível de maturidade G passam a ser executados no nível de capacidade correspondente ao nível F.

O MPS.BR possui 5 AP (Atributos de Processo):

* AP 1.1 (o processo é executado),

* AP 2.1 (o processo é gerenciado),

* AP 2.2 (os produtos de trabalho do processo são gerenciados).

* AP 3.1 (o processo é definido).

* AP 3.2 (o processo está implementado).

4. *SCRUM*

O *Scrum* foi criado inicialmente como um framework para gerenciamento de projetos na indústria convencional, em 1995.

Seu nome é originado de um momento da partida de *Rugby* onde o time trabalha junto em prol da posse de bola, ou seja, em prol da mesma causa.

Ken Schwaber formalizou o *Scrum* para projetos de desenvolvimento de software. *Scrum* foi fortemente baseado no processo Lean da Toyota.

4.1 CONCEITOS BÁSICOS

O *Scrum* funciona basicamente a partir de uma lista de funcionalidades a serem desenvolvidas (*Product Backlog*). O dono do produto/projeto é chamado de *Product Owner*, é ele quem prioriza o *Product Backlog*. O líder do projeto (*Scrum Master*) escolhe algumas funcionalidades (*Product Backlog*) que julga possível desenvolver no ciclo do *Sprint*, este é baseado em ciclos de 2 a 4 semanas (podendo ser alterado de acordo com cada necessidade). A cada *Sprint* o esforço é para se entregar itens com o maior valor de negócio (e prioridade) ao *Product Owner*, dando a ele algo real e de valor para o negócio.

4.2 *PRODUCT OWNER*

O *Product Owner* representa o cliente, que conhece o negócio e as regras do funcionamento dele, se tem alguém no projeto que tem todo interesse pelo ROI *(Return of investment)* tem que ser o *Product Owner*.

O *Product Owner* é responsável por criar o *Product Backlog* e priorizá-lo. Como ele é quem sabe o que é mais importante para o negócio, ele também fará as alterações dos itens, seja prioridade ou remoção e adição de novos.

Em algumas empresas esse papel é conhecido como Gerente de Produto, e também outros nomes semelhantes.

Esse papel gera muitas discussões entre os agilitas mundo à fora, o que se deve ter em mente não é se esse papel será composto por um representante do cliente dentro da equipe ou por uma pessoa da própria empresa prestadora do serviço, o importante é que seja uma pessoa capacitada a sanar as dúvidas que surjam no dia-a-dia, pois ele faz parte da equipe também e deve ter todo o comprometimento do mundo com o projeto. O que acontece muito é colocarem pessoas que se acham o *Gerente do Projeto*, ou que não tem conhecimento do real papel do *Product Owner* dentro do *Scrum* e essas pessoas acabam dificultando muitos mais o trabalho do que ajudando.

4.3 *SCRUM MASTER*

O *Scrum Master* é o papel responsável por fazer o ambiente *Scrum* funcionar, verificando se o time esta respeitando e cumprindo os valores e práticas do *Scrum*.

Ele também orienta o *time* no *Daily Meeting* corretamente e é o responsável por remover todos os impedimentos apontados.

Ele protege a equipe de interferências externas, assegura que os *Sprints* não contenham itens além do que pode ser realmente entregue. Em alguns lugares se tem a visão de que o *Scrum Master* é um Gerente de Projetos, e não é! O *Scrum Master* é um facilitador, alguém que tem a missão de fazer o time funcionar e aplicar corretamente o Scrum.

4.4 O TIME (*TEAM*)

O *Time* é responsável por transformar itens do *Product Backlog* em itens do *Sprint Backlog* e transformar esses itens em software pronto para ser entregue.

As equipes de *Scrum* contém geralmente entre 5 e 9 pessoas, não mais do que 10, isso é essencial para a boa prática do *Scrum*. Os membros são multifuncionais, podendo conter desenvolvedores, designers, arquitetos da informação, etc.

Outra característica importante é que os times são auto-gerenciáveis, sendo eles responsáveis por controlar as tarefas do desenvolvimento do *Sprint*.

4.5 *PRODUCT BACKLOG*

Product Backlog é a lista de tudo que se deseja que seja desenvolvido para o software (estórias, requisitos, etc), de uma maneira enxuta, sem detalhamento, utilizando a terminologia do Cliente.

O grande diferencial é que essa lista não precisa estar completa logo no início, não precisa ter 100% dos itens possíveis e imagináveis. O *Product Owner* define essa lista, que pode ir ganhando outros itens ao decorrer do desenvolvimento das *Sprints*, ele também detalha os itens em cada planejamento de *sprint*, e prioriza os itens; a equipe define quais itens cabem ou não dentro da Sprint, essa lista gerada tem o nome de *Sprint Backlog*. O processo se repete a cada ciclo de desenvolvimento.

Podem ser criadas planilhas, ou outros documentos de preferência, para controle dos *Product Backlogs*. O ideal é que esse documento contenha uma identificação única (como um ID), um nome curto e descritivo para a estória (o item), uma pontuação para relevar o grau de importância (mais pontos, mais importante), uma estimativa inicial (multiplicação entre o número de pessoas e o número de dias que finalizariam o sprint se tivessem 100% do tempo para realizá-lo), e uma descrição de como demonstrar o item. A tabela 2 apresenta um exemplo:

PRODUCT BACKLOG					
ID	Nome	Importância	Estimativa	Como Demonstrar	
1	Depósito	30	5	Logar-se, abrir a página de depósito, depositar R$10,00, ir para página menu saldo e verificar se aumentou R410,00.	Precisa de um diagrama UML de sequência. Não é necessário se preocupar com o criptograma por enquanto.
2	Verificar seu próprio histórico de transações	10	8	Logar-se, clicar em “transações”. Fazer um depósito. Voltar para transações, verificar se o novo depósito é listado.	Usar paginação para evitar consultas muito grandes ao banco e dados. Projetarde forma similar à pagina de visualização do susuários.

Tabela 2 – Modelo de *Product Backlog*

4.6 REUNIÃO DE PLANEJAMENTO DO *SPRINT*

O propósito de uma reunião de planejamento do *Sprint* é dar à equipe informações suficiente para trabalharem tranquilamente por algumas semanas, e dar ao *product owner* confiança suficiente para deixar a equipe trabalhar.

Este encontro dele visa definir um objetivo do *Sprint*, uma lista de membros da equipe (e seus níveis de comprometimento, se não 100%), um *sprint Backlog* (lista de estórias inclusas no *sprint*), uma data definida para apresentação do *sprint* e por fim, data e local para a reunião diária.

O ideal é que o *Product owner* participe dessa reunião, pois é ele quem define o Escopo e a Importância do *sprint*, enquanto a equipe de desenvolvimento define a estimativa. Essas três variáveis serão fundamentais para o cumprimento do *sprint* no prazo desejado.

4.6.1 TAMANHO DO SPRINT

Uma das tarefas mais difíceis no *Scrum* é definir o tamanho do *Sprint.*

Sprints curtos são bons, eles permitem que a equipe seja “ágil”, ou seja, mude de direção frequentemente. *Sprints* curtos tem um ciclo curto de feedback (entregas mais freqüentes ao cliente), e ainda *sprints* curtos faz com que menos tempo seja perdido indo em direção errada (aprender e melhorar rápido).

Entretanto, *sprints* longos são bons também. A equipe tem mais tempo para ganhar ritmo, ela tem mais espaço para se recuperar dos problemas, e conseguir atingir o objetivo do *sprint*.

No geral, *Product owners* gostam de *sprints* curtos e desenvolvedores preferem os longos. O tamanho dos *sprints* representa um compromisso, entre o que satisfaça o *product owner* e o desenvolvedor. O ideal é que sejam feitos testes para ver o desempenho das equipes, com sprints de diferentes tamanhos.

Segundo Henrik Kniberg (Scrum e XP direto das Trincheiras,12/2008, p.31): “a maioria das nossas equipes (mas não todas) faz *sprints* de 3 semanas. Curtas o

bastante para nos dar agilidade corporativa adequada, longas o bastante para a equipe conseguir fluidez e se recuperar de eventuais problemas durante o *sprint*".

4.6.2 GRÁFICO DE *BURNDOWN*

O gráfico de *Burndown* apresenta o trabalho cumulativo restante em uma *Sprint*, atualizado diariamente.

Após o *Planning Meeting*, é feita a totalização do *Sprint Backlog* para montar o gráfico de *Burndown*. A cada dia de trabalho, o time faz a somatória do que foi produzido e subtrai do total no gráfico de *Burndown*. Caso alguma tarefa precise ser reestimada, o tempo adicional também deve ser adicionado ao gráfico.

4.7 REUNIÃO DIÁRIA (*SCRUM DAILY MEETING*)

A reunião diária serve para a equipe se alinhar em relação ao desenvolvimento dos itens do *Sprint Backlog*. Esta reunião deve durar no máximo 15 minutos, seu conteúdo é exposto pela equipe, basicamente respondendo 3 perguntas:

- O que eu fiz ontem?
- O que farei hoje?
- Quais impedimentos eu tive?

Essas são as perguntas que compõem o conteúdo da reunião, que é voltada para o time; cada membro dirige suas respostas para o time todo e não direcionada para o *Scrum Master*. Não é uma forma de cobrança vindo de um gerente de projetos, é a maneira onde toda a equipe se sincroniza em relação às tarefas e relatam os impedimentos que possam estar interferindo no bom andamento do *Sprint*.

4.7.1 REVISÃO (*SPRINT REVIEW*)

Está é uma reunião muito importante do ponto de vista do cliente, nela, tudo o que foi desenvolvido durante o *Sprint* será apresentada para os responsáveis pelo projeto, o cliente em si, ou as pessoas que o represente. Cada estória (*user stories*)

é apresentada de acordo com a ordem de prioridade definida no *Sprint Backlog*, e o cliente dá o aceite final a estória e indica se o *Sprint* atingiu a meta proposta.

4.8 RETROSPECTIVA (*SPRINT RETROSPECTIVE*)

Ao término de cada *Sprint*, a equipe se reúne para analisar o *Sprint* que encerrou, com o propósito de um constante aperfeiçoamento.

Nessa reunião serão debatidos o que funcionou bem no *Sprint*, o que precisa ser melhorado e quais ações serão tomadas para colocar essas melhorias em prática. Geralmente o *Sprint Retrospective* tem um tempo médio de 4hs de duração.

5. APLICABILIDADE DO *SCRUM*

Neste capítulo será descrito como a utilização do *Scrum* organizou, agilizou e qualificou o desenvolvimento de um projeto real de uma empresa de software.

Para manter a privacidade da empresa que nos cedeu as informações, sua identificação será mantida em sigilo.

5.1. O PROJETO

O projeto em questão trata-se de um novo módulo financeiro. A empresa possui um software próprio muito pobre em relação ao controle de contas a pagar e receber. Por este motivo o diretor quer prioridade máxima neste projeto.

Nesse contexto o diretor da empresa assumiu o papel do *Product Owner*, o dono do projeto.

5.2. OS PERSONAGENS

Na realidade desta empresa ficou bem claro quem devia assumir cada papel. O Gerente de TI, responsável por facilitar o desenvolvimento do projeto e eliminar todas as possíveis barreiras do mesmo passou a ser o *Scrum Master*.

O time foi composto por quatro integrantes do departamento de desenvolvimento: um testador e três desenvolvedores.

Como dito anteriormente o diretor da empresa tornou-se o dono do projeto, o *Product Owner*.

5.3. PRIMEIRA REUNIÃO

O *Scrum Master* se reuniu com o *Product Owner* que lhe entregou o *Product Backolog*, ou seja, a lista de todas as tarefas que ele deseja que sejam desenvolvidas.

Juntos, o *Scrum Master* e o *Product Owner*, definiram a estimativa de cada tarefa. A Figura 5 ilustra uma das estórias do *product backlog*:

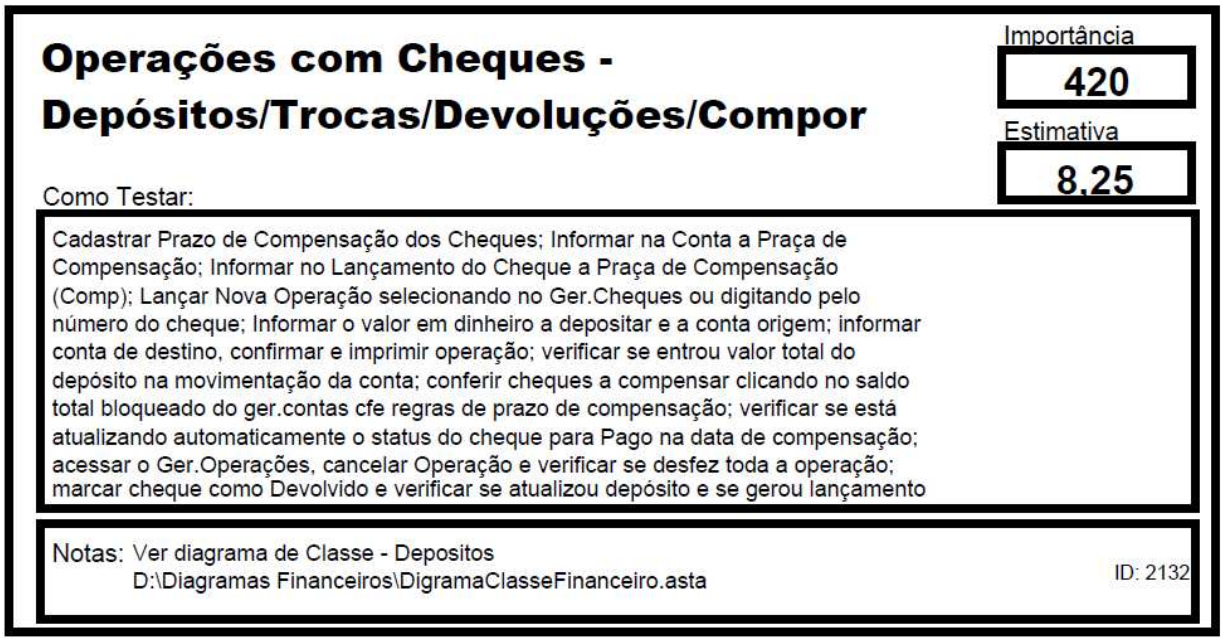

Figura 5 – Exemplo de estória do *Product Backlog*

É importante observar que para melhor organização essa empresa deu ao *product backlog* um formato de ficha, contendo informações essenciais como: título, descrição da forma de testes, nível de importância, estimativa e um ID para identificação e controle.

Após concluída a etapa de apresentação do *Product Backlog* para o *Scrum Master*, é marcada a *Planning Meeting* (Reunião de Planejamento).

A Reunião de Planejamento foi marcada para uma segunda-feira às 8 horas (primeiro horário).

Esta reunião tem o propósito de gerar um *Sprint* bom o bastante para o time trabalhar por algumas semanas tranquilamente e o *Product Owner* deixar a equipe trabalhar, satisfeito com os prazos.

Antes de começar a definir o *Sprint,* são apresentados os níveis de comprometimento do time (ou seja, os integrantes que possuem outras atividades não poderão se dedicar 100%). Os níveis foram definidos da seguinte forma:

- Testador: baixa atividade extra, comprometimento: 90%
- Programador1: sem atividades extras, comprometimento: 100%
- Programador2: médio nível de atividades extras, comprometimento: 70%
- Programador3: médio nível de atividades extras, comprometimento: 70%

Esta informação é extremamente importante para que possa ser informado um prazo real ao *Product Owner.*

Após essa etapa, através de uma conversa amigável ficou definido que o *Sprint* teria duas semanas de duração.

O próximo passo foi definir quantas estórias do *product backlog* fariam parte do *sprint backlog*. Para isso o grau de estimativa de cada estória foi fundamental.

Após algumas saudáveis e relativamente longas discussões entre o time e o *Product Owner,* foi definido desta forma o *Sprint Backlog*:

Alterar Tabelas conforme diagrama de Classe **Estimativa:0,25 ID: 2149**	Criar Cadastro de Planos e suas configurações **Estimativa:0,5 ID: 2149**	Criar TreeView para Cadastrar/Alterar/Excluir/Inativar o Plano de Contas **Estimativa:1,5 ID: 2149**
Criar tela de busca no Plano de Contas **Estimativa:1 ID: 2149**	Cadastrar um plano de contas padrão **Estimativa:0,5 ID: 2149**	Cadastrar um plano de custo padrão **Estimativa:0,5 ID: 2149**
Importar o plano de contas ja existente no cliente **Estimativa:0,5 ID: 2149**	Alterar Relatorio do Plano de Conta para filtrar o plano e imprimir em árvore lendo a tabela PLANOCONTAS **Estimativa:0,5 ID: 2149**	Revisar rotina de Lançamento no Caixa (Alt+L) onde gerar registro na PAGAR e PAGARRATEIO **Estimativa:0,5 ID: 2149**
Alterar Relatorios e Graficos para listar conforme o nivel selecionado **Estimativa:1 ID: 2149**	Revisar lugares onde usa a tabela PLANOCONTAS: Cadastro de Despesas/Receitas; Cadastro de Tipos de Movimento; Parcelamento da Entrada de Notas; Configurações do Sistema; **Estimativa:0,25 ID: 2149**	CheckList1 **Estimativa:0,5 ID: 2149**
Criar/Alterar Tabelas e Data Módulos conforme Diagrama de Classes **Estimativa:0,25 ID: 2134**	Criar configurações de Plano de Contas Padrão e Plano de Custo Padrão **Estimativa:0,25 ID: 2134**	Criar Tela de Configuração de Rateio por Conta **Estimativa:0,5 ID: 2134**
Criar Cadastro de Patrimônios **Estimativa:0,25 ID: 2134**	Adicionar na Entrada de Nota opção para cadastrar Patrimônio quando for operação de Compra de Ativo **Estimativa:0,5 ID: 2134**	Criar tela de Rateio de Custos ao lançar/alterar no Contas a Pagar **Estimativa:1 ID: 2134**
Criar rotina que gere rateio automaticamente conforme as configurações **Estimativa:0,25 ID: 2134**	Alterar Lançamento do Caixa para gerar na PAGARRATEIO conforme regras de rateio **Estimativa:0,25 ID: 2134**	Acrescentar filtro por Centro de Custo na Análise do Contas a Pagar e no Resumo por Contas **Estimativa:0,5 ID: 2134**
Acrescentar filtro por Patrimônio	Mostrar dados do rateio no Espelho	Tratar nos Relatórios de Compras para

nos Resumo por Contas **Estimativa:0,25 ID: 2134**	do Contas a Pagar **Estimativa:0,5 ID: 2134**	mostrar o Patrimônio **Estimativa:0,5 ID: 2134**
CheckList2 **Estimativa:0,5 ID: 2134**	Criar/Alterar Tabelas e Data Módulos conforme Diagrama de Classes **Estimativa:0,25 ID: 2199**	Adicionar no Cadastro da Conta o tipo da conta **Estimativa:0,25 ID: 2199**
Criar Cadastro de Formas de Pagamento e gerar os cadastros padrões (Pagamento em Dinheiro pelo Caixa, Pagamento em Dinheiro por Conta, Pagamento com Cheque Terceiros pelo Caixa, Pagamento com Cheque Terceiros, Pagamento em Cheque, Pagamento por TED, Pagamento por DOC, Pagamento por Depósito **Estimativa:0,25 ID: 2199**	Alterar Baixa para escolher primeiro a Forma de Pagamento, de acordo com a forma listar as contas, preencher Tipo de Movimento e habilitar campos para informar Dados do Cheque, Dados para Depósito/Transferência ou selecionar Cheque de Terceiros **Estimativa:0,5 ID: 2199**	Alterar Efetivação da Baixa para gerar operação com cheque de terceiros (CHEQUEALT) e se for uma título a pagar de um cartão de crédito (PAGAR.IDCONTAFINCARTAO <> 0) deve-se gerar um lançamento de crédito na mov.conta; **Estimativa:0,5 ID: 2199**
Adicionar no Lançamento do Contas a Pagar opção "Ref.Fatura de Cartão" para vincular com a conta de controle do cartão de crédito **Estimativa:0,25 ID: 2199**	Adicionar no Relatório de Pagamentos opção para detalhar as formas de pagamento (agrupar por IDTRANSACAOPAG e listar PAGAMENTOORIGEM) **Estimativa:1 ID: 2199**	Adicionar no Espelho do Título as formas de pagamento **Estimativa:0,5 ID: 2199**
Revisar Pagamento de pelo caixa **Estimativa:0,25 ID: 2199**	CheckList2 **Estimativa:0,5 ID: 2199**	Criar/Alterar Tabelas e Data Módulos conforme Diagrama de Classes **Estimativa:0,25 ID: 2204**
Criar Rotinas para gerar/atualizar repetições **Estimativa:0,5 ID: 2204**	Criar tela para configurar dados da frequência, gerar títulos e mostrar na grid as repetições já geradas; essa tela deve ser chamada quando alterar ou excluir um título; deve ter a opção de excluir somente o título ou toda a repetição; deve incluir botão para chamar essa tela na inclusão do títu **Estimativa:0,5 ID: 2204**	Alterar baixa para atualizar ou gerar novas repetições conforme a configuração **Estimativa:0,25 ID: 2204**
1 - Criar/Alterar Tabelas e Data Módulos conforme Diagrama de Classes **Estimativa:0,25 ID: 2132**	Criar tela para Configuração dos Prazos de Compensação **Estimativa:0,5 ID: 2132**	Adicionar Praça de Compensação no Cadastro da Conta e no Lançamento do Cheque **Estimativa:0,25 ID: 2132**
Criar a opção Nova Operação no Ger.Cheques que deve trazer preenchido os cheques já selecionados e permitir adicionar outros cheques através de uma pesquisa pelo número; deve ter opção para escolher o tipo de operação (Depósito/Troca/Compor) e permitir adicionar valores em dinheiro informando a **Estimativa:2 ID: 2132**	Criar a opção Gerenciar Operações no no Ger.Cheques que deve permitir Filtrar, Cancelar, Reabrir, Imprimir e marcar Cheque como Devolvido; **Estimativa:2 ID: 2132**	Criar Impressão da Operação **Estimativa:0,5 ID: 2132**
Criar rotina para cancelamento da Operação (IDOPERACAOCHEQUE) na MOVCONTA **Estimativa:0,5 ID: 2132**	No Gerenciador de Contas criar uma tela para demonstração de Saldo Bloqueado, onde o cheque automaticamente irá se desbloquear mediante a DATAPREVCOMPENSAR, mostrando um Total de Saldo Bloqueado, sendo possivel clicar e ver esse espelho do que esta bloqueado. **Estimativa:0,5 ID: 2132**	No Gerenciador de Cheques retirar a opção de (Registrar Operação com Cheque). **Estimativa:0,25 ID: 2132**

Criar rotina para atualzação do Status do Cheque e do Deposito conforme a data prevista de compensação **Estimativa:0,5 ID: 2132**	Ao marcar um cheque como Devolvido, deve atualizar o status do Deposito e gerar lançamento na MOVCONTA gravando o IDMOVCONTADEVOL na tabela CHEQUEALT **Estimativa:0,5 ID: 2132**	CheckList3 **Estimativa:0,5 ID: 2132**

O *Sprint* trouxe de forma abreviada a descrição de cada *product backlog*, sua estimativa e ID.

Sprint definido, data de inicio e fim também definidas, faltou apenas definir local e horário da reunião diária. Todos concordaram que a reunião seria mais rentável nos 15 primeiros minutos do dia. Portanto, diariamente das 8:00 as 8:15 horas (no máximo), na sala de desenvolvimento.

Agora estava tudo definido, portanto era hora de começar o *Sprint*.

5.4. *SPRINT* EM ANDAMENTO

Sabiamente, seguindo a dica de Henrique Kniberg, o *Scrum Master* organizou o time de forma que todos sentassem próximos um do outro, fator que posteriormente foi tido como um grande facilitador.

Cada um iniciou sua tarefa, e foi assim até o fim do dia.

5.4.1. A REUNIÃO DIÁRIA

A reunião de Planejamento durou quase um dia inteiro. No inicio do primeiro dia após esta reunião, as 8:00 horas estavam todos no local combinado para iniciar a primeira reunião diária.

Neste primeiro dia cada integrante do time fez um breve resumo das atividades que iria exercer neste dia que se iniciara e após esta etapa todos seguiram seu trabalho.

No segundo dia os integrantes relataram rapidamente as tarefas exercidas no dia anterior e apresentaram a programação para este dia.

Juntos, somaram os pontos das tarefas finalizadas no dia anterior e atualizaram o gráfico de *BurnDown*.

E assim, religiosamente, foi feito a cada dia do *Sprint*. As figuras 6 e 7 mostram a evolução dos gráficos:

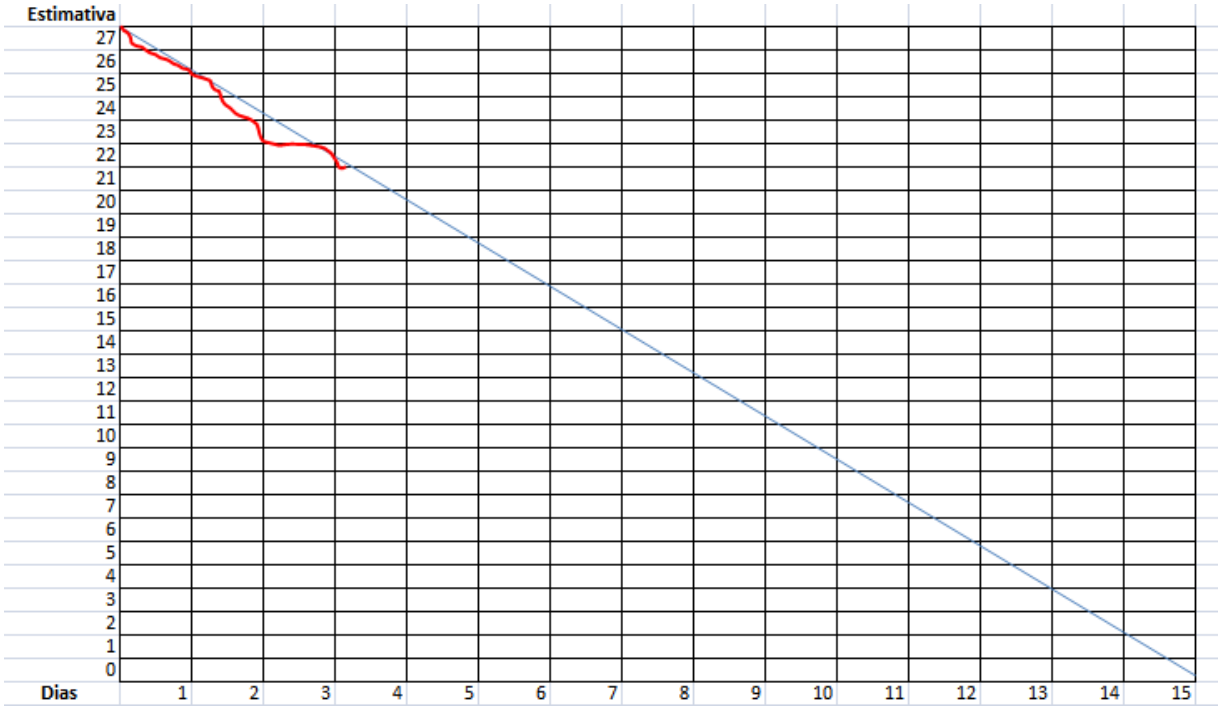

Figura 6 - Gráfico de *BurnDown* (3º dia de *Sprint*, 6,5 pontos de estimativa concluídos).

É importante observar que o eixo ‘y’ traz a soma das estimativas (27 pontos) e o eixo ‘x’ traz os dias da *Sprint*.

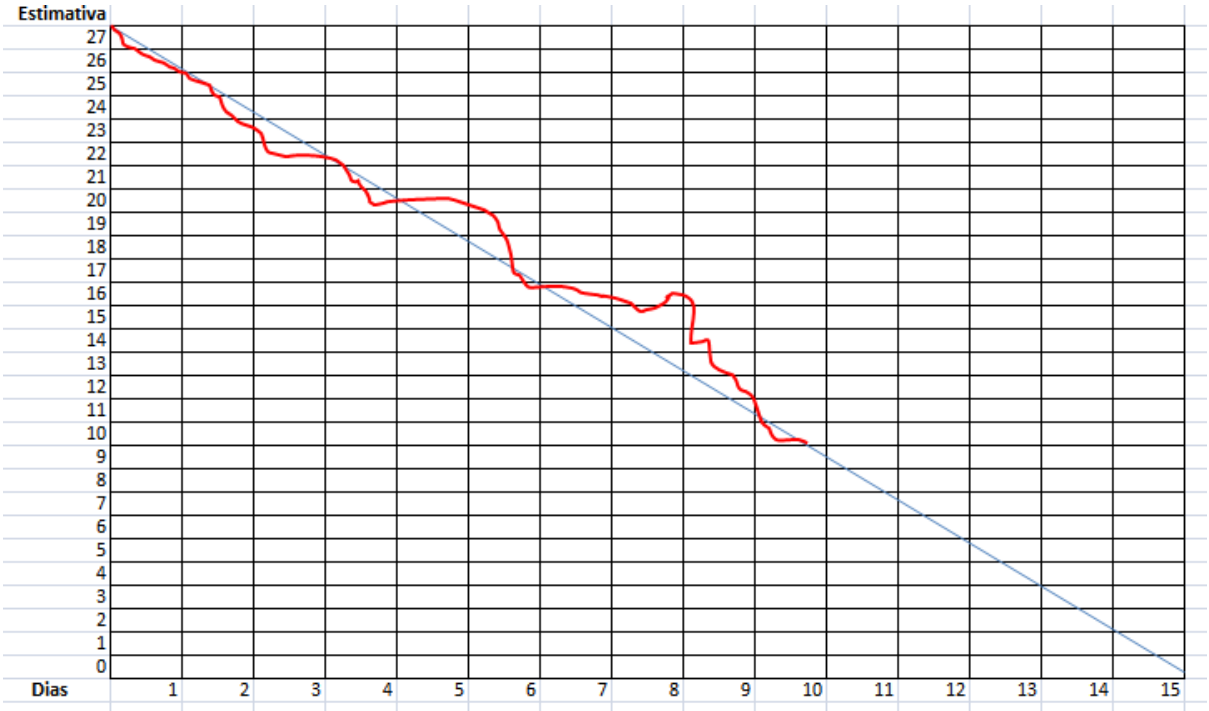

Figura 7 - Gráfico de *BurnDown* (10º dia de *Sprint*, 18 pontos de estimativa concluídos).

5.4.2. OBSTÁCULOS NO CAMINHO

No geral o *Scrum Master* estava satisfeito com o resultado, esporadicamente se preocupava com a interrupção dos outros departamentos para esclarecimentos de dúvidas, mas lá estava ele para não deixar que estes fatores prejudicassem o time e o *Sprint*.

O time estava unido em prol do projeto, a cada linha traçada no Gráfico de *BurnDown* era uma comemoração ou um lamento.

Esporadicamente um integrante queixava-se de alguma dificuldade na reunião diária, mas quase que imediatamente os outros integrantes do time apresentavam soluções antes mesmo que o *Scrum Master* pudesse dizer algo.

Nesta altura o *Sprint* já estava no fim e sim, o time vai conseguir!

5.5. REUNIÃO DE APRESENTAÇÃO

Enfim chegou o dia! O time conseguiu e o *Scrum Master* está orgulhoso pela conclusão do primeiro *Sprint* de sua equipe.

Agora é a hora da Revisão. É na *Sprint Review* que se reúnem o time, o *Scrum Master*, o *Product Owner* e quem estiver interessado em assistir a apresentação do *Sprint*.

E lá estavam: o *Scrum Master*, os quatro integrantes do time, o diretor da empresa (*Product Owner*) e mais dois colaboradores da empresa.

Ao final do *Sprint* o gráfico de *BurnDown* estava completo e foi apresentado a todos. A figura 8 mostra o gráfico de BurnDown no 15º dia de *Sprint*.

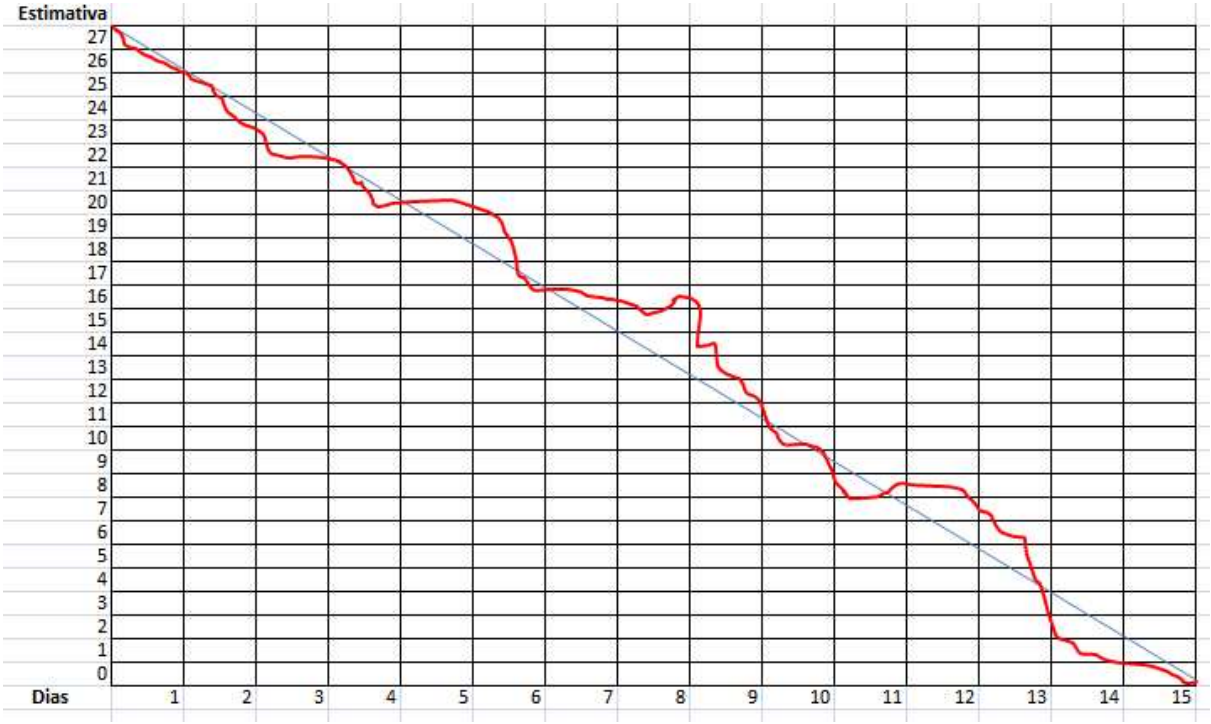

Figura 8 - Gráfico de BurnDown (15º dia de *Sprint*, 27 pontos de estimativa concluídos).

Além da apresentação em formato de relatório, as atividades concluídas no *Sprint* foram demonstradas na Prática.

Após elogios e sugestões, encerra-se a apresentação.

5.6. RETROSPECTIVA

Para quem achava que todo o procedimento estava concluído, enganou-se! A última reunião, e importantíssima é a Retrospectiva.

Nessa reunião todos os integrantes conversaram, admitiram falhas no *Sprint* finalizado e traçaram metas de melhoria para o próximo *Sprint.*

Em comum acordo ficou muito claro que além da organização, o *Sprint* trouxe para a realidade deste grupo qualidade, flexibilidade e agilidade no projeto.

6. CONCLUSÃO

Nunca chegará ao fim a lista de itens a serem melhorados quando o assunto em questão é Qualidade. Novas ferramentas serão criadas em todos os setores do mercado, novas estratégias, novas metodologias, etc. Além é claro, dos aperfeiçoamentos das existentes.

A Qualidade de Software veio acrescentar os avanços tecnológicos. As metodologias são cada vez mais úteis e eficazes. Os resultados são claros e satisfatórios. A partir das metodologias e boas práticas existentes são criadas evoluções cada vez mais abrangentes.

A Metodologia ágil, Scrum, conforme apresentada neste trabalho é um exemplo ideal de evolução dos modelos de qualidade. Os gerentes dos projetos sentiam a necessidade de organizar suas equipes de forma rápida e rentável, conseguindo um resultado robusto e que atendesse a real solicitação do cliente, o Scrum atendeu essa necessidade.

O Scrum funciona perfeitamente quando cada integrante assume seu papel e faz valer a confiança depositada nele pelos demais.

Possibilidades para criar existem de sobra, o que não existe mais é espaço para projetos com falhas e inconsistências. Para se tornar competitivo qualquer software precisa funcionar completamente, atender tudo o que é prometido para seu cliente e mostrar um diferencial. Fica então o desafio para os Gerentes de TI: usem o que for necessário para entregarem sempre mais do que lhe foi pedido. Metodologias para serem seguidas, adaptadas e aplicadas existem, só é necessário colocá-las em prática.

E nunca se esqueçam, Qualidade é inegociável!

REFERÊNCIAS BIBLIOGRÁFICAS

ASR, Consultoria e Qualidade – *CMMI – O que é?* Disponível em: http://www.asrconsultoria.com.br/qualidade-de-software/cmmi/o-que-e.php >, acessado em: 2 set, 2010.

DURÃES, Ramo. *O que é MPS-BR?* . Disponível em: <http://www.vstsbrasil.net/group/metodologias/forum/topics/o-que-e-o-mpsbr>, acesso em: 22 julho, 2010.

EGYPTIAN Organization for Standardization and Quality Control (EOS) *Quality – Introduction.* Disponível em: <http://www.eos.org.eg/Public/en-us/Quality/Introduction.htm>, acessado em: 23 julho, 2010.

FERNANDES, Daniel Batista. *Análise de Sistemas Orientada ao Sucesso: Por que os projetos atrasam?* – Editora: Ciência Moderna, 2005, 272p.

HENRIK, Kniberg. *Scrum e XP Direto das Trincheiras* - InfoQ,12/2008, 141 p.

JURAN, Joseph Moses. *A History of Managing Quality* - ASQC Quality Press, 1995.

JURAN, J.M., GRYNA, F.M. *Juran's quality control handbook.* New York: McGraw-Hill, 1988.

KOSCIANSKI, André; SOARES, Michel dos Santos Qualidade de Software - Aprenda as Metodologias e Técnicas Mais Modernas para o Desenvolvimento de Software – Editora: Novatec, 2ª Edição (2007), 395p.

MARÇAL, A. Freitas, SOARES, B., BELCHIOR, A. *Mapping CMMI Project Management Process Areas to SCRUM Pratices*, 3 1st Annual Software Engineering, Workshop, Loyola College, Baltimore, MD, USA (6-8 March 2007).

MESQUITA, João. *Por que adotar o MPS-BR?* Disponível em: http://longhigh.wordpress.com/2008/03/18/por-que-adotar-o-mps-br/ Acessado em: Junho, 2010.

PELEGRIN, Leandro. *Sistema de Gerenciamento para Estruturas Metálicas* - Projeto divulgado pela HSM Management em 3 julho,1997.

REBELO, Paulo. PSP (Personal Software Process), Disponível em: <http://phrebelo.wordpress.com>, acesso em: Agosto/2010.

ROCHA, Álvaro. *Qualidade de Software* . Disponível em: <http://www2.ufp.pt/~amrocha/mq/Qualidade%20de%20Software.pdf>, acesso em: 22 julho, 2010.

RODRIGUES, C. e Silva, Afonso. A *Qualidade na Construção ao Nível das Instalações Prediais de Águas e Esgotos. Sultuação e Perspectivas em Portugual*, 2007. Artigo Apresentado no 3.º Congresso Nacional 17 a 19 de Dezembro (2007), Coimbra, Portugal, Universidade de Coimbra.

SOFTEX, MPS-*BR Melhoria de Processos do Software Brasileito*. Disponível em: <http://www.softex.br/mpsbr/_home/default.asp>, acessado em: 12 maio, 2010.