UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATEUS LUIZ GAMBA

APLICAÇÃO DE MÉTRICAS DE SOFTWARE NO MÉTODO SCRUM DA METODOLOGIA ÁGIL

CRICIÚMA, NOVEMBRO DE 2009

MATEUS LUIZ GAMBA

APLICAÇÃO DE MÉTRICAS DE SOFTWARE NO MÉTODO SCRUM DA METODOLOGIA ÁGIL

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientadora: Profa. MSc. Ana Cláudia Garcia Barbosa

CRICIÚMA, NOVEMBRO DE 2009

MATEUS LUIZ GAMBA

Aplicação de Métricas de Software no Método Scrum da Metodologia Ágil

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Profa. MSc. Ana Claudia Garcia Barbosa (UNESC)
Orientadora

Prof. MSc. Gustavo Bisognin (UNESC)

Esp. Marcelo Mazon (Depto de TI - UNESC)

Dedico este trabalho aos meus familiares e amigos por todo o apoio recebido ao longo de minha formação e pela confiança e compreensão.

AGRADECIMENTOS

Primeiramente gostaria de me agradecer, por ter conseguido realizar esse trabalho, onde fiquei muitas e muitas madrugadas realizando-o com todo meu esforço, dedicação e carinho, e por ter conseguido realizar esse sonho de me formar em uma universidade.

Agradeço também:
Aos meus pais, pela ajuda, educação, amor, respeito e confiança para que eu alcançasse esse objetivo.

A Deus e ao Universo pelo poder que me deram e que estão me dando para completar minhas futuras realizações.

À minha orientadora Ana Cláudia Garcia Barbosa pela compreensão e empenho por ter acreditado na possibilidade da realização deste trabalho.

A todos meus amigos que acreditaram e me apoiaram nessa trajetória.

A Monique pelo carinho, compreensão, dedicação e alegria a mim.

Ao Roger pela parceria nos trabalhos acadêmicos.

A Webmais Sistemas pela ajuda para a realização desse trabalho.

A Profª Leila e o Profº Gustavo pela força que me deram em alguns momentos de dificuldade na elaboração desse trabalho.

E por fim, a todas pessoas que me ajudaram indiretamente.

RESUMO

Este trabalho estudou a aplicação de métricas de software para obter estimativas de prazo de tempo de desenvolvimento de uma iteração do método Scrum da metodologia ágil. Além da apresentação da fundamentação teórica sobre metodologias ágeis, método Scrum e métricas, foi realizado um estudo de caso para validar as métricas definidas, contribuindo para aumentar a precisão do tempo de desenvolvimento no Scrum, onde foram utilizadas as métricas *Ideal Day*, *Planning Poker* e Pontos de Função. No desenvolvimento do projeto a metodologia ágil se propõem a contribuir com o desenvolvimento rápido e flexível a mudanças. Essa metodologia atende as necessidades para obter resultados significativos no processo de desenvolvimento de software. O gerenciamento de projetos pelo método ágil Scrum, oferece flexibilidade, é adaptável as constantes mudanças, qualificando assim o projeto e com a aplicação de métricas de software pode-se obter estimativas mais precisas.

Palavras-chave: Engenharia de Software; Metodologias de Desenvolvimento Ágil; Métricas de Software.

ABSTRACT

This study is on the application of software metrics to obtain estimates of stated period of time in the development of an iteration of the Scrum method from the agile methodology. Besides the presentation of theoretical background on agile methodologies, Scrum method and metrics, it was carried out a case study to validate the defined metrics, contributing to increase the time precision of the development in the Scrum, which were used the metrics Ideal Day, Planning Poker and Function Points. During the development of the project, the agile methodology contributes with the fast and flexible development to changes. This methodology fulfills the necessities to obtain significant results in the process of software development. The management of projects agile method Scrum, offers flexibility, is adaptable the constant changes, qualifying the project and with the application of software metrics can obtain more precise estimates.

Keywords: Software Engineering; Methodologies of Agile Development; Software Metrics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AIE	Arquivos de interface externa
ALI	Arquivos lógicos internos
AR	Arquivos referenciados
BFPUG	*Brazilian Function Point Users Group*
CE	Consulta externas
CSS	*Cascading Style Sheets*
DER	Dados elementares relacionados
EE	Entradas externas
Html	*HyperText Markup Language*
PF	Pontos de função
IEC	*International Electrotechnical Commission*
IFPUG	*International Function Point Users Group*
ISO	*International Organization for Standardization*
PFA	Pontos de função ajustados
PFNA	Pontos de função não ajustados
RLR	Registros lógicos referenciados
SE	Saídas externas
TD	Tipos de dados
VFA	Valor do fator de ajuste

SUMÁRIO

1 INTRODUÇÃO

Em projetos em que há muitas mudanças, refazer partes do código é uma atividade inevitável, geralmente com equipes pequenas e prazos de entrega curtos, o desenvolvimento de um planejamento para suprir essas necessidades se torna algo difícil.

Devido esses motivos e entre outros se originou a metodologia ágil com a intenção de ser adaptável a processos de desenvolvimento e mudanças de software de uma maneira flexível e rápida, alterando em partes o paradigma da engenharia de software tradicional (AGILE MANIFESTO, 2009).

De acordo com Koscianski e Soares (2007) a metodologia ágil não menospreza os processos, ferramentas, documentação, negociação de contratos nem planejamento, mas sugere que estes vêm em segundo plano.

A metodologia ágil enfatiza o desenvolvimento iterativo, incremental e o compartilhamento de idéias para o desenvolvimento de software, onde os projetos sofrem constantes alterações, agindo de forma a adequar-se as mudanças. Com isso a metodologia ágil foca em como receber, avaliar e responder às mudanças, possibilitando atender aos requisitos do cliente (ISOTTON NETO, 2008).

Atualmente existem inúmeros métodos da metodologia ágil, podendo-se destacar as seguintes: *Crystal*, *Dynamic Systems Development Method*, *Extremme Programming*, *Feature Driven Development*, Scrum, *MSF for Agile* entre outras (AGILE MANIFESTO, 2009, nossa tradução).

Um método que está sendo citado nos dias de hoje é a Scrum, sendo utilizado por grandes corporações como Google, Nokia e Yahoo entre outras.

De acordo com Pressman (2006) o método Scrum tem como objetivo trabalhar com pequenas equipes para maximizar a comunicação, conhecimento e diminuir a supervisão,

podendo-se assim atingir um objetivo comum. Este método apresenta como principais atividades: requisitos, análise, projeto, evolução e entrega.

A metodologia ágil está em fase de aprimoramento, deixando a desejar outros requisitos importantes para atingir a qualidade do software, podendo-se destacar as estimativas de custo, avaliação da produtividade e satisfação do cliente. Por essas razões Pressman (2006) define que o desenvolvimento do projeto deve utilizar métodos para medir, ou seja, métricas para ter controle e avaliação desses requisitos.

As métricas na computação referem-se a mensurar o software desenvolvido ou que está sendo desenvolvido, com as razões de indicar a qualidade, avaliar a produtividade da equipe e ter estimativas (PRESSMAN, 2006).

Segundo Inthurn (2001) as métricas podem ser divididas nas seguintes propriedades: produtividade, qualidade, técnicas, tamanho, ponto de função e pessoas.

Com as afirmações de Pressman, pode-se definir que para um projeto ter maior probabilidade de ser bem sucedido é necessário aplicar métricas para controlar o desenvolvimento, ter estatísticas e comparações, pois não há como ter uma precisão ou controle naquilo que não se pode mensurar.

Devido o método Scrum ser atual, é carente nos aspectos de medições e métricas, devido a falta de documentação, avaliações sobre sua utilização, diretrizes e maiores detalhes para sua efetiva aplicação.

1.1 OBJETIVO GERAL

Analisar métricas de software aplicadas no método Scrum da metodologia ágil.

1.2 OBJETIVO ESPECIFICOS

Os objetivos específicos do projeto são baseados nos seguintes itens:

a) pesquisar os princípios e processos da metodologia ágil na gerência e construção de software;
b) compreender a forma de utilização do método Scrum;
c) analisar as métricas de software;
d) propor métricas a serem utilizadas no método Scrum;

1.3 JUSTIFICATIVA

Em organizações com pequenas equipes de desenvolvimento geralmente não se tem uma estrutura ou recursos suficientes para adotar certos formalismos da engenharia de software. Em muitos casos isso gera um software com baixa qualidade, podendo ter atraso de prazos e custos predefinidos, inviabilizando atualizações futuras.

Na engenharia de software existem vários processos definidos, dentre os conhecidos existe a metodologia ágil, que procura ter o mínimo de documentação e ainda vem a contribuir para otimização dos esforços dos recursos humanos como financeiros de uma organização (BORGONOVO; SILVA, 2007)

A metodologia ágil possui diversos métodos, um que pode se destacar é o Scrum, que visa ser utilizado em pequenas equipes que trabalham juntas para atingir um objetivo comum, podendo se adequar bem nessas organizações.

Segundo Pressman (2006) o processo é medido a fim de melhorá-lo e aumentar sua qualidade, precisão na estimativa e maior produtividade. Aplicando métricas no método Scrum da metodologia ágil tem-se uma forma objetiva para saber se o processo de desenvolvimento do software está apresentando melhorias significativas.

Com a mensuração, as organizações conseguem atingir níveis superiores de maturidade, ajudando os tomadores de decisões “fornecendo informações significativas com relação à qualidade, adequação e progresso evolutivo de processos, produtos e projetos de software” (MACHADO; SOUZA, 2008, p. 1).

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em seis capítulos. O Capítulo um contextualiza a introdução referente ao trabalho, com a definição do objetivo desse estudo e também os objetivos específicos e concluindo então com a justificativa do trabalho realizado.

A metodologia ágil no Capítulo dois visa apresentar o surgimento de formas mais ágeis de se desenvolver um software, no capítulo três explica detalhadamente o método proposto Scrum. No Capítulo quatro são apresentadas as métricas de software, onde são descritas as aplicadas.

A apresentação dos trabalhos correlatos está descrito no Capítulo cinco, e por fim é abordado no Capítulo seis a aplicação das métricas de software no método Scrum, no qual foi aplicado em um estudo de caso e ao final realizada uma análise para verificar qual se obteve maior compatibilidade.

2 METODOLOGIA DE DESENVOLVIMENTO

De acordo com Kantorski e Kroth (2004) as metodologias são um conjunto de atividades que auxiliam na produção de software, que sendo utilizadas resultam em um produto que reflete do qual o processo foi conduzido.

2.1 ORIGEM DAS METODOLOGIAS DE DESENVOLVIMENTO

Antes das metodologias de engenharia de software, os sistemas eram escritos sem nenhum plano defino e o projeto era repleto de diversas decisões de curto prazo, esses procedimentos eram conhecidos pela expressão "codificar e consertar". Isso conseguia atender as pequenas organizações de software, mas à medida que o sistema crescia, tornava-se mais difícil atender as exigências (ZANATTA, 2004).

Para os grandes sistemas, existiam as necessidades das atividades de desenvolvimento do software serem mais eficientes, disciplinadas e previsíveis. Com isso surgiram as metodologias de desenvolvimento de software.

As metodologias impõem disciplinas no desenvolvimento de software, assim tornando mais previsível e eficiente, fazendo o desenvolvimento do projeto ter um processo de trabalho focando no planejamento (FOWLER, 2005, nossa tradução).

Ainda um dos grandes desafios da engenharia de software é entregar um software dentro do prazo estabelecido, com o orçamento previsto e com todos os requisitos atendidos (ZANATTA, 2004).

Do modo que as empresas estão em constantes mudanças, onde as necessidades dos usuários finais evoluem e novas ameaças de concorrência surgem, fica difícil prever como os sistemas de computadores terão que evoluir (PRESSMAN, 2006).

Com a utilização das metodologias para resolver esses tipos de problemas, muitas vezes são vistas como um “gargalo” pela equipe de desenvolvimento, pois exigem diversos processos, documentação abrangente e planos. O cliente somente via o software quando estava pronto, que às vezes não era realmente o esperado (BORGONOVO; SILVA, 2007).

A partir desses problemas da metodologia, desenvolvedores, produtores e consultores de software resolveram adotar novos princípios de metodologias, onde teriam de ser ágeis, com isso foi feito o Manifesto Ágil.

2.2 MANIFESTO ÁGIL

Em 2001, Kent Beck e outros 16 desenvolvedores, produtores e consultores de software, motivados pelas suas experiências, iniciaram uma discussão sobre como aplicar métodos mais ágeis para o desenvolvimento de software, assim, com essa discussão resultou-se no chamando Manifesto Ágil (ZANATTA, 2004).

O Agile Manifesto (2009, nossa tradução) elaborou um documento que reúne os princípios e práticas desta metodologia de desenvolvimento, onde também apresenta diversos métodos ágeis, que se destacam Scrum, *Crystal Clear*, *Extreme Programming*, *Adaptive Software Development*, *Feature Driven Development* e *Dynamic Systems Development Method*.

Após o Manifesto Ágil, alguns dos integrantes do manifesto, mais outras pessoas formaram a Aliança Ágil (*Agile Alliance*), uma organização não lucrativa que promove desenvolvimento ágil.

De acordo com Agile Manifesto (2009, nossa tradução) a fim de melhorar o desenvolvimento de software, são valorizados os seguintes itens:

a) indivíduos e interações em vez de processos e ferramentas;

b) softwares funcionando em vez de documentação abrangente;

c) colaboração do cliente em vez de negociação de contratos;

d) resposta a modificações em vez de seguir um plano.

Os métodos ágeis surgem desses valores, onde eles não desprezam as metodologias tradicionais da engenharia de software e nem são contrarias as sólidas práticas, mas sim, dão mais valores aos itens citados a cima.

Os métodos ágeis foram desenvolvidos para vencer as fraquezas da engenharia de software convencional, trazendo diversos benefícios, mas também não sendo aplicadas em todos os projetos, produtos, pessoas e situações (PRESSMAN, 2006).

2.3 METODOLOGIA ÁGIL

A metodologia ágil não é exatamente metodologia, mas sim, um conjunto de práticas e princípios, cuja finalidade é guiar o gerenciamento do projeto.

Para que uma metodologia seja ágil, ela não deve tentar prever o futuro, e sim apenas aceitar as mudanças, sendo particularmente utilizadas para projetos que têm que lidar com mudanças rápidas ou imprevisíveis nos requisitos.

De acordo com Kantorski e Kroth (2004) os métodos ágeis tem as seguintes características:

a) desenvolvimento iterativo e incremental, onde deve ocorrer a entrega de uma versão do software de duas a quatro semanas;

b) comunicação em tempo real, e não através de documentos;

c) redução de produtos intermediários, como a documentação excessiva;

d) requisitos instáveis.

Como todos os modelos de processo de desenvolvimento de software, os métodos ágeis também possuem alguns princípios, onde o Agile Manifesto (2009, nossa tradução) define doze princípios para aqueles que querem alcançar a agilidade:

a) a prioridade é satisfazer o cliente através da entrega continua e rápida da versão do software;

b) modificações sobre os requisitos são sempre aceitas, mesmo que tardias no desenvolvimento. As modificações dão vantagens competitivas ao cliente;

c) entrega do software freqüentemente, a cada duas semanas até dois meses, de preferência com menor tempo possível;

d) os especialistas do negócio e os desenvolvedores trabalham juntos diariamente durante todo o projeto;

e) manter os indivíduos da equipe motivados, fornecendo-lhes ambiente a confiança necessária para o trabalho ser executado;

f) fazendo que a equipe se comunique face-a-face para a troca de informações serem mais eficiente;

g) deixar sempre o software funcionando ao decorrer do progresso;

h) promover desenvolvimento sustentável. A equipe, os patrocinadores e os usuários devem manter um ritmo constante;

i) atenção continua a excelência técnica e ao bom projeto melhoram a agilidade;

j) simplicidade é essencial;

k) a partir das equipes auto-organizadas surgem as melhores arquiteturas, requisitos e projetos;

l) em intervalos regulares, a equipe reflete como se tornar mais eficiente. Então ajusta seu comportamento para atingir esse objetivo.

De acordo com Fowler (2005, nossa tradução) os métodos ágeis têm os seguintes aspectos que diferenciam das metodologias tradicionais:

a) adaptativos ao invés de preditivos: pois buscam a adaptação as rápidas mudanças do mercado, quando a necessidade do projeto muda, a equipe tem que se adaptar essa mudança;

b) orientado a pessoas ao invés de processos: a metodologia ágil afirma que nenhum processo jamais será equivalente a habilidade da equipe de desenvolvimento.

A metodologia ágil compreende num processo de desenvolvimento baseando em entregar incrementais do software, assim permitindo que o cliente tenha um rápido retorno do investimento que está sendo feito.

As entregas incrementais são obtidas por meio do desenvolvimento em iterações, de duas semanas a dois meses, tendo foco em resolver as necessidades do cliente e não nas regras de contrato, assim a equipe de desenvolvimento tem que manter contato direto com o cliente durante todo processo (SATO, 2007).

Dentro dos métodos ágeis citados, o Scrum será apresentando detalhado devido ser o foco do trabalho.

3 MÉTODO SCRUM

Scrum é um método ágil para gerenciamento de projeto ou atividades complexas com base no processo de desenvolvimento iterativo e incremental da abordagem orientada a objetos (ZANATTA, 2004 apud SCHWABER, 2002).

Segundo Schwaber (2004, nossa tradução) o método Scrum é utilizado em trabalhos complexos, onde é quase impossível prever o que ira ocorrer durante o desenvolvimento do projeto. Assim, Scrum oferece um conjunto de práticas que mantém tudo visível. Isso permite que os membros do Scrum saibam exatamente o que está acontecendo e quais são os ajustem necessários para manter o projeto em direção as metas desejadas.

O método Scrum não define ferramentas e nem técnicas de desenvolvimento, mas sim, como as equipes devem trabalhar em ambientes com constantes alterações e com surgimentos de novos requisitos (ZANATTA, 2004).

De acordo Schwaber (2004, nossa tradução) pode se disser que Scrum incentivando cada membro a participar, mostrando suas idéias, críticas e sugestões para chegar ao seu objetivo de forma mais satisfatória.

3.1 ORIGEM DO SCRUM

O Scrum[1] surgiu com o artigo *The New Product Development Game* publicado na *Harvard Business Review* em Janeiro de 1968, escrito por Takeuchi e Nonaka. Nesse artigo é descrito as dez melhores e mais inovadoras praticas em empresas japonesas (MARTINS, 2007).

[1] O termo Scrum é uma metáfora a uma tática utilizado no jogo de rúgbi para recuperar a bola perdida.

Segundo Isotton Neto (2008) Jeff Stutherland, em 1994, após ler o artigo de Takeuchi e Nonaka começou a introduzir técnicas de Scrum dentro da empresa Easel Corporation onde era vice-presidente. No ano de 1995, Ken Schwaber começou a trabalhar junto com Stutherland na formação do Scrum. E em 1996, criaram a empresa Individual Inc. onde começaram a ampliar o Scrum, criando versões mais atuas.

3.2 CARACTERÍSTICAS DO SCRUM

Segundo Martins (2007) e Zanatta (2004) o Scrum tem como característica a flexibilidade para os processos imprevisíveis e complexos do desenvolvimento, que ao decorrer do desenvolvimento envolve aspectos de mudanças nos requisitos, recursos e tecnologia.

O Scrum trabalha com partes pequenas de cada vez no projeto, conhecida como iterações. Nessas iterações são realizadas captura de requisitos, análises, programação e testes (MARTINS, 2007).

Por o Scrum ser iterativo, consegue antecipar a entrega do projeto, assim fornecendo para o cliente de uma forma incremental.

As equipes em Scrum são auto-organizadas, sempre tendo comunicação verbal entre os membros e entres as partes que estejam envolvidas no projeto. A equipe deve possuir a habilidade de responder as questões de forma ágil aos desafios.

De acordo com Linda e Norman (2000, nossa tradução), os princípios de Scrum são: equipes de sete membros no máximo, apenas desenvolvem e não definem o que tem que ser desenvolvido, divide o desenvolvimento em intervalos de tempos e utilizado para projetos com os requisitos pouco estáveis.

O Scrum é composto papéis e responsáveis, conhecidos como *Product Owner*, *Scrum Master* e *Scrum Team*. O seu processo é dividida pelas fases de *PreGame*, *Game* e

PostGame, onde são realizadas as práticas do *Sprint Planning Meeting*, *Daily Scrum*, *Sprint Review Meeting* e *Sprint Retrospective* e também composto pelos artefatos *Product Backlog* e *Sprint Backlog* que são descritos nos itens a seguir.

3.3 DESENVOLVIMENTO COM SCRUM

O gerenciamento do projeto com Scrum, se início definindo a lista dos requisitos necessários para o desenvolvimento, essa lista é conhecida como *Product Backlog*.

O desenvolvimento ocorre num período de no máximo trinta dias, conhecido como *Sprint*, onde são planejados quais requisitos do *Product Backlog* tem maior prioridade, assim criando uma nova lista conhecida como *Sprint Backlog*. Durante o *Sprint* não ocorrem mudanças nos requisitos selecionados. Ao decorrer do *Sprint* a equipe desenvolve os requisitos do *Product Backlog*.

A cada dia do *Sprint* é realizada uma reunião onde cada membro relata como está o desenvolvimento de sua tarefa e se necessita de ajuda ou recursos para continuar.

Sempre no último dia de cada *Sprint* é realizada uma revisão onde a equipe mostra o trabalho desenvolvido para o *Product Owner* (pessoa responsável pelo projeto) e para o *stakeholders* (pessoas interessadas pelo projeto). Também é realiza uma retrospectiva do *Sprint*, que nada mais é que uma revisão do trabalho feito.

Para que esses passos sejam seguidos de forma correta, há uma pessoa, chamada de *Scrum Master*, responsável para gerenciar e liderar a equipe. Tendo também o dever de fazer o levantamento dos requisitos até a entrega do projeto final.

A Figura 1 representa o funcionamento do Scrum, em como é dividido seus processos.

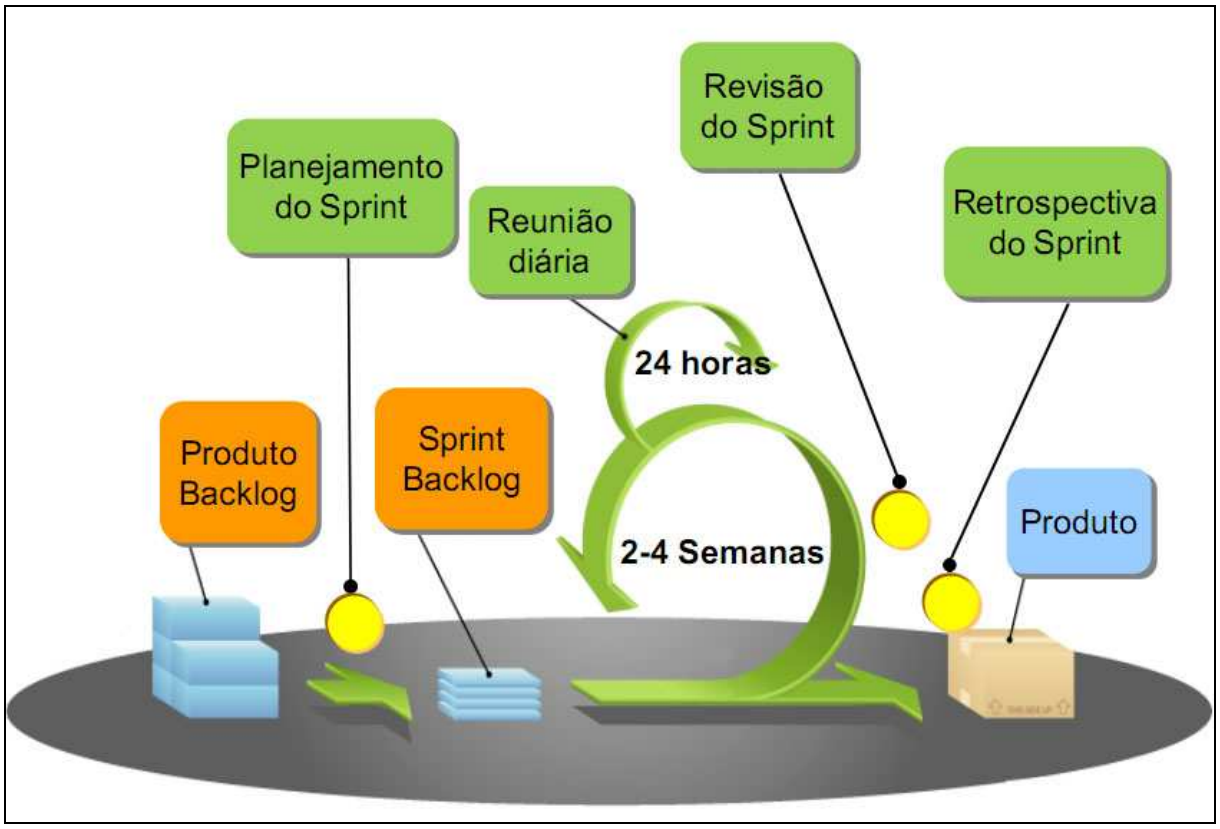


Figura 1. Representação do Ciclo do Scrum
Fonte: Adaptado de *Mountain Goat Software*[2] (2009)

3.4 PAPÉIS DO SCRUM

O Scrum é dividido em papéis e responsabilidades que desempenham tarefas durante o processo de utilização das práticas.

Os papéis são compostos pelo *Product Owner*, *Scrum Master* e *Scrum Team* que são descrito a seguir conforme Martins (2007) e Zanatta (2004, apud SCHWABER, 2002).

3.4.1 Product Owner

É o representante dos *stakeholder* em defender os interesses do negócio no projeto, ou seja, definir as funcionalidades do software.

[2] Disponível em http://www.mountaingoatsoftware.com/scrum-figures

O *Product Owner* tem como objetivo definir os requisitos e prioridades do *Product Backlog*, e também, por revisar e aceitar as entregas ao final de cada *Sprint*. Tem a responsabilidade pela liberação da versão e a da criação do próximo *Sprint Backlog*.

3.4.2 Scrum Master

É o gerente do projeto, responsável por garantir a realização das tarefas e pelo sucesso do Scrum. É considerado como a "ponte" entre a equipe, a tecnologia e os negócios, também tendo a responsabilidade pelas seguintes atividades:

a) remover obstáculos e impedimentos do projeto;

b) ser o intermediador da equipe com o *Product Owner*;

c) organizar as reuniões diárias;

d) conduzir a revisão de cada *Sprint*;

e) proteger a equipe de interferências externas.

3.4.3 Scrum Team

Conhecido também como os membros da equipe ou time. O *Scrum Team* tem a responsabilidade de executar e completar os itens definidos em cada *Sprint*, garantindo que todas as funcionalidades necessárias sejam entregues.

Tem as seguintes características:

a) organizar e gerenciar suas próprias tarefas;

b) criação do *Sprint Backlog*;

c) revisão dos itens do *Product Backlog*;

d) verificação dos impedimentos que precisam ser retirados;

e) conhecimento técnico sobre todo o processo de desenvolvimento do software;

f) ter pessoas com capacidades de analisar as soluções, codificá-las e testá-las.

3.5 FASES DO SCRUM

Segundo Isotton Neto (2008) o Scrum é separado nas fases de *PreGame*, *Game* e *PostGame*.

a) *PreGame*: essa fase é dividida em dois estágios, sendo o planejamento (*Planning*) e a arquitetura (*Architecture/Staging*).O planejamento consiste no desenvolvimento das atividades iniciais para o *Product Backlog*. Nele também estão definidas as estimativas de tempo para cada atividade e o custo. Na arquitetura é defino como as atividades do *Product Backlog* serão implementadas e quais serão os membros de cada equipe;

b) *Game*: é conhecida como a fase de desenvolvimento. É realizada em *Sprints*, onde cada *Sprint* são sempre incrementadas novas funcionalidades até que o produto esteja finalizado. Os riscos são avaliados periodicamente, e também quando necessários são inseridos novas atividades no *Product Backlog*;

c) *PostGame*: é a preparação da entrega do produto. Nessa fase são realizados os testes, a integração, a documentação e a preparação do material para treinamento.

3.6 PRÁTICAS E ARTEFATOS DO SCRUM

O Scrum iniciasse com o artefato *Product Backlog* para após realizar a prática do *Sprint*, onde o Sprint é composto por outras práticas e artefatos.

Nos itens a seguir são descritos o *Product Backlog* e o *Sprint*.

3.6.1 Product Backlog

O *Product Backlog* é o ponto inicial do Scrum, sendo que nessa fase ocorre a coleta dos requisitos para o desenvolvimento do projeto. Após a coleta, são realizadas reuniões com os *stakeholders* para definir os "itens com todas as necessidades do negócio e os requisitos técnicos a serem desenvolvidos" (ZANATTA, 2004, p. 50).

Com a definição dos requisitos (ou estórias) é criado uma lista com as atividades que serão desenvolvidas para a construção do projeto, essa lista é chamada de *Product Backlog*. Para a criação dessa lista, deve possuir as funcionalidades, características e padrões, prioridades e estratégias do projeto.

Este documento tem que conter requisitos visíveis e requisitos técnicos dos quais o cliente não necessariamente precise saber. Os requisitos devem se detalhados e caso a expectativa de codificação ultrapasse dez dias pode-se quebrar em sub-requisitos.

Segundo Isotton Neto (2008) o *Product Backlog* tem as seguintes características;

a) nenhum item pode ser adicionado à lista sem o consentimento do *Product Owner*;
b) mudanças podem ser feitas de acordo com as necessidades que aparecem;
c) os itens com maior prioridade sempre serão selecionados para os próximos *Sprint*;
d) o *Product Owner* é o responsável pela *Product Backlog*.

De acordo com Martins (2007) o *Product Backlog* pode ser elaborado com as seguintes colunas:

a) Iter#: número da iteração ou *Sprint* em que o item será trabalho. No início do projeto somente os itens do primeiro *Sprint* são detalhados, os demais itens não necessariamente precisam ser bem detalhados, pois ao decorrer do projeto

poderão surgir outros requisitos que não foram identificados pela equipe ou pelo *Product Owner*;

b) Id: é o código único para cada item da lista, servindo como referência. Mesmo surgindo alterações na descrição o Id deve continuar o mesmo;

c) Assunto: é uma forma de agrupar os itens em assuntos, apenas servindo para ajudar os itens do *Product Backlog*;

d) Item: é a descrição do item;

e) Prior: é a prioridade do item, começando do número mais baixo, onde tem maior prioridade, e diminuindo a prioridade sucessivamente de acordo com o incremento do número;

f) Estimativa: estima de tempo para concluir o requisito, podendo ser representado em horas, dias ou qualquer outro tipo de métrica.

g) Risco: representa a certeza quando à estimada, quando maior for o risco, maior é a incerteza da estimativa informada. O risco pode ser representado em três escalas, que são (A) alta, (M) média e (B) baixa.

A Tabela 1 ilustra um exemplo do *Product Backlog* com as colunas descritas acima.

Tabela 1. *Product Backlog*

PRODUCT BACKLOG						Atualização:13/01/2005	
Iter#	**Id**	**Assunto**	**Item**	**Prior.**	**Resp**	**Estimativa**	**Risco**
1	1	Filas/Backup	Mudar arquitetura para trabalhar com filas de chamadas de processamento	1	AG	4	A
1	2	Watchdog	Criar mecanismo de watchdog e sistemas de alarmes por e-mail	1	AG	4	M
1	4	Web	Migrar telas para Asp+IIS	1	AC	0,5	B
1	5	Segurança	Logar as pesquisas de gravação feitas pelos usuários e as reproduções	2	AC	2	
1	3	Playback	Fazer reprodução através de playlist	2	AG	2	B
1	8	Segurança	Criptografar DVD com as gravações	2	AC	1	M
1	6	Playback	Enviar gravação por e-mail	3	C	0,5	M

Fonte: Adaptado de Martins, J (2007)

O *Product Backlog* pode variar de acordo com cada projeto, onde a projetos que necessitam de mais informações como também menos, podendo ser adaptada para cada estilo de equipe e projeto, de forma que deixa a equipe mais ágil.

3.6.2 Sprint

Conforme Martins (2007) *Sprint* é o nome dado para cada intervalo de tempo que não possa passar de trinta dias, onde a equipe executa um conjunto de atividades chamada de *time box*. Essas atividades consistem em:

a) desenvolver: define as necessidades para implementação dos itens do *Product Backlog* como: análise, projeto, programação, testes e documentação;
b) empacotamento: empacotar o projeto de forma que seja executado ou distribuído no cliente;
c) revisão: as equipes reúnem-se para apresentar e discutir o trabalho realizado e revisar o progresso, resolver os problemas e adicionar novos itens ao *Product Backlog*;
d) ajustes: com as informações obtidas na reunião de revisão devem fazer as mudanças necessárias no planejamento.

O intervalo de trinta dias é baseado na complexidade do produto e na avaliação dos riscos, onde a equipe é forçada a se concentrar nos requisitos definidos como mais importantes, assim não perdendo tempo em tarefas desnecessárias

De acordo com a Yoshima (2007) a vantagem de se trabalhar com intervalo de tempo é que no final do intervalo tem algo de valor para se demonstrar aos *stakeholders*.

Conforme Zanatta (2004, apud ABRAHAMSSON, 2002) o *Sprint* inclui fases tradicionais do desenvolvimento de software, como: requisitos, análise, projeto e entrega.

A Figura 2 representa a estrutura do *Sprint*.

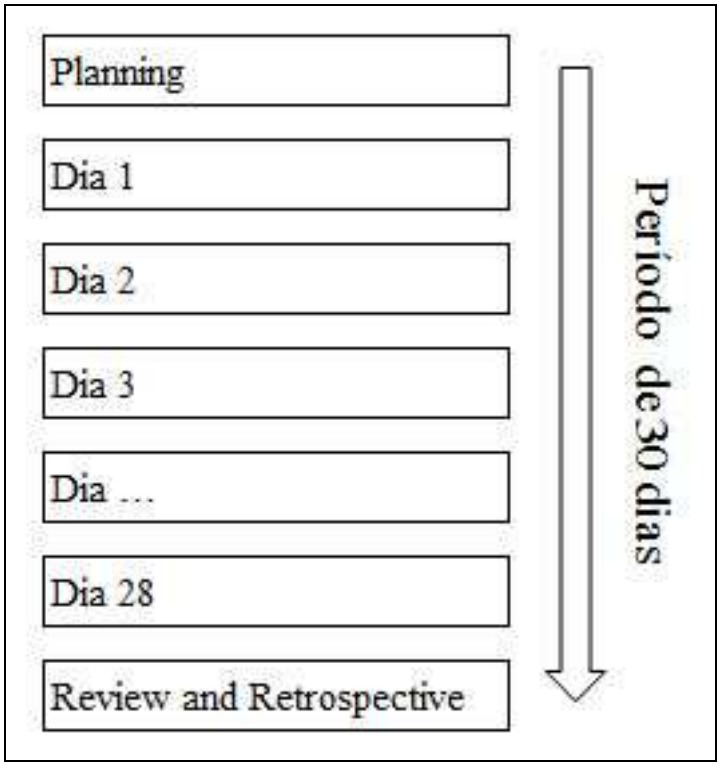


Figura 2. Estrutura do *Sprint*

Os *Sprints* realizados devem seguir sempre a mesma estrutura: o primeiro dia se tem o *Sprint Planning Meeting*, onde é feito um planejamento do que será desenvolvido durante o *Sprint*, no planejamento selecionados os itens do *Product Backlog* criando uma lista conhecida como *Sprint Backlog*. Após o do planejamento se inicia o desenvolvimento, onde a equipe executa e cumpre o planejamento do *Sprint Backlog*, e o último dia é realizado *Sprint Review Meeting* se avalia os resultados e se faz os ajustes para obter melhorias e o *Sprint Retrospective* que é a revisão do trabalho feito.

Durante os dias do *Sprint Backlog* são realizadas reuniões periódicas denominadas de *Daily Scrum*, para coordenar as atividades que estão sendo realizadas (ZANATTA, 2004).

Caso seja constatado que o *Sprint* será inviável, o *Scrum Master* pode encerrar e iniciar um novo planejamento. No caso que a equipe não consiga completar os itens do *Sprint Backlog*, ela pode consultar o *Product Owner* para eliminar alguns itens do *Sprint Backlog* ou

se a equipe terminar o *Sprint Backlog* antes do prazo, ela pode consultar o *Product Owner* para acrescentar mais itens ao desenvolvimento.

Os itens a seguir descrevem cada fase do *Sprint*, conhecidos como: *Sprint Planning Meeting*, *Daily Scrum*, *Sprint Review Meeting* e *Sprint Retrospective.*

3.6.2.1 Sprint Planning Meeting

É a primeira parte do *Sprint*, sendo divido em duas partes de quatro horas, assim evitando discussões e anotações do que pode ser feito a mais. Segundo Martins (2007, p. 263) “o objetivo é definir o que vai ser trabalhado e não como vai ser feito”.

Conforme Martins (2007) a primeira parte do *Sprint* se iniciam com uma reunião de planejamento que envolve o *Product Owner*, o *Scrum Master* e a equipe para definir o que será trabalhado na próxima iteração.

O *Product Owner* seleciona os itens do *Product Backlog* detalhando-o a equipe até ela obter informações suficientes para desenvolver e para definir até qual item acredita completar no Sprint, para isso são aplicadas estimativas.

Na segunda parte é elaborado *Sprint Backlog*, que é um planejamento que a equipe faz do *Sprint*.

3.6.2.2 Sprint Backlog

É o ponto inicial da execução das tarefas, onde a lista das tarefas definida no *Sprint Planning Meeting* define o trabalho do equipe durante o *Sprint*.

Os itens *Product Backlog* selecionados se divididos em requisitos menores que podem ser cumpridas em um curto prazo de tempo, assim formando os itens do *Sprint Backlog*.

Tem as seguintes características:

a) cada tarefa tem definido um responsável que irá executar ela;

b) tarefas devem estar organizadas para que sejam finalizadas de uma dois dias de trabalho;

c) não se podem incluir novas atividades durante o andamento do *Sprint*.

De acordo com Martins (2007) o *Sprint Backlog* é elaborado numa planilha semelhante ao *Product Backlog*, mas possuindo colunas adicionais, uma para cada dia do *Sprint* com a finalidade de apontar quantas horas resta para finalizar o item do *Sprint Backlog*, como é representado na Figura 3.

Backlog da Iteração
Super Monitor

Qtde horas do dia 8

	Seg	Ter	Qua	Qui	Sex
Dia da iteração	23/jan	24/jan	25/jan	26/jan	27/jan
Horas pendentes	214	195	204	209	209

Id	Categoria	Item	Prior.	Resp.	Dias est.	Horas Est.	1	2	3	4	5
15	Filas/backup	Mudar arquitetura para trabalhar com filas de chamadas para processamento	1	AG	6,67	53	53	40	30	35	35
18	Watchdog	Criar mecanismo de watchdog e sistema de alarmer por e-mail	1	AG	6,67	53	47	47	47	47	47
19	Web	Migrar telas para ASP + IIS	1	AC	6,67	53	40	35	35	35	35
3	Segurança	Logar as pesquisas de gravação feitas pelos usuário e as reproduções	2	AC	0,83	7	7	7	25	25	25
8	Playback	Fazer reprodução através de playlist	2	AG	3,33	27	27	27	27	27	27
13	Segurança	Criptografar DVD com as gravações	2	AC	3,33	27	27	27	27	27	27
9	Playback	Enviar gravação por e-mail	3	AC	1,67	13	13	13	13	13	13

Figura 3. *Sprint Backlog*
Fonte: Martins, J. (2007)

Além de a planilha mostrar quantidade de horas restantes para a finalização dos requisitos, ela fornece a possibilidade de apresentar o gráfico de tendência ou B*urndown Chart* do *Sprint* (MARTINS, 2007).

Neste processo de desenvolvimento para ter um acompanhamento mais simples dos requisitos que estão sendo desenvolvidos, pode ser utilizado um painel conhecido como *Taskboard*.

Nos itens a seguir são descritos o gráfico *Burndown Chart* e o painel *Taskboard*.

3.6.2.2.1. Gráfico Burndown Chart

De acordo com Isotton Neto (2008) o gráfico *Burndown Chart* tem como principal característica representar o restante de horas para finalizar o *Sprint*. Os valores que dão origem ao gráfico vêm do *Sprint Backlog*.

O gráfico é formado pelo total de horas, que é a parte vertical, o total de dias que é a parte horizontal, e a linha de tendência que representa o total de horas trabalhadas nos requisitos selecionados, de forma decrescente, como pode ser representado na Figura 4.

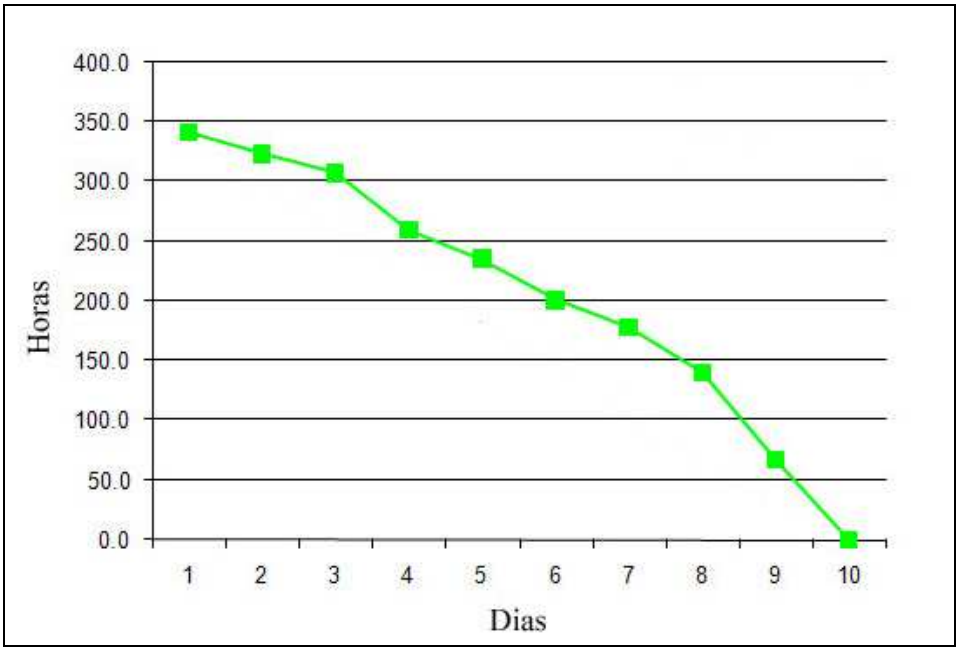


Figura 4. Gráfico *Burndown Chart*

Com o gráfico se torna mais fácil de apontar como está o processo do *Sprint*, como por exemplo, descobrir se a equipe conseguira finalizar os requisitos no prazo, ou se está rápida de mais.

3.6.2.2.2 Taskboard

É um painel onde são colocadas as informações para o acompanhamento e controlamento do *Sprint Backlog*, sendo conhecida também como um técnica denominada *Kambam*.

O *Taskboard* é montado a cada *Sprint*, e a idéia é que ele esteja visível a cada membro da equipe durante o período de trabalho.

A Figura 5 representa o painel *Taskborad*, sendo constituído por quatro colunas, e pelos itens que são conhecidos como *post-it's*.

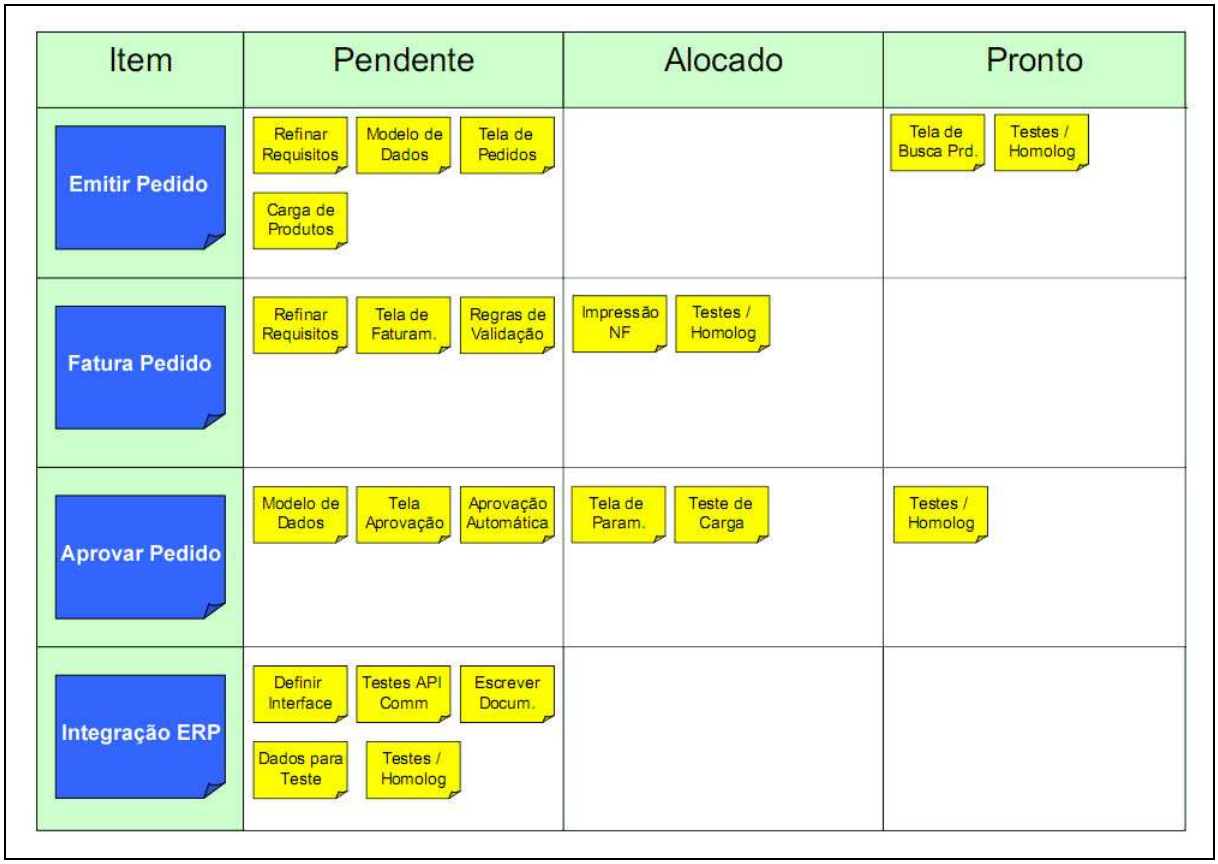


Figura 5. Painel *Taskboard*
Fonte: Adaptado de Yoshima (2007)

Conforme Yoshima (2007) a primeira coluna denominada itens, contem os itens do *Sprint Backlog* (quadros azuis) que serão desenvolvidos, esses itens são subdivididos em

partes menores (quadros amarelos) com duração de um dia no máximo e são colocados na coluna de pendente.

Na coluna alocado ficam os itens que estão sendo desenvolvidos pela equipe e posteriormente na coluna pronto ficam os itens finalizados.

Para um *post-it* ficar na coluna pronto, é necessário que esteja totalmente pronto, testado, homologado e aprovado pelo *Product Owner*. O Scrum não aceita itens parcialmente prontos.

Como os itens estão ordenados de acordo com a prioridade do *Product Backlog*, é importante que os itens no topo da lista sejam resolvidos primeiro.

3.6.2.3 Daily Scrum

É o nome dado para as reuniões que acontecem no *Sprint* diariamente ou em dias alternados, com duração de quinze a trinta minutos, onde os membros devem ficar de pé. A reunião se possibilita a identificação dos obstáculos e impedimentos do projeto.

Conforme Schwaber (2004, nossa tradução) na reunião são levantados três pontos importantes a cada membro da equipe:

a) o que foi finalizado desde a última reunião? O *Scrum Master* registra as tarefas completadas e quais ainda estão pendentes;
b) quais são as dificuldades e impedimentos para a conclusão da tarefa? O *Scrum Master* registra esses obstáculos encontrados para posteriormente encontrar soluções para resolvê-las;
c) quais tarefas específicas que cada membro deseja finalizar até a próxima reunião? O *Scrum Masters* ajuda os membros da equipe a selecionar as tarefas de maior prioridade.

A reunião realizada tem como um dos objetivos retirar os itens concluídos do *Product Backlog*. E também possibilita que todas as pessoas fiquem informadas sobre o progresso e as dificuldades encontrados.

3.6.2.4 Sprint Review Meeting

É o último dia do *Sprint*, é o dia que a equipe e o *Scrum Master* apresentam os resultados ao *Product Owner*, que posteriormente, analisam os resultados e decidem sobre as novas atividades que poderão se integrar ao *Product Backlog* (ZANATTA, 2004).

Na reunião todos os membros visualizam as funcionalidades que foram implementadas ao decorrer do *Sprint*, e definem as funcionalidades que podem ser adicionadas ao próximo *Sprint*.

Segundo Schwaber (2004, nossa tradução) o *Sprint Review Meeting* é representada nas seguintes características:

a) a reunião não pode ultrapassar de quatro horas;
b) a equipe ao se preparar para reunião não deve passar de uma hora;
c) a equipe deve apresentar as funcionalidades completamente feitas para o *Product Owner* e aos *stakeholders*;
d) não deve ser apresentada as funcionalidades que estão incompletas, ou seja, não estejam totalmente feitas ainda;
e) as funcionalidades devem ser apresentadas num ambiente semelhante ao de homologação;
f) a reunião tem início com a apresentação do *Sprint*, os itens do *Product Backlog* comprometidos e os que foram completados;
g) a prioridade da reunião é a equipe apresentar as funcionalidades, respondendo as perguntas sobre as funcionalidades e tomar nota das mudanças solicitadas;

h) no final da apresentação da funcionalidades, os *stakeholders* são questionadas quanto às suas impressões, mudanças necessárias e alterações de prioridades;

i) o *Product Owner* pode identificar funcionalidades que não foram entregues conforme o esperado, assim solicitando que sejam incluídas no próximo *Sprint* com prioridade alta;

j) o *Scrum Master* avalia quantas pessoas irão participar da reunião para organizar o ambiente de forma que fique confortável;

k) no final da reunião o *Scrum Master* agenda a próxima reunião junto com o *Product Owner* e os *stakeholders*.

3.6.2.5 Sprint Retrospective

Essa reunião é realizada após o *Sprint Review Meeting* que tem como objetivo encorajar a equipe a revisar o seu processo de trabalho, tendo em vista as práticas do Scrum, para ter um desempenho melhor na próxima iteração (MARTINS, 2007).

Segundo Schwaber (2004, nossa tradução) está reunião é composta pelas seguintes características:

a) a reunião deve ser realizada em um período de três horas;

b) o *Product Owner*, *Scrum Master* e a equipe devem estar presentes nessa reunião;

c) a reunião iniciasse com as membro identificando o que deu certo no *Sprint* e o que pode ser melhorado no próximo;

d) o *Scrum Master* anota as respostas de cada membro;

e) o *Scrum Master* participa da reunião para ajudar a equipe a descobrir novas formas de trabalhar dentro do processo de Scrum;

f) a equipe planeja soluções para problemas irritantes.

4 METRICAS DE SOFTWARE

De acordo com Pressman (2006) a mensuração é aplicada no processo de desenvolvimento de software ou atributos de um produto com o objetivo de melhorá-lo de forma continua que utilizando ao longo do projeto auxilia na estimativa, no controle de qualidade, na avaliação da produtividade e no controle do projeto.

Com as métricas os gerentes conseguem informações quantitativas quem auxiliam na tomada de decisões e para obter uma visão melhor do trabalho que está sendo realizado, podendo assim desenvolver um software mais confiável (INTHURN, 2001).

É possível também obter um entendimento imediato ao invés de posterior, caso estiver ocorrendo algum problema, é possível corrigir antes que esse problema se agrave mais (PRESMAN, 2006).

Basicamente a medição abrange a coleta dos dados para a computação das métricas e por fim, a avaliação dos resultados obtidos.

De acordo com Pressman (1995) as métricas oferecem os seguintes benefícios, como:

a) maior controle do processo de desenvolvimento;
b) possibilidade de criar estimativas mais precisas;
c) redução de custos e tempo de desenvolvimento;
d) aumento da satisfação e confiança do cliente, devido à maior qualidade do software;
e) melhor produtividade da equipe de desenvolvimento da organização.

Com os diversos benefícios que as métricas proporcionam, existem dificuldades em sobre o que medir, como decidir qual o método mais apropriado ou em questionar a validade de comparações envolvendo pessoas, produto, processos (NUERNBERG, 2007).

Segundo Nuernberg (2007) cabe a empresa de desenvolvimento ter a responsabilidade de buscar o conhecimento do objetivo das métricas para avaliar quais delas proporcionam os benefícios para melhorar continuamente o processo de desenvolvimento de software.

Ao se aplicar métricas é necessário ter em mente os seguintes conceitos:

a) medida: fornece uma indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo de um produto ou processo. Um exemplo seria o número de linhas de código;

b) métricas: uma métrica é um método para determinar se um sistema, componente ou processo possui um certo atributo. Ela é geralmente calculada ou composta por duas ou mais medidas. Um exemplo de métrica seria o número médio de erros encontrados por revisão ou por implantação;

c) indicadores: é uma métrica ou combinação de métricas que fornece uma compreensão da visão do processo do software, projeto do software, produto em si ou a entidade que está sendo medida. Um indicador permite ajustar o produto, o projeto ou o processo para melhorá-los.

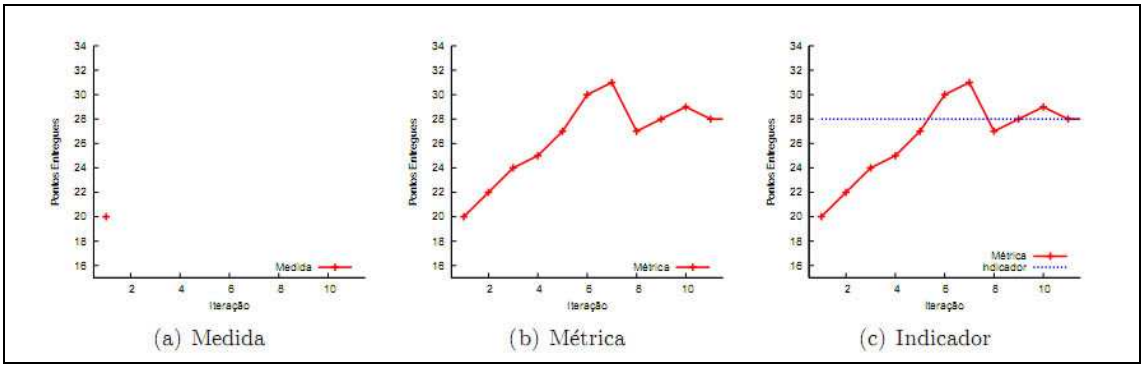


Figura 6. Representação de medida, métrica e indicador
Fonte: Sato, 2007

Com a Figura 6 pode se utilizar como um exemplo mais completo para esclarecer medida, métrica e indicador, onde no item (a) mostra medida dos pontos de entrega na

primeira iteração, no item (b) mostra as medidas dos pontos entregue em diversas iterações, que acaba se tornando uma métrica, e por último, no item (c), a uma linha horizontal pontilhada que representa o indicador, que pode ser utilizado para avalizar se a equipe conseguiu superar os pontos de entrega por iteração.

As métricas podem ser utilizadas no processo de desenvolvimento ou no projeto, onde existem dois contextos de medição que pode se atingir.

As métricas de projeto segundo Pressman (2006) tem os objetivos de:

a) avaliar o estado de qualidade de um projeto durante o desenvolvimento;
b) minimizar os riscos;
c) descobrir problemas antes que eles se tornem mais críticos;
d) ajustar o fluxo de trabalho ou de tarefas;
e) avaliar a capacidade da equipe de projeto de controlar a qualidade dos produtos.

As métricas de processo têm objetivos de fornecer um conjunto de indicadores que são coletados ao decorrer de outros diversos projetos, assim utilizados para levar o aperfeiçoamento e nível de maturidade ao processo.

Sendo analisados os seguintes itens:

a) erros descobertos antes da entrega do produto;
b) defeitos entregues aos usuários finais;
c) produtos de trabalho entregues;
d) tempo gasto;
e) cumprimento de cronograma.

Métricas de projeto avaliam a qualidade do produto em si, enquanto as métricas de processo avaliam a produtividade do software.

Ambos contextos são essências para o projeto, pois a falta ou a má aplicação podem ocasionar atrasos, falhas e perdas de qualidade no projeto de desenvolvimento de software.

Com a aplicação de estimativas é possível coletar métricas que permitam prever a quantidade de pessoas necessárias, o tempo necessário e o os custos para o desenvolvimento do projeto. "Assim, torna-se importante o investimento na implantação de um processo de estimativas" (HAZAN, 2008, p. 25).

O item a seguir descreve a importância da aplicação de estimativas.

4.1 MÉTRICAS DE ESTIMATIVA

Estimar um projeto é necessário ter experiência e dados históricos de outros projetos concluídos, assim, podendo aplicar métricas de software para encontrar o tamanho do projeto.

Segundo Nuemberg (2007, p. 24) "as estimativas de tamanho são consideradas a base principal para determinar o prazo, esforços e custos adequados".

A estimativa geralmente é realizada por um engenheiro de software ou especialista que analisa os requisitos garantindo a qualidade, e então estima o tamanho do projeto de software. Após a definir o tamanho, é realizada a derivação para determinar o esforço, prazo (cronograma) e custo (orçamento). Caso ocorram mudanças nos requisitos é necessário re-estimar (HAZAN, 2008).

A cada projeto finalizado se torna importante a documentação das estimativas e de outros atributos relevantes. Essa documentação pode ser utilizada como dados históricos para outros projetos, assim garantido melhorias do processo de estimativa dos próximo projetos.

Para estimar existem várias técnicas e métodos, como: *Ideal Day*, *Planning Poker* e Ponto de função que são descritas nos itens a seguir.

4.2 IDEAL DAY

É utilizado para realizar estimativas de forma ágil, sendo aplicada para planejar o projeto e iterações.

Segundo F. Alves, M. Alves e Fonseca (2008) *Ideal Day* corresponde à quantidade de trabalho que um profissional da área consegue concluir em um dia de trabalho, levando em consideração os seguintes aspectos e interrupções:

a) natureza humana do desenvolvedor (comer, beber, alongar, socializar, sono, mal-estar eventual, etc);
b) deficiências técnicas do desenvolvedor (desconhecimentos do assunto ou tecnologia específicos);
c) interrupções da empresa (reuniões administrativas, conversa com o chefe, ligações de clientes);
d) interrupções pessoais.

A equipe deve ter sua velocidade medida pelo tempo gasto para se implementar um *Ideal Day*. Quanto menor o tempo será maior a velocidade e a produtividade do mesmo. O importante não medir a velocidade individual, mas sim a média da equipe (ALVES, F.; ALVEZ, M.; FONSECA, 2008).

De acordo com Martins (2007) velocidade é calculada a partir do número de horas que a equipe gasta para implementar um trabalho equivalente a um *Ideal Day*. Caso o item passe um *Ideal Day* é sugerido que decompõe esse item e itens menores que se consiga implementar em apenas um *Ideal Day*.

O cálculo dos dias estimados utiliza a seguinte fórmula:

$$DE = \frac{IED}{1 - IED\_REAL\%}$$

Onde:

DE: quantidade de dias estimado para concluir a tarefa;

IED: prazo necessário para implementar o item, esse prazo é definido pela equipe;

IED_REAL%: percentual que indica a estimativa de quanto tempo do dia o desenvolvedor ficara dedicado a implantação do item.

4.3 PLANNING POKER

O *Planning Poker* é uma ferramenta simples, pois é semelhante a um jogo de cartas, mas considerada poderosa para dar estimativas rápidas, precisas e de um modo que a equipe não precise se esforçar.

O nome foi colocado por James Grenning em 2002 e popularizado por Mike Cohn no seu livro *Agile Estimating and Planning*.

Os participantes no *Planning Poker* incluem todos os programadores, testadores, engenheiros, Analistas e designers, ou seja, a equipe inteira.

No *Planning Poker* para estimar é necessário do *Product Backlog* e de um baralho de cartas, onde as cartas devem ter os seguintes números 0, 1, 2, 3, 5, 8, 13, 20, 40 e 100, ou seja, semelhante à seqüência de *Fibonacci* (PlanningPoker, 2009).

Cada membro da equipe deve possuir uma seqüência de cartas, pois cada item do *Product Backlog* corresponde a um valor definido através de rodadas entre os membros, semelhante a um jogo de cartas.

A cada funcionalidade do *Product Backlog*, os membros relacionam uma carta com o valor que acham ser o ideal. Após todos os membros jogarem, é discutido qual o valor ideal para aquele item, caso a equipe não chegue a um consenso, e feito mais rodadas até a equipe chegar a um consenso, e assim, continua com os outros itens do *Product Backlog*.

De acordo com *Planning Poker* (2009), essa métrica pode oferecer os seguintes benefícios por se utilizado:

a) para estimar, são reunidos todos os especialistas que vão desenvolver, assim se torna mais preciso o cálculo;
b) os membros são chamados para estimar e assim justificando as suas estimativas. Dessa forma melhora a precisão da estimativa, principalmente nos itens com grandes quantidades de tarefas, onde muitas vezes a equipe entra com o papel de ajudar e compensar a falta de informação para estimar;
c) ter transparência das diferentes opiniões, assim os membros podem discutir as diferenças.

Com *Planning Poker* pode ser definido que diminui bastante os erros da definição das estimativas, pois as discussões realizadas para estimar têm maior valor a analise dos requisitos.

4.4 PONTOS DE FUNÇÃO

Conforme Dekkers (1998) Pontos de Função (PF) medem o tamanho funcional do software, onde o tamanho funcional pode ser definido como uma medida de tamanho de software, essa medida é realizada a partir da funcionalidade implementada em um sistema sob o ponto de vista do usuário.

Segundo Cabral (2006) análise de PF permite ser utilizados em todos os tipos de projetos e organizações, pois é independente dos métodos de desenvolvimento, linguagens de programação utilizada no desenvolvimento do sistema, ferramentas e do estilo do programador.

No final da definição de requisitos do software, já pode ser utilizado PF, pois os dados já estão mais claros, conhecidos e interpretados (NUEMBERG, 2007).

4.4.1 Origem

Segundo Dekkers (1998) PF foi introduzida em 1979 por Allan Albrecht da empresa International Business Machines (IBM). Posteriormente os conceitos foram refinados em uma metodologia formal e publicados no domínio público em 1984. Em 1986 teve padronizações adicionais nas regras de contagem de PF pelo *International Function Point Users Group* (IFPUG).

Em 2002 PF passou a ser um padrão internacional, por meio da norma *International Organization of Standardization* / *International Electrotechnical Commission* (ISO/IEC) 20926. No Brasil PF é representada pelo *Brazilian Function Point Users Group* (BFPUG).

4.4.2 Benefícios

Segundo Hazan (2001) PF podem ser utilizados da seguinte maneira nas organizações:

a) ferramenta para determinar o tamanho funcional do software;
b) ferramenta para apoiar o processo de medição de qualidade e produtividade;
c) mecanismo para derivação de esforço, prazo, custo e recursos envolvidos em projetos;
d) como medida estável para comparação de projetos.

Segundo Cabral (2006) PF além de medir o tamanho funcional do software, oferecem os seguintes benefícios:

a) melhor acompanhamento na produtividade da equipe;
b) melhoria no controle da qualidade e os custos de desenvolvimento;

c) medição da satisfação do usuário em relação às soluções desenvolvidas e o processo empregado;

d) melhor previsão do projeto por possibilitar realizações de estimativas nas fases iniciais do desenvolvimento de software;

e) justificativa a necessidade de recursos e de pessoal;

f) possibilidade de refazer estimativas;

g) melhora o processo de desenvolvimento e manutenção, através da análise de pontos fortes e fracos e custos das falhas;

h) Pode ser utilizada como fator de normalização de preço, baseando-se na norma da IFPUG.

Com os diversos benefícios oferecidos pelos PF se pode gerenciar o projeto de uma forma mais adequada.

4.4.3 Etapas de Contagem

O fluxograma da Figura 7 apresenta a ordem dos procedimentos de PF:

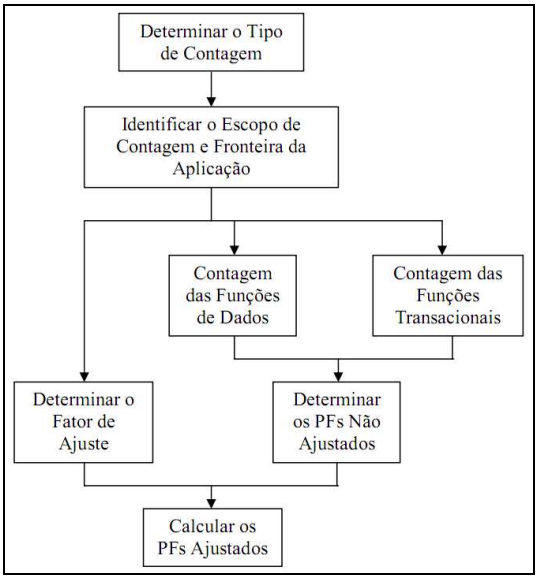


Figura 7. Procedimento de Contagem de PF.
Fonte: HAZAN, 2001.

A sete passos contidos no procedimento de contagem de PF como representado na Figura 7, são as seguintes: tipo de contagem, identificar o escopo da contagem e a fronteira da aplicação, contagem das funções de dados, contagem das funções tipo transação, calcular contagem de pontos de função não ajustados (PFNA), calcular valor do fator de ajuste (VFA) e calcular os pontos por função ajustados (PFA) onde são representados no itens a seguir.

4.4.3.1 Tipo de Contagem

Segundo Vazquez, Simões e Albert (2007) existem três tipos de contagens de PF:

a) projeto de desenvolvimento: conta as funcionalidades fornecidas aos usuários finais ao decorrer do desenvolvimento do software;
b) projeto de manutenção: conta as modificações do sistema, podendo ser a funcionalidades adicionadas, alteradas e excluídas no software já existente;
c) aplicação: fornece uma medida das funcionalidades do software já concluído. Essa contagem fornece uma medida da atual funcionalidade obtida pelo usuário da aplicação.

4.4.3.2 Identificar a Fronteira da Aplicação e o Escopo da Contagem

Fronteira da aplicação é a interface conceitual entre o software que será medido e o mundo exterior (seus usuários).

O escopo da contagem define as funcionalidades que serão incluídas em uma contagem de pontos de função, podendo abranger um ou mais sistemas ou apenas parte de um sistema (VAZQUEZ; SIMÕES; ALBERT, 2007).

4.4.3.3 Contagem das funções de dados

Segundo Vazquez, Simões e Albert (2007) as funções de dados representam as funcionalidades pelo sistema aos usuários, para atender as necessidades.

São classificadas em:

a) arquivos lógicos internos (ALI): segunda Guerra (2006) são os grupos lógicos de dados relacionados ou informações de controle identificado pelo usuário mantido dentro da fronteira da aplicação. Podem ser banco de dados lógico da aplicação (arquivos e tabelas mantidos dentro da fronteira da aplicação) (HAZAN, 2001);

b) arquivos de interface externa (AIE): A intenção fundamental de um AIE é manter dados referenciados através de um ou mais processos elementares da aplicação que está sendo contada. Isso significa que um AIE deve obrigatoriamente ser um AIE de outra aplicação (VAZQUEZ; SIMÕES; ALBERT, 2007). Podem ser banco de dados de outras aplicações, onde são apenas referenciados pela aplicação que está sendo estimada, ou seja, apenas lê as informações gravadas por outra aplicação (HAZAN, 2001).

A contagem da função de dados deve ser classificada com relação à sua complexidade funcional (simples, média, complexa) em que os elementos devem ser contabilizados como registros lógicos referenciados (RLR) e dados elementares relacionados (DER).

Um RLR é um subgrupo de dados reconhecido pelo usuário, dentro de um ALI ou AIE, o DER é um campo único não-repetido e reconhecido pelo usuário.

A identificação do número de funções da complexidade de ALI e AIE, utiliza-se como base a Tabela 2.

Tabela 2. Complexidade da Funcional ALI e AIE

Número de Registros Lógicos	**Número de Itens de Dados Referenciados**		
	De 1 a 19	De 20 a 50	51 ou mais
Apenas 1	Simples	Simples	Média
De 2 a 5	Simples	Média	Complexa
6 ou mais	Média	Complexa	Complexa

Fonte: IFPUG (2001).

Após a identificação da complexidade dos arquivos, deve-se calcular sua contribuição utilizando a Tabela 3.

Tabela 3. Contagem de PF

Tipo de Função	**Grau de Complexidade**		
	Simples	Média	Complexa
ALI	7 PF	10 PF	15 PF
AIE	5 PF	7 PF	10 PF

Fonte: IFPUG (2001).

Em casos que não é possível identificar a complexidade da função de dados, recomenda-se a utilização da complexidade simples (HAZAN, 2001).

4.4.3.4 Contagem das Funções Tipo Transação

De acordo com Vazquez, Simões e Albert (2007) as funções transacionais representam as funcionalidades fornecidas ao usuário para atender suas necessidades de processamento de dados pela aplicação.

Os tipos de transação são classificados em três categorias:

a) entradas externas (EE): é um processo elementar de dados ou informações de controle recebidos de fora da fronteira da aplicação, com o objetivo de manter (incluir, alterar ou excluir dados de) um ou mais ALI e/ou modificar o comportamento do sistema;

b) saídas externas (SE): é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação, cujo objetivo é apresentar informação ao usuário por meio de lógica de processamento. Essa lógica deve conter no mínimo uma fórmula matemática ou cálculo ou criar dados derivados. Podem-se citar como exemplo os relatórios impressos ou mostrados na tela com totalizadores;

c) consulta externas (CE): é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação, com o objetivo de apresentar informações ao usuário por meio de uma simples recuperação de dados ou informações de controle de ALIs ou AIEs.

O cálculo da complexidade se baseia no número de arquivos referenciados (AR) e de tipos de dados (TD), onde cada EE, SE e CE devem ser classificados com relação à sua complexidade funcional (simples, média, complexa). Para a identificação do número de funções da complexidade, utilizando-se como base a Tabela 4 para EE e a Tabela 5 para SE e CE.

Tabela 4. Complexidade das EE

Número de Registros Lógicos	**Número de Itens de Dados Referenciados**		
	De 1 a 4	De 5 a 15	16 ou mais
0 a 1	Simples	Simples	Média
2	Simples	Média	Complexa
3 ou mais	Média	Complexa	Complexa

Fonte: IFPUG (2001).

Tabela 5. Complexidade das SE e CE

Número de Registros Lógicos	**Número de Itens de Dados Referenciados**		
	De 1 a 4	De 5 a 15	16 ou mais
0 a 1	Simples	Simples	Média
2 a 3	Simples	Média	Complexa
4 ou mais	Média	Complexa	Complexa

Fonte: IFPUG (2001).

Após a determinação da complexidade das funções do tipo de transação, deve-se realizar a multiplicação do número de funções simples, médias e complexas pelos valores referenciados da Tabela 6.

Tabela 6. Contagem de Pontos por Função

Tipo de Função	**Grau de Complexidade**		
	Simples	Média	Complexa
EE	3 PF	4 PF	6 PF
SE	4 PF	5 PF	7 PF
CE	3 PF	4 PF	7 PF

Fonte: IFPUG (2001).

Em casos que não é possível identificar a complexidade das funcionalidades, recomenda-se a utilização da complexidade média (HAZAN, 2001).

4.4.3.5 Calcular contagem de Pontos de Função Não Ajustados

Quando finalizada a contagem dos ALI, AIE, EE, SE e CE, então é feito a contagem dos PFNA. Segundo Nuemberg (2007) os PFNA fornecem uma unidade de produção bruta, a contagem se baseia na seguinte fórmula:

$$PFNA = ALI + AIE + EE + SE + CE$$

Onde PFNA é o resultado da soma dos números de funções identificados de cada complexidade (Simples, Média e Complexa) existente no sistema pelo seu peso de complexidade ser visto no capitulo anterior.

4.4.3.6 Calcular Valor do Fator de Ajuste

O VFA é baseado em quatorze características gerais de sistema, cada uma dessas características possui um grau de influência sobre a aplicação que pode variar em um intervalo de zero a cinto com visto na Tabela 7.

Tabela 7. Grau de Influência

Grau	Descrição
0	Nenhuma influência
1	Influência mínima
2	Influência moderada
3	Influência média
4	Influência significativa
5	Influência forte

Fonte: IFPUG (2001).

O fato de ajuste quando aplicado aos PFNA, pode influenciar numa variação de +/- 35% no resultado final dos PF.

O cálculo do VFA utiliza a seguinte fórmula:

$$VFA = (TDI \times 0{,}01) + 0{,}65$$

Onde TDI é o somatório dos níveis de influência das características gerais.

As quatorzes características gerais de sistema que são citados abaixo conforme Hazan (2001) e Vazquez, Simões e Albert (2007).

4.4.3.6.1 Comunicação de Dados

São utilizados pela aplicação para enviar ou receber dados ou informações de controle por meio de recursos de comunicação. Descrever se a aplicação utiliza protocolos diferentes para recebimento e envio das informações do sistema.

A pontuação é realizada de acordo com as seguintes orientações:

0. aplicação é somente *batch* ou uma estação de trabalho isolada;
1. aplicação é somente em *batch*, mas utiliza uma entrada de dados ou impressão remota;
2. aplicação é *batch*, mas utiliza entrada de dados e impressão remota;
3. aplicação com entrada de dados on-line ou *front-end* de teleprocessamento para alimentar processamento batch ou sistema de consulta;
4. aplicação com entrada de dados *on-line*, mas com suporta apenas para um tipo de protocolo de comunicação;
5. aplicação com entrada de dados *on-line* e com suporta a mais de um tipo de protocolo de comunicação.

4.4.3.6.2 Processamento Distribuído

Descreve se o sistema utiliza dados ou processamento distribuído entre seus componentes (processadores).

A pontuação é realizada de acordo com as seguintes orientações:

0. a aplicação não participa na transferência de dados ou funções entre os processadores da empresa;
1. a aplicação prepara dados para processamento pelo usuário final em outros componentes (ex. planilhas eletrônicas);
2. dados são preparados para transferências, após são transferidos e processados em outro processador (não o usuário final);
3. processamento é distribuído e a transferência de dados é realizada on-line e apenas em uma direção;

4. processamento é distribuído e a transferência de dados é realizada on-line e em ambas as direções;
5. as funções de processamento são executadas dinamicamente no componente mais apropriada.

4.4.3.6.3 Desempenho

Descreve se os requisitos de desempenho (tempo de resposta e taxa de transações) estabelecidos pelo usuário influenciam o projeto, desenvolvimento, implantação e suporte do sistema.

A pontuação é realizada de acordo com as seguintes orientações:

0. o usuário não estabeleceu nenhum requisito especial sobre desempenho;
1. requisitos de desempenho e projeto foram estabelecidos e revistos, mas nenhuma ação em especial foi requerida;
2. tempo de resposta ou volume de transações são críticos durante as horas de picos. Não é necessário nenhum projeto especial para a utilização do processamento. O limite para a disponibilidade de processamento é sempre o próximo dia útil;
3. tempo de resposta ou volumes de transações são críticos durante todas as horas de trabalho. Não é necessário nenhum projeto especial para a utilização de processamento. O limite de processamento é crítico;
4. os requisitos de desempenho estabelecidos pelo usuário são rigorosos o bastante para requerer tarefas de análise e projeto da aplicação;
5. além do anterior descrito, é necessário o uso de ferramentas de análise de desempenho são usadas nas fases de planejamento, desenvolvimento e/ou

implementação para garantir a obtenção dos requisitos de desempenho estabelecidos pelos usuários.

4.4.3.6.4 Configuração Altamente Utilizada

Descreve o nível de utilização dos recursos computacionais requeridos para o desenvolvimento da aplicação.

A pontuação é realizada de acordo com as seguintes orientações:

0. não existem restrições operacionais nos requisitos;
1. existem restrições operacionais consideradas leves da aplicação. Mas sem a necessidade de esforço extra para ao atendimento dessas restrições;
2. existem restrições operacionais, mas são consideradas típicas da aplicação. Sendo necessário um esforço extra para o atendimento dessas restrições;
3. existem necessidades específicas de processador para uma parte especifica da aplicação;
4. restrições operacionais necessitam de um processador central ou no processador dedicado para executar a aplicação;
5. além das características do item anterior, há considerações especiais que exigem utilização de ferramentas de análise de desempenho para a distribuição do sistema e seus componentes nas unidades processadoras.

4.4.3.6.5 Volume de Transações

Descreve em que nível o volume de transações influência o projeto, desenvolvimento, implantação e manutenção do sistema.

A pontuação é realizada de acordo com as seguintes orientações:

0. nenhum período de picos de volume de transações é esperado;
1. são previstos picos de transações mensalmente, trimestralmente, anualmente ou em certo período do ano, mas tendo um impacto mínimo no projeto;
2. são previstos picos semanais, tendo alguns impactos no projeto;
3. são previstos picos diários com impactos significativos no projeto;
4. altas taxas de transações são estabelecidas pelo usuário, ou os níveis de serviço são alto o suficiente para requerer análise de desempenho na fase de projeto;
5. além do item anterior, é necessário utilizar ferramentas de análise de desempenho nas fases de projeto, desenvolvimento e/ou implantação.

4.4.3.6.6 Entrada de Dados On-line

Descrevem em que nível são efetuadas entradas de dados no modo *on-line* no sistema por meio de transações interativas.

A pontuação é realizada de acordo com as seguintes orientações:

0. todas as transações são processadas em lote;
1. de 1% a 7% das transações são entradas de dados *on-line*;
2. de 8% a 15% das transações são entradas de dados *on-line;*
3. de 16% a 23% das transações são entradas de dados *on-line*;
4. de 24% a 30% das transações são entradas de dados *on-line*;
5. mais de 30% das transações são entradas de dados *on-line*.

4.4.3.6.7 Eficiência do Usuário Final

Descreve o grau de influência relativo aos recursos implementados com vista a tornar o sistema amigável, permitindo incrementos na eficiência e satisfação do usuário final, tais como:

a) auxílio à navegação (Ex.: teclas de função, acesso direto e menus dinâmicos);

b) menus;

c) documentação e ajuda *on-line*;

d) movimento automático do cursor;

e) paginação;

f) impressão remota por meio de transações on-line;

g) teclas de função predefinidas;

h) tarefas em lote submetidas a transações on-line;

i) utilização intensa de vídeo reverso, sublinhados, coloridos e outros indicadores;

j) impressão da documentação das transações;

k) utilização de mouse;

l) janelas *pop-up*;

m) menor número possível de telas para executar as funções de negócio;

n) suporte a dois idiomas (contar como quatro itens);

o) suporte a mais de dois idiomas (contar como seis itens).

A pontuação é realizada de acordo com as seguintes orientações:

0. nenhum dos itens descritos;

1. de um a três itens descritos;

2. de quatro a cinco dos itens descritos;

3. mais de cinco dos itens descritos, mas não há requisitos específicos do usuário quanto à usabilidade do sistema;
4. seis ou mais dos itens descritos, onde foram estabelecidos requisitos quanto à eficiência para o usuário final, sendo forte o suficiente para gerarem atividades específicas envolvendo fatores, tais como minimização da digitação, para mostrar inicialmente os valores utilizados com mais freqüência;
5. seis ou mais dos itens descritos e foram estabelecidos requisitos quanto à eficiência para o usuário final forte o suficiente para requerer ferramentas e processos especiais para demonstrar antecipadamente que os objetivos foram alcançados.

4.4.3.6.8 Atualização On-line

Descreve em que grau os ALI do sistema são atualizados *on-line*.

A pontuação é realizada de acordo com as seguintes orientações:

0. não há atualização *on-line*;
1. existe atualização *on-line* de um a três ALI. O volume de atualização é baixo e a recuperação de dados é simples;
2. existe atualização *on-line* de mais de três ALI. O volume de atualização é baixo e a recuperação dos dados é simples;
3. atualização *on-line* da maioria ALI;
4. em adição ao item anterior, é essencial a proteção contra perdas de dados que foi projetada e programada no sistema;
5. além da descrição item anterior, altos volumes trazem considerações de custo no processo de recuperação. São incluídos procedimentos para automatizar a recuperação para a intervenção do operador.

4.4.3.6.9 Complexidade de Processamento

Descreve o grau de influência que o processamento lógico ou matemático influência o desenvolvimento da aplicação, com base nas seguintes categorias:

a) processamento especial de auditoria e/ou processamento especial de segurança;

b) processamento lógico extensivo (Ex.: sistema de gestão de crédito);

c) processamento matemático extensivo (Ex.: sistemas de otimização de corte de tecidos);

d) processamento gerando muitas exceções, resultando em transações incompletas que devem ser processadas novamente. (Ex.: transações de auto-atendimento bancário interrompidas por problemas de comunicação ou com dados incompletos);

e) processamento complexo para manipular múltiplas possibilidades de entrada e saída (Ex.: sistema de extrato de conta corrente que emite via terminal de retaguarda).

A pontuação é realizada de acordo com as seguintes orientações:

0. nenhum dos itens descritos;
1. apenas um dos itens descritos;
2. dois dos itens descritos;
3. três dos itens descritos;
4. quatro dos itens descritos;
5. todos os cinco itens descritos.

4.4.3.6.10 Reusabilidade

Descreve em que nível o sistema e seu código gerado foram especificamente projetados, desenvolvidos e suportados para serem reutilizados em outros sistemas.

A pontuação é realizada de acordo com as seguintes orientações:

0. não há código reutilizado;
1. código reutilizado é utilizado somente dentro do próprio sistema;
2. menos de dez por cento do código-fonte foi projetado prevendo a utilização posterior do código por outra aplicação;
3. dez por cento ou mais do código-fonte foi projetado prevendo a utilização posterior do código por outra aplicação;
4. a aplicação foi especificamente projetada e/ou documentada para ter seu código reutilizado. Ela é customizada pelo usuário no nível de código;
5. a aplicação foi especificamente projetada e/ou documentada para ter seu código reutilizado. Ela é customizada por meio de manutenção de parâmetros.

4.4.3.6.11 Facilidade de Instalação

Descreve o grau para conversão de ambientes já existentes influência o desenvolvimento da aplicação, observando-se um plano e/ou ferramentas de conversão foram fornecidas e testadas durante a fase de testes do sistema.

A pontuação é realizada de acordo com as seguintes orientações:

0. o usuário não definiu considerações especiais e nenhum procedimento especial é requerido na implantação;
1. o usuário não definiu considerações especiais, mas procedimentos especiais são necessários na implementação;

2. requisitos de conversão e implantação foram estabelecidos pelo usuário e roteiro de conversão e implantação foram fornecidas e testadas. Não é considerado um importante o impacto da conversão;
3. requisitos de conversão e implantação foram estabelecidos pelo usuário e roteiro de conversão e implantação foram fornecidas e testadas. É considerado um importante o impacto da conversão;
4. além do item dois, ferramentas de instalação e conversão automáticas foram fornecidas e testadas;
5. além do item três, ferramentas de instalação e conversão automáticas foram fornecidas e testadas.

4.4.3.6.12 Facilidade de Operação

Descreve as características da aplicação que atende a alguns procedimentos operacionais automáticos que reduzem os procedimentos manuais, bem como mecanismos de inicialização, salvamento e recuperação, verificados durante os testes do sistema.

A pontuação é realizada de acordo com as seguintes orientações:

0. nenhuma consideração especial de operação, além do processo normal de salvamento que é estabelecida pelo usuário;

1-4. quais dos itens seguintes são válidos para a aplicação. Selecionar as que forem aplicadas, onde cada item vale um ponto, a exceção de onde seja citado o contrário:

a. procedimentos de inicialização, salvamento e recuperação foram fornecidos, mas é necessária a intervenção do operador;

b. procedimentos de inicialização, salvamento e recuperação foram fornecidos, e não é necessário a intervenção do operador (contar como dois itens);

c. a aplicação minimiza a necessidade de montagem de fitas;

d. a aplicação minimiza a necessidade de manipulação de papel.

5. não é necessário nenhuma intervenção do operador para operar o sistema, que não seja a inicialização e o término da aplicação. A recuperação automática de erros é uma característica da aplicação.

4.4.3.6.13 Múltiplos Locais

Descreve se a aplicação foi especificamente projetada, desenvolvida e suportada para diferentes ambientes de hardware e software.

A pontuação é realizada de acordo com as seguintes orientações:

0. os requisitos do usuário não consideraram a necessidade de instalação em mais de um local;
1. a necessidade de múltiplos locais foi considerada no projeto e a aplicação foi desenhada para operar apenas em ambientes de software e hardware idênticos;
2. a necessidade de múltiplos locais é considerada no projeto e a aplicação foi desenhada para operar apenas em ambientes de software e hardware similares;
3. necessidade de múltiplos locais foi considerada no projeto, e a aplicação foi projetada para operar em ambientes diferentes de hardware e de software;
4. em adição aos itens um e dois, plano de documentação e suporte são fornecidos e testados parar suportar a aplicação em múltiplos locais;
5. em adição ao item três, plano de documentação e suporte são fornecidos e testados parar suportar a aplicação em múltiplos locais.

4.4.3.6.14 Facilidade de Mudanças

Descreve o grau em que a aplicação foi especificamente desenvolvida para facilitar a mudança de sua lógica de processamento ou estrutura de dados.

As seguintes características podem ser atribuídas à aplicação:

a) são fornecidos mecanismos com facilidades como consultas e relatórios flexíveis para atender necessidades simples (contar como um item);

b) são fornecidos mecanismos com facilidades como consultas e relatórios flexíveis para atender necessidades de complexidade média (contar como dois itens);

c) são fornecidos mecanismos com facilidades como consultas e relatórios flexíveis para atender necessidades complexas (contar como três itens);

d) dados de controle do negócio são mantidos pelo usuário por meio de processos interativos, mas as alterações só têm efeito no próximo dia útil;

e) dados de controle do negócio são mantidos pelo usuário por meio de processos interativos, e as alterações têm efeito imediato (contar como dois itens).

A pontuação é realizada de acordo com as seguintes orientações:

0. nenhum dos itens descritos;
1. um dos itens descritos;
2. dois dos itens descritos;
3. três dos itens descritos;
4. quatro dos itens descritos;
5. todos os cinco itens descritos.

4.4.3.7 Calcular os Pontos por Função Ajustados.

A última etapa da contagem de PF envolve o cálculo final para três tipos de contagem, que são: projeto de desenvolvimento, projeto de manutenção e aplicação

4.4.3.7.1 Projeto de Desenvolvimento

O cálculo do número de PF de um projeto em desenvolvimento é baseado na seguinte fórmula:

$$DFP = (UFP + CFP) \times VAF$$

Onde:

DFP: número de PF do projeto em desenvolvimento;

UFP: número de PFNA das funções disponíveis após a instalação;

CFP: número de PFNA das funções de conversão;

VAF: valor do fator de ajuste.

4.4.3.7.2 Projeto de Melhoria

A contagem de PF de um projeto de melhoria envolve apenas as alterações que atendam os novos requisitos de negócio do usuário, não sendo permitidas manutenções de correções e preventivas (VAZQUEZ; SIMÕES; ALBERT, 2007).

Nesse procedimento não é necessário saber o número de pontos de função da aplicação para determinar o tamanho do projeto de melhoria. Apenas medem-se as funções que serão afetadas para melhoria, caso tendo disponível, a contagem da melhoria é facilitada.

As funções excluídas apesar de reduzir o tamanho da aplicação, são contadas, pois contribuem para aumentar o tamanho do projeto de melhoria.

Uma função do tipo dado (ALI ou AIE) é considerada como um projeto de melhoria, quando ela é modificada em sua estrutura, ou seja, deve ser acrescentado, excluído ou alterado um tipo de dado para atender os requisitos de negócio.

Para considerar que uma função do tipo transação seja considerada como uma melhoria é necessária que haja inserção, exclusão ou alteração nas funções de TD, arquivos referenciados ou lógica de processamento.

Quando uma função é alterada tanto do TD como do tipo transação, conta-se toda ela no projeto de melhoria, e não apenas campos ou arquivos referenciados que mudaram.

É utilizada a seguinte fórmula para o cálculo:

$$EFP = \left[\left(ADD + CHGA + CFP\right) \times VAFA\right] + \left(DEL \times VAFB\right)$$

Onde:

EFP: número de PF do projeto de melhoria;

ADD: número de PFNA das funções incluídas pelo projeto de melhoria;

CHGA: número de PFNA das funções modificadas, refletindo as funções depois das modificações;

CFP: número de PFNA adicionados pela conversão;

VAFA: é o VFA da aplicação depois do projeto de melhoria

DEL: número de PFNA das funções excluídas pelo projeto de melhoria;

VAFB: é o VFA da aplicação antes do projeto de melhoria.

4.4.3.7.3 Aplicação

Para calcular os PF da aplicação existem duas fórmulas. Uma fórmula para contagem de PF da Aplicação e a outra para recalcular o tamanho após o projeto de melhoria ter alterado as suas funcionalidades.

A contagem de PF da aplicação é utilizada para representar todas as funcionalidades do projeto desenvolvido ao usuário. Para o cálculo é utilizada a seguinte fórmula:

$$AFP = ADD + VAF$$

Onde:

AFP: número de PFA da aplicação;

ADD: número de PFNA das funções instaladas;

VAF: VFA da aplicação.

O cálculo de PF de projeto de melhoria é utilizado para analisar as modificações aplicadas.

O cálculo utiliza a seguinte fórmula:

$$AFP = [(UFPB + ADD + CHGA) - (CHGB + DEL)] \times VAFA$$

Onde:

AFP: número de PFA da aplicação;

UFPB: número de PFNA da aplicação antes do projeto de melhoria;

ADD: número de PFNA das funções incluídas pelo projeto de melhoria;

CHGA: número de PFNA das funções alteradas pelo projeto de melhoria depois de seu término;

CHGB: número de PFNA das funções alteradas pelo projeto de melhoria antes de seu término;

DEL: número de PFNA das funções excluídas pelo projeto de melhoria;

VAFA: é o VFA depois do projeto de melhoria.

5 TRABALHOS CORRELATOS

Durante os estudos desse trabalho de pesquisa, foram analisados alguns trabalhos científicos com propósitos semelhantes a essa pesquisa, mas com outros objetivos focados, como são descritos a seguir.

5.1 USO EFICAZ DE MÉTRICAS EM MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

Dissertação de Danilo Toshiaki Sato para obtenção do grau de Mestre em Ciências em 2007 pelo Instituto de Mátematica e Estatística da Universidade de São Paulo.

A dissertação tem como objetivo apresentar uma investigação de métricas de acompanhamento de projetos mo método *Extremme Programming* da metodologia ágil. O trabalho é composto por um estudo de caso de sete projetos acadêmicos e governamentais, onde foram aplicadas algumas métricas para validar o nível de aderência às práticas.

5.2 ESTUDO COMPARATIVO DE MÉTRICAS DE SOFTWARE PARA UTILIZAÇÃO EM EMPRESAS DE DESENVOLVIMENTO DE SOFTWARE DE PEQUENO PORTE

Trabalho de Conclusão de Curso de Francielle Warmling Nuernberg para obtenção do grau de Bacharel em Ciência da Computação, em 2007, pela Universidade Do Extremo Sul Catarinense em Criciúma no estado de Santa Catarina.

O trabalho apresenta um estudo comparativo de métricas de software para utilização em empresas de desenvolvimento de software de pequeno porte para estimar custos e prazos mais precisos e ter melhor visão do progresso do projeto. Sendo realizada uma

pesquisa com as empresas para analisar a compatibilidade dos métodos de estimativas com os requisitos das empresas de desenvolvimento.

5.3 SCRUMMING: FERRAMENTA EDUCACIONAL PARA ENSINO DE PRÁTICAS DO SCRUM

Trabalho de Conclusão de Curso de Erasmo Isotton Neto para obtenção do grau de Bacharel em Sistemas de Informação, em 2008, pela Pontifícia Universidade Católica Do Rio Grande Do Sul em Porto Alegre no estado de Rio Grande do Sul.

O trabalho apresenta uma ferramenta de apoio ao ensino de práticas do Scrum, com o principal objetivo de disponibilizar essa ferramenta de aprendizagem de forma prática e interativa, denominada *Scrumming*.

6. MÉTRICAS DE SOFTWARE APLICADAS NO SCRUM

Este trabalho tem como objetivo descrever uma analise de estimativas de prazo no método Scrum sobre o estudo das métricas de software. O Scrum é um método ágil para gerenciamento de projeto ou atividades complexas com base no processo de desenvolvimento iterativo e incremental.

Entre as estimativas estudadas, encontra-se o *Ideal Day* que corresponde à quantidade de requisitos que um profissional da área consegue concluir em um dia de trabalho.

O *Planning Poker* que é realizado a partir de um baralho de cartas que possuem uma seqüência de valores, onde cada membro joga sua carta definindo um valor e explica o porque do valor e PF que é definido de acordo com a norma do IFPUG.

Após o estudo das métricas, foi realizado um estudo de caso com a finalidade de analisar qual métrica se adaptava melhor ao método Scrum.

As etapas da metodologia realizadas para a elaboração do trabalho de analise consistiu na busca em livros, revistas, sites e trabalhos científicos com conteúdos referentes à fundamentação teórica e para validar o trabalho proposto foi aplicado em um estudo de caso.

Foram realizadas analises em uma empresa que estava no início da implantação do método Scrum e com a finalidade de adotar uma métrica para estimar o prazo. Os detalhados estão na seção seguinte.

6.1 ESTUDO DE CASO

O estudo foi realizado na empresa Webmais Sistemas que é desenvolvedora de sistemas integrados de Gestão Empresarial, tendo aplicações voltadas à gestão empresarial.

Os sistemas dessa empresa são construídos de forma a permitir que o usuário possa definir suas necessidades através de parâmetros.

O sistema estudado conhecido como ERPWebmais é em plataforma web, sendo desenvolvido sobre o banco de dados pós-relacional Caché, que além da linguagem nativa, envolve as linguagens *HyperText Markup Language* (Html), *Cascading Style Sheets* (Css) e Javascript.

A empresa resolveu adotar Scrum e métricas de estimativas com o objetivo de obter maior controle dos processos de construção de software, maior produtividade no desenvolvimento e estimar um prazo mais eficiente.

Inicialmente a empresa está em fase de implantação do método Scrum para o gerenciamento de projetos. A decisão por optar pelo Scrum foi por atender os seguintes requisitos:

a) funciona de forma iterativa e incremental;
b) exige pouca documentação;
c) comunicação direta;
d) equipes de no máximo sete membros;
e) interações de trinta dias no máximo.

Ao adotar o Scrum, a empresa definiu os seguintes requisitos de forma que agilizasse os seus processos:

a) equipes de quatro integrantes, sendo geralmente composto por: dois desenvolvedores, um analista de teste e um engenheiro de software;
b) *Sprint* de dez dias úteis;
c) *Product Backlog* compostos pelas seguintes colunas: Id, Classe, Descrição, Prioridade, Iteração (o que o cliente deseja até a próxima iteração) e estimativa de prazo.

Na aplicação das métricas para estimativa de prazo de desenvolvimento, a empresa optou estudar e testar às métricas: *Ideal Day*, *Planning Poker* e Pontos de Função. As métricas foram avaliadas para analisar a que se adapta melhor nos processos de desenvolvimento.

O projeto em desenvolvimento foi sob demanda para um de seus clientes, que consistia em três novos módulos: pedido, faturamento e contas a receber. Esses módulos seriam desenvolvidos tendo base o produto já existente, assim não sendo necessário recriar alguns cadastros, como de clientes, fornecedores, cidades e entre outros.

Por os módulos serem sob demanda, necessitava a colaboração de um especialista da empresa do cliente solicitante, o *Product Owner*, para descrever e priorizar os requisitos a serem desenvolvidos.

O *Scrum Master* com o *Product Owner* e os *stakeholder* definiram o *Product Backlog* que pode ser representando no Apêndice A.

Após a elaboração do *Product Backlog* tem-se o início do *Sprint*, começando pela primeira etapa do *Sprint Planning Meeting*. Nessa primeira etapa são definidos os itens do *Product Backlog* de maior prioridade e as estimativas. O *Product Owner* selecionava os itens do *Product Backlog* detalhando-o a equipe até ela obter informações suficientes para desenvolver e para definir até qual item consegue completar no *Sprint*.

A cada item do *Product Backlog* a equipe aplicou a métrica *Ideal Day* representando no Apêndice B *e Planning Poker* representando no Apêndice C. Assim definindo a primeira iteração.

Na segunda etapa do *Sprint Planning Meeting* se planejou como seria realizado o *Sprint* a partir do *Product Backolog*. Nesta etapa a equipe criou o *Sprint Backlog* com os itens do *Product Backlog* descritos de forma mais resumida e técnica, onde alguns itens foram divididos em partes menores, por uma ordem que se adaptasse de forma mais ágil ao

desenvolvimento, também definiu os desenvolvedores responsáveis pelos itens do *Sprint Backlog*, como pode ser representando no Apêndice D.

Pela necessidade da divisão dos itens tornou-se necessário a divisão das estimativas para cada item. As estimativas do *Sprint Backlog* são feitas em horas, nesse caso foi necessário a conversão do tempo estimado no *Product Backlog* em horas de acordo com a definição da métrica.

O *Sprint Backlog* foi elaborado em uma planilha, onde possui as colunas dos dias do *Sprint* que representavam as horas trabalhadas do item por dia, representada no Anexo A.

Essa planilha sempre permanecia compartilhada para todos os membros da equipe, com a principal funcionalidade de gerar o gráfico *Burndown Chart*.

O próximo passo foi elaborar o *Taskboard* para ter um acompanhamento mais simples e prático do *Sprint*, uma parte do *Taskboard* está representado no Anexo B em forma de ilustração semelhante à utilizada.

De acordo com análise das informações discutidas na reunião, se inicia o processo de contagem dos PF[3]. Os detalhes da aplicação dessa estimativa estão detalhados no Apêndice E.

Durante o período da manha do dias do *Sprint*, a equipe se reunia de pé em círculo para realizar o *Daily Scrum*, e sempre após encerrar o *Daily Scrum* cada membro preenchia a planilha *Sprint Backlog* para ter um acompanhamento.

Após o termino do desenvolvimento do Sprint a equipe se reunião para o *Sprint Review Meeting* para discutir o *Sprint*. As principais discussões foram às seguintes:

a) relação à alteração da quantidade do produto na nota fiscal de remessa e nota fiscal de serviço;

[3] A aplicação de pontos de função foi baseada principalmente na publicação dos autores Vazquez, Simões e Albert (2007), que consta nas referências.

b) elaborações de digramas de caso de uso para facilitar posteriormente o entendimento do projeto, pois a equipe com os digramas acredita que seria mais ágil.

Outras questões foram às métricas analisadas, onde se encontram melhor descrita no item seguinte.

6.2 ANÁLISE DAS MÉTRICAS APLICADAS

O objetivo deste estudo foi verificar a viabilidade de aplicação das métricas I*deal Day*, *Planning Poker* e PF para uma primeira estimativa de prazo necessário para o desenvolvimento do primeiro *Sprint*.

O início da coleta destas estimativas se fez necessário para que a empresa começasse a formar um histórico de dados através da comparação das horas estimadas com as horas efetivas ao final do *Sprint*.

Para um melhor entendimento do projeto foram elaborados *briefings* (esboçados), servindo como um meio de auxílio para o desenvolvimento e para coleta de métricas.

A única modelagem elaborada foi o modelo de classes para auxiliar a coleta das métricas de PF.

As métricas *Ideal Day* e *Planning Poker* analisadas pela equipe deram um resultado bem similar, pois as informações foram sugeridas com valores de pouca diferença.

O preenchimento da planilha foi de acordo com as horas trabalhadas no requisito, ao invés de como *Sprint Backlog* sugere. Foi realizada dessa maneira para ter um comparativo melhor e mais fácil de ser compreendido para a definição da métrica a ser implantada. O Anexo C representa as horas efetivas nos requisitos.

A Tabela 8 representa de forma decrescente a comparação entre as horas efetivas, *Planning Poker* e *Ideal Day* por dia. A representação das horas decrescente se torna essencial para a elaboração do gráfico de *Burndown Chart.*

Tabela 8. Comparação de horas por dia

Comparação	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Horas efetivas	107,2	91,2	75,7	59,7	42,7	26	12,5	4,5
Estimativa *Planning Poker*	117,5	101,5	85,5	69,5	53,5	37,5	21,5	5,5
Estimativa *Ideal Day*	117,2	101,2	85,2	69,2	53,2	37,2	21,2	5,2

A Figura 8 representa o gráfico de *Burndown Chart* das horas efetivas do *Planning Poker*, e a Figura 9 representa o gráfico de *Burndown Chart* das horas efetivas do *Ideal Day*.

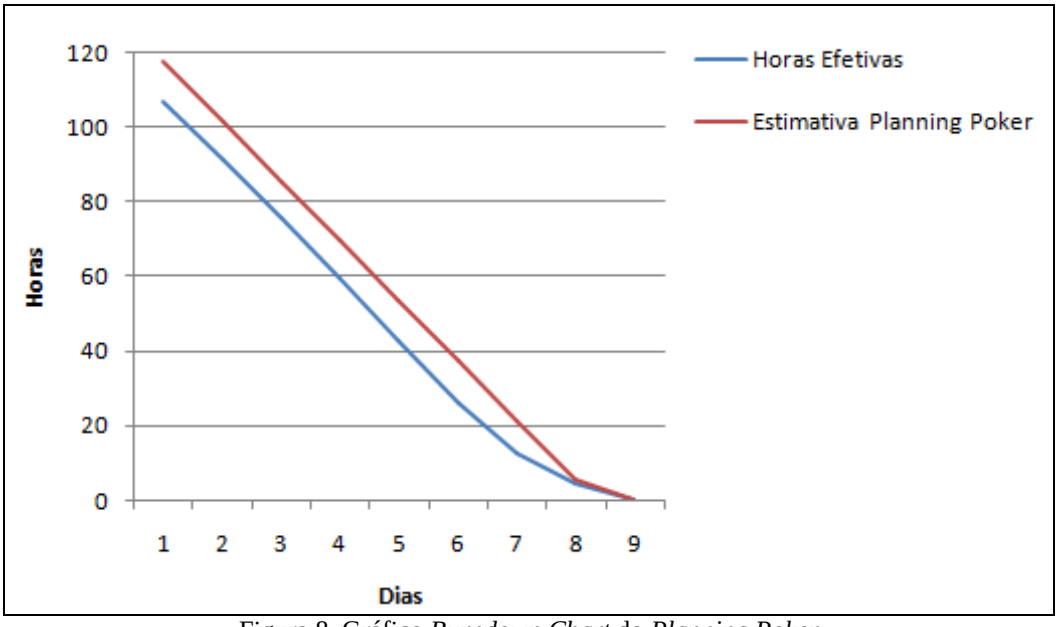


Figura 8. Gráfico *Burndown Chart* do *Planning Poker*

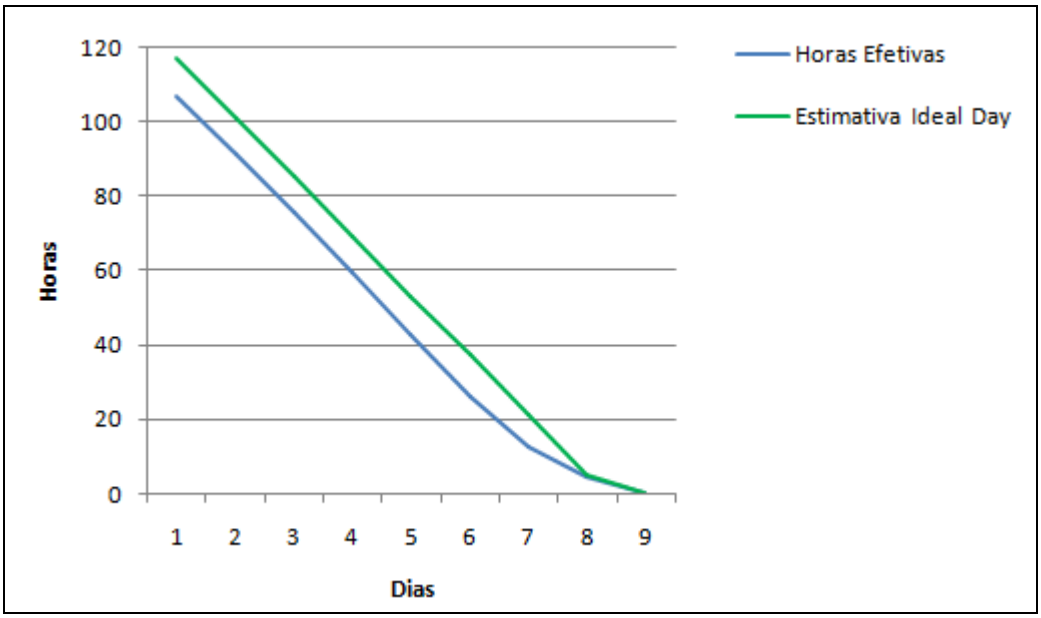


Figura 9. Gráfico *Burndown Chart* do *Ideal Day*

A Tabela 9 representa a diferença total em horas das métricas utilizadas.

Tabela 9. Comparação das estimativas

Horas Efetivas	Planning Poker	Ideal Day	Pontos de Função
107,2	117,5	117,2	40,27

A quantidade de horas calculadas pela equipe durante o *Sprint* ficou na média de 8,66% abaixo do tempo estimado das métricas *Ideal Day* e *Planning Poker*. Para o próximo *Sprint* a equipe avaliou e sugeriu novos valores das cartas do *Planning Poker*, como pode ser representado na Tabela 10.

Tabela 10. Estimativa para os próximos *Sprint* utilizando *Planning Poker*

Valor	**Horas**
2	1,5
5	3,5
8	5,5
13	8
20	12

Esses valores comparando com o *Sprint* realizado pode se obter uma diferença mais próximo do valor realizado. Com esses novos valores tem-se um valor estimado de 113 horas.

No *Ideal Day* foi sugerido para os próximos *Sprint* um percentual menor de tempo do dia que o desenvolvedor fica dedicado a implantação do item.

A coleta da estimativa de PF não foi possível definir a complexidade do projeto e sim, o tamanho. Para as próximas estimas realizadas se sugerem novos valores, sendo representados na Tabela 11.

Tabela 11. Sugestão de Horas/PF para próximo *Sprint*

Tipo	Total	Horas/PF	Total Horas
ALI	56	0,02	1,12
EE	71	0,9	63,9
CE	39	0,75	29,25
SE	45	0,27	12,15
Total	**211**		**106,42**

Os novos valores foram definidos de acordo com os resultados do primeiro *Sprint*, como não foi analisada a complexidade, para os próximos *Sprint* o valor pode ter uma variação muito alto como muita baixa.

Pela primeira analise de PF a empresa optou por analisar mais algumas vezes, para posteriormente fazer um outro estudo para analisar custos, onde PF tem a vantagem de se basear na norma da IFPUG, assim tendo um preço fixo por pontos.

As estimativas *Ideal Day* e *Planning Poker* ficaram bem próximas da realidade atual da equipe de desenvolvimento.

Para obter estimativas mais precisa é necessário que todas as iterações da fase de elaboração estejam completas, assim servindo como um histórico para outros projetos.

Neste estudo houve dificuldade na elaboração e coleta de dados de PF, pois o pelo Scrum ser uma metodologia ágil e PF não ser considerada uma métrica de estimativa de prazo ágil.

Nessa primeira aplicação do *Planning Poker* a equipe encontrou dificuldades em estimar o prazo de tempo, devido não conseguir encontrar uma estimativa de prazo ideal para a seqüência de *Fibonacci*.

As discussões realizadas para estimar tiveram maior valor a analise dos requisitos, após algumas rodadas jogando das cartas do Planning Poker para determinar o requisito e as discussões realizadas, a equipe muitas vezes encontrou problemas e soluções que alguns membros da equipe não haviam considerado e conseqüentemente obtendo um planejamento mais preciso.

Apesar de a equipe sentir dificuldade em estimar métricas de prazo no primeiro *Sprint*, elas conseguiram se tornar importante, não só por estimar o prazo, mas sim pelas discussões e trocas de idéias entre os membros da equipe.

Com a adoção do Scrum a questionamentos sobre como comparar horas estimadas com horas efetivas. A expectativa a está comparação é para melhorar cada vez mais as próximas estimativas, permitindo melhorar a previsibilidade do progresso do projeto. Contudo, quando a equipe consegue entregar dentro do prazo, isto é um fruto de empenho e determinação, e não da precisão das estimativas.

CONCLUSÃO

A adoção da metodologia ágil vem para completar as necessidades das metodologias tradicionais, visando apresentar uma alternativa para aumentar a produtividade nos projetos, de forma rápida e incremental ao decorrer do projeto.

O método Scrum pode ser considerado um método adaptativo, pois pelas incertezas do cliente em relação ao projeto, o Scrum permite as constantes modificações no projeto e sem a necessidade de exigir todos os requisitos logo no início do projeto.

Esse método não necessita de muito investimento financeiro para ser aplicado, e propõe uma nova abordagem de gerenciamento de projetos flexível e com menos formalismo.

No método Scrum como em qualquer outro método de gerenciamento, torna-se importante a aplicação de métricas para medir a produtividade, qualidade, prazo e custo. Assim a implantação de uma dessas métricas aumenta essas estimativas qualificando o projeto.

Por meio de um estudo de caso, este trabalho analisou o papel das métricas para estimar o prazo de desenvolvimento de um projeto. Nesse projeto foi aplicado o método Scrum com as métricas *Ideal Day*, *Planning Poker* e Pontos de Função.

No projeto avaliado, apesar das estimativas do *Ideal Day* e do *Planning Poker* terem resultados semelhantes, por serem empíricas, o *Ideal Day* se apresentou de melhor forma a equipe nesse início de projeto, devido a sua facilidade em estimar o prazo em dias que se pretendia terminar o requisito.

A métrica *Planning Poker* pela análise se torna interessante para medir o tamanho dos requisitos futuros, pois ao decorrer do projeto o prazo de dias do *Ideal Day* pode alterar devido a troca de membros da equipe ou outros problemas não previstos, nesse caso, com o resultado do *Planning Poker* apenas seriam alterados os valores das horas dos pontos, assim não tendo a necessidade de medir novamente.

A métrica Pontos de Função obteve o resultado divergente das outras métricas, devido não analisar a complexidade da iteração do usuário com o sistema em desenvolvimento, um dos propósitos podendo ser pelo fato do sistema ser web.

Pelo fato de o sistema ser web, tornou-se complexo medir o tamanho, mas por pontos de função avaliar o tamanho por meio dos requisitos, onde o *Product Backlog* é a elaboração de uma lista de requisitos, é possível ser aplicado em outros projetos com método Scrum. Para essa comprovação necessitaria de um estudo mais aprofundado.

Houve dificuldades em analisar mais *Sprints* e aplicar outras métricas, por questões que levaria mais tempo que o definido no cronograma. Entretanto foi delimitado o número de métricas a serem analisadas para realizar a análise mais adequada.

Contudo o estudo foi satisfatório apesar de ser estimado somente um *Sprint*, com a aplicação de duas métricas consideradas ágil e uma considerada tradicional, onde houve um entendimento mais específico da aplicação no projeto.

Como sugestão para trabalhos futuros e continuidade nessa pesquisa sugere-se:

a) analisar métricas no Scrum em outros tipo de projetos, baseado em outras linguagens;
b) aplicar outras métricas, como por exemplo, pontos de caso de uso;
c) aplicar essas métricas em outras metodologias ágeis;
d) desenvolver uma ferramenta de gerenciamento do projeto baseado em Scrum com a integração de analise de métricas.

REFERÊNCIAS

AGILE MANIFESTO. **Manifesto for Agile Software Development**. Disponível em: <http://www.agilemanifesto.org>. Acesso em: out. 2009.

ALVES, Fernanda; ALVES, Marcia; FONSECA, Isabella. **Ideal Day e Priorização**: Métodos Ágeis no Planejamento. Engenharia de Software, Rio de Janeiro, n., p.8-13, 01 nov. 2008.

BORGONOVO, Ana Marta; SILVA, Roberta Santos da. **Aplicação de metodologia ágil em grandes empresas com pequenos grupos de TI**. 2007. 82 f. - Universidade Federal De Santa Catarina, Florianópolis - Sc, 2007.

DEKKERS, Carol A.. **Pontos de Função e Medidas O que é um Ponto de Função?** Disponível em: <http://www.bfpug.com.br/Artigos/Dekkers-pontosDeFuncaoEMedidas.htm>. Acesso em: 30 jun. 2009

FOWLER, Martin. **The New Methodology**. Disponível em: < http://www.martinfowler.com/articles/newMethodology.html>. Acesso em: jun 2009.

GUERRA, Antonio Carlos Marques do Amaral. **Uma ferramenta para apoio a gestão de escopo de projetos em tecnologia da informação.** 2006. 140 f. Dissertação (Mestrado em Ciências) - Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia.

HAZAN, Cláudia. **Análise de Pontos de Função**: Uma aplicação nas estimativas de tamanho de Projetos de Software. Engenharia de Software, Rio de Janeiro, n. , p.25-30, 01 jun. 2008.

HAZAN, Cláudia. Medição da Qualidade e Produtividade em Software. In: WEBER, Kival Chaver; ROCHA, Ana Regina Cavalcanti; NASCIMENTO, Célia Joseli. **Qualidade e Produtividade em Software**. 4. ed. São Paulo: Makron Books, 2001. p. 25-41.

IFPUG, International Function Point Users Group. **Function Point Counting Practices**: Case Study 3 – Analysis, Construction. Release 2.0. Princeton Junction: IFPUG. 2001. 246 p.

INTHURN, Cândida. **Qualidade & Teste de Software**. 2. ed. Florianópolis: Visual Books Editora, 2001. 108 p.

ISOTTON NETO, Erasmo. **Scrumming**: Ferramenta Educacional para Ensino de Práticas do SCRUM. 2004. 80 f. - Pontifícia Universidade Católica Do Rio Grande Do Sul, Porto Alegre, 2004.

LINDA, Rising; NORMAN, Janoff. **The SCRUM Software Development Process for Small Teams**. IEEE Software. Ago. 2000.

KANTORSKI, Gustavo Zanini; KROTH, Marcelo Lopes. **Controle de métricas no processo de desenvolvimento de software através de uma ferramenta de workflow**. In: I WORKCOMP-SUL, 2004, Florianópolis.

KOSCIANSKI, André; SOARES, Michel Dos Santos. **Qualidade de Software**: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2. ed. São Paulo: Novatec, 2007. 395 p.

MACHADO, Marcio P.; SOUZA, Sotério F. **Métricas e Qualidade de Software.** Disponível em: < http://www.fattocs.com.br/download/qualidade-sw.pdf>. Acesso em: 27 out. 2008.

MARTINS, José Carlos Cordeiro. T**écnicas para Gerenciamento de Projetos de Software**. 1. ed. Florianópolis: Brasport Editora, 2001. 465 p.

NUERNBERG, Francielle Warmling. **Estudo Comparativo de Métricas de Software para Utilização em Empresas de Desenvolvimento de Software de Pequeno Porte**. 2007. 75 f. Trabalho de Conclusão de Curso (Graduação em Ciencia da Computação) – Universidade do Extremo Sul Catarinense, Criciúma.

PLANNING POKER. **Play. Estimate. Plan**. Disponível em: < http://www.planningpoker.com/>. Acesso em: 01 nov. 2008.

PRESSMAN, Roger S. **Engenharia de Software**. 6. ed São Paulo: McGraw-Hill, 2006. 720 p.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995. 1056 p.

SATO, Danilo Toshiaki. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software**. 2007. 155 f. Mestre Em Ciências - Universidade De São Paulo, São Paulo, 2007.

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

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado. **Análise de pontos de função**: medição, estimativas e gerenciamento de projetos de software. São Paulo: Érica, 2007. 230 p.

YOSHIMA, Rodrigo. **Gerenciamento de Projetos com Scrum**. 2007. Disponível em <http://www.aspercom.com.br/ead/index.php>. Acesso em: 12 ago. 2009.

ZANATTA, Alexandre Lazaretti. **XScrum**: uma proposta de extensão de um Método Ágil para Gerência e Desenvolvimento de Requisitos visando adequação ao CMMI. 2004. 180 f. - Universidade Federal De Santa Catarina, Florianópolis, 2004.

APÊNDICE A – PRODUCT BACKLOG

A Tabela 12 representa o *Product Backlog* da *release* do projeto proposto.

Tabela 12. Product Backlog

PRODUCT BACKLOG Realese 1					**Atualização: 14/09/2009**
Iter	**Id**	**Assunto**	**Item**	**Prior.**	**Classe**
1	1	Cadastro	Cadastro de pedidos. Consiste em gravar código, cliente, data de emissão, data de saída. No momento apenas cadastrar um produto com sua quantidade e valor de venda.	1	Pedido
1	2	Edição	Editar pedidos. Consiste em poder alterar um pedido já gravado, sendo possível alterar o cliente, a data de saída, o produto e seu respectivo o valor e quantidade. Para editar o pedido é necessário informar o código do pedido ou ter uma forma para mostrar os pedidos já cadastrados, assim possibilitando escolher e abrir o pedido para ser alterado. O pedido não pode estar selecionado em nenhuma nota fiscal de remessa para ser editado.	1	Pedido
1	3	Consulta	Consulta de pedidos. Ter os seguintes campos de filtro: período de data de emissão, código do pedido, cliente e produto. E na lista da consulta mostrar as colunas: código, data de emissão, cliente, produto, valor, quantidade, valor total e ter um link para abrir o espelho do pedido.	2	Pedido
1	4	Relatório	Espelho do pedido. Consiste em um relatório para mostrar todas as informações que foram gravadas no pedido. Sendo apenas possível abrir na consulta de pedidos.	3	Pedido
1	5	Cadastro	Cadastro da nota fiscal de remessa. Consiste em gravar o código, cliente, cidade do cliente, data de emissão, data de saída, seleção dos pedidos do mesmo cliente com opção de alterar a quantidade e o valor do produto dos seus respectivos pedido. A quantidade a ser alterada não pode ser maior que a cadastrada no pedido.	1	NFRemessa

Iter	Id	Assunto	Item	Prior.	Classe
1	6	Edição	Editar nota fiscal de remessa. Consiste em poder alterar uma nota fiscal de remessa já cadastrada, sendo possível alterar a data de saída, os pedidos com seus respectivos valores e quantidades do produto, acrescentar novos pedidos, sendo da mesma cidade e do mesmo cliente e de remover da nota fiscal de remessa. Para editar a nota fiscal de remessa é necessário informar o código da nota fiscal de remessa ou ter uma forma de mostrar as notas fiscais de remessa já cadastradas, assim possibilitando escolher e abrir a nota fiscal de serviço para ser alterada. A nota fiscal de remessa não pode estar selecionada em nenhuma nota fiscal de serviço para ser editada.	1	NFRemessa
1	7	Consulta	Consulta da Nota Fiscal de Remessa. Ter os seguintes campos de filtro: período de data de emissão, código da nota fiscal de remessa, cliente, cidade e código do pedido. E na lista da consulta mostrar as colunas: código, data de emissão, cliente, cidade, valor total e ter um link para mostrar o espelho da nota fiscal de remessa.	2	NFRemessa
1	8	Relatório	Espelho da Nota Fiscal de Remessa. Consiste em mostrar todas as informações que foram gravadas no cadastro de nota fiscal de remessa. Sendo apenas possível abrir na consulta de nota fiscal de remessa selecionadas.	3	NFRemessa
1	9	Cadastro	Cadastro da Nota Fiscal de Serviço. Consiste em gravar o código, cliente, cidade, data de emissão, data de saída e seleção das notas fiscais de remessa do mesmo cliente e da mesma cidade com opção de alterar a quantidade e o valor do produto das respectivas notas ficais de remessa. A quantidade a ser alterada não pode ser maior que a cadastrada na nota fiscal de remessa.	1	NFServico

Iter	Id	Assunto	Item	Prior.	Classe
1	10	Edição	Editar nota fiscal de serviço. Consiste em poder alterar uma nota fiscal de serviço já cadastrada, sendo possível alterar a data de saída, as notas fiscais de remessa com os respectivos valores e quantidades do produto, acrescentar novas notas fiscais de remessa, sendo da mesma cidade e do mesmo cliente e de remover da nota fiscal de serviço. Para editar a nota fiscal de serviço é necessário informar o código da nota fiscal de serviço ou ter uma forma de mostrar as notas fiscais de serviço já cadastradas, assim possibilitando escolher e abrir a nota fiscal de serviço para ser alterada.	1	NFServico
1	11	Consulta	Consulta da Nota Fiscal de Serviço. Ter os seguintes campos de filtro: período de data de emissão, código da nota fiscal de serviço, cliente, cidade e código da nota fiscal de remessa. E na lista da consulta mostrar as colunas: código, data de emissão, cliente, cidade, valor total e ter um link para mostrar o espelho da nota fiscal de serviço.	2	NFServico
1	12	Relatório	Espelho da Nota Fiscal de Serviço. Consiste em mostrar todas as informações que foram gravadas no cadastro de nota fiscal de serviço. Sendo apenas possível abrir na consulta de nota fiscal de serviço.	3	NFServico
1	13	Cadastro	Cadastro do Contas a Receber. Consiste em gravar o código, cliente, portador, tipo de cobrança, data de emissão, data de vencimento e de selecionar os pedidos do mesmo cliente. Tendo o valor do documento a partir da soma dos pedidos selecionados.	2	ContasReceber
1	14	Edição	Editar Contas a Receber. Consiste em poder alterar o portador, tipo de cobrança e a seleção de pedidos do mesmo cliente, acrescentando ou removendo. Para editar o contas a receber é necessário informar o código ou ter uma forma de mostrar as contas a receber já cadastradas, assim possibilitando escolher e abrir o contas a receber para ser alterado. No contas a receber não pode ter ocorrido nenhum movimento para ser editado.	3	ContasReceber

Iter	Id	Assunto	Item	Prior.	Classe
1	15	Consulta	Consulta do Contas a Receber. Ter os seguintes campos de filtro: período de data de emissão e vencimento, código do contas a receber, cliente, pedido. E na lista da consulta mostrar as colunas: código, data de emissão, data de vencimento, cliente, valor total e ter um link para mostrar a impressão do contas a receber.	2	ContasReceber
1	16	Relatório	Impressão do Contas a Receber. Consiste em mostrar todas as informações que foram gravadas no cadastro de contas a receber. Sendo apenas possível abrir na consulta de contas a receber.	2	ContasReceber
1	17	Edição	Ter um campo de status no pedido para quando o pedido for salvo colocar o status como finalizado, e quando estiver em uma nota fiscal de remessa trocar para nota fiscal de remessa, quando estiver em nota fiscal de serviço trocar para nota fiscal de serviço ou quando for cancelado colocar o status como cancelado.	1	Pedido
	18	Edição	Ter um campo de status na nota fiscal de remessa para quando for salva colocar o status como finalizado, quando estiver em nota fiscal de serviço trocar o status para nota fiscal de serviço ou quando for cancelada trocar o status para cancelado.		NFRemessa
	19	Edição	Ter um campo de status na nota fiscal de serviço para quando for salva colocar o status como finalizado ou quando for cancelado trocar o status para cancelado.		NFServico
	20	Gerenciador	Gerenciador de impressão. Esse gerenciador consiste em uma tela similar a de consulta de nota de nota fiscal de serviço, com o acréscimo de um botão de imprimir para cada nota fiscal de serviço. Após imprimir o botão deve ser inativado, pois a nota fiscal de serviço pode ser impresso uma única vez. Caso precise ser impressa novamente, será necessário digitar uma senha para habilitar a impressão, após impresso, o botão será inativado novamente, assim sucessivamente. Somente as notas fiscais de serviço com o status finalizado podem ser impressas. Ao imprimir o status da nota fiscal de serviço deve ser trocado para impressa.		NFServico

Iter	Id	Assunto	Item	Prior.	Classe
	21	Gerenciador	Gerenciador de impressão. Acrescentado um campo status para filtro, e como padrão vim somente as notas fiscais de serviço com status finalizado. O filtro status tem as seguintes opções: finalizadas, impressas e canceladas.		NFServico
	22	Relatório	Impressão da nota fiscal de serviço. Consiste na impressão da nota fiscal de serviço direto em uma impressora matricial.		NFServico
	23	Edição	Cancelamento de nota fiscal de serviço. Consiste em mudar o status da nota fiscal de serviço para cancelado. É necessário digitar uma senha e gravar o motivo, o usuário, a data e a hora do cancelamento.		NFServico
	24	Cadastro	Cadastrar movimento do contas a receber. Consiste em gravar o recebimento. Para gravar é necessário informar o código do contas a receber, o código da conta que foi ou que deseja baixar, a data, o valor pago, o valor de desconto, o valor de juros e o valor de multa.		ContasMovimentos
	25	Consulta	Mostrar os movimentos do contas a receber na consulta do contas a receber. Para cada item do contas a receber, deve ter uma forma que mostre os movimentos. Também deve ser acrescentado um campo de filtro para escolher somente as contas a receber sem recebimentos, com recebimentos ou com ambas.		ContasMovimentos
	26	Consulta	Consulta de pedidos. Acrescentar um campo status de filtro e uma coluna de status na lista de consulta. O filtro status tem as seguintes opções: finalizado, nota fiscal de remessa, nota fiscal de serviço ou cancelado.		Pedido
	27	Consulta	Consulta de nota fiscal de remessa Acrescentar um campo status de filtro e uma coluna de status na lista de consulta. O filtro status tem as seguintes opções: finalizado, nota fiscal de serviço ou cancelado.		NFRemessa

Iter	Id	Assunto	Item	Prior.	Classe
	28	Edição	Cancelamento de nota fiscal de remessa. Consiste em mudar o status da nota fiscal de remessa para cancelado. É necessário informar o código da nota fiscal de remessa, informar uma senha de cancelamento, gravar o motivo, o usuário, a data e a hora do cancelamento. A nota fiscal de remessa somente pode ser cancelada quando o status for finalizado.		NFRemessa
	29	Edição	Cancelamento de pedido. Consiste em mudar o status do pedido para cancelado. É necessário informar o código da nota fiscal de serviço, informar uma senha de cancelamento, gravar o motivo, o usuário, a data e a hora do cancelamento. O pedido somente pode ser cancelado quando o status for finalizado.		Pedido
	30		Exclusão do movimento do contas a receber. Consiste em apagar o movimento do contas a receber. Possuir um botão de exclusão na mesma tela de cadastro de movimento.		ContasMovimentos
	31	Relatório	Relatório de notas fiscais de serviço emitidas por data. Deve ser informar um período de data de emissão, cliente e os status. Assim gerando o relatório com as seguintes informações: código, data de emissão, cliente, cidade, status e valor.		NFServico
	32	Relatório	Relatório de notas fiscais de remessa emitidas por data. Deve se informar um período de data de emissão, cliente, e os status. Assim gerando o relatório com as seguintes informações: código, data de emissão, cliente, cidade, status e valor.		NFRemessa
	33	Relatório	Relatório de pedidos emitidos por data. Deve se informar um período de data de emissão, cliente, e os status. Assim gerando o relatório com as seguintes informações: código, data de emissão, cliente, status, produto, quantidade, valor de venda do produto e valor total.		Pedido
	34	Relatório	Relatório dos movimentos do contas a receber recebidos por data. Deve ser informado um período de data de recebimento, cliente, e o status. Assim gerando o relatório com as seguintes informações: código, data de recebimento, cliente e valor do contas a receber e valor total recebido.		ContasMovimentos

APÊNDICE B – IDEAL DAY

Ao decorrer da primeira parte do *Product Backlog* a equipe aplicou a métrica *Ideal Day* para estimar o prazo. A previsão foi realizada de acordo com a seguinte escala de porcentagem do dia: 0,25, 0,5, 0,75, 1, 1,25, 1,5 e assim por diante.

A equipe resolveu aplicar *Ideal Day* primeiro, pois considerou uma métrica fácil de ser utilizada e para ter um indicador na extração das outras métricas.

Para encontrar o *Ideal Day* a equipe discutiu cada item até chegar a um consenso como pode ser visto na Tabela 13.

Tabela 13. Estimativa *Ideal Day – Sprint* 1

Id	IED	IED Estimado	Horas	Horas Arredondados
1	1	1,12	8,96	9
2	0,75	0,84	6,72	6,7
3	0,75	0,84	6,72	6,7
4	0,5	0,44	3,52	3,5
5	1,25	1,40	11,20	11,2
6	1	1,12	8,96	9
7	0,75	0,84	6,72	6,7
8	0,5	0,44	3,52	3,5
9	1,25	1,40	11,20	11,2
10	0,75	0,84	6,72	6,7
11	0,75	0,84	6,72	6,7
12	0,5	0,44	3,52	3,5
13	1,25	1,40	11,20	11,2
14	1	1,12	8,96	9
15	0,75	0,84	6,72	6,7
16	0,5	0,44	3,52	3,5
17	0,25	0,28	2,24	2,5
Total:	**13,5**	**14,64**	**117,12**	**117,3**

Foi considerado que a equipe trabalharia 90% do seu dia que é representado por oito horas de trabalho diário, assim foi calculado o *Ideal Day* estimado de acordo com Martins (2007).

APÊNDICE C - PLANNING POKER

Para aplicar a métrica *Planning Poker*, cada membro da equipe possuía um jogo de cartas com a seqüência de *Fibonacci*. A equipe inicia o jogo escolhendo o requisito mais fácil a ser desenvolvido que tem o peso de dois pontos.

O Tabela 14 representa a estimativa do *Planning Poker* definida pela equipe.

Tabela 14. Estimativa *Planning Poker – Sprint* 1

Id	Planning Poker	Horas
1	13	8
2	8	6
3	8	6
4	5	4
5	20	12
6	13	8
7	13	8
8	5	4
9	20	12
10	8	6
11	8	6
12	5	4
13	20	12
14	13	8
15	8	6
16	8	6
17	2	1,5
Total:	**177**	**117,5**

O *Planning Poker* necessitou de duas rodadas, pois na primeira os Id 2, 5, 6, 7, 8, 9, 10, 11, 12, 17 e 19 houve divergência entre a equipe.

A Tabela 15 e a Tabela 16 representam duas das divergências ocorridas no projeto.

Tabela 15. Planning Poker - Rodada Id 2

Equipe	Rodada 1	Rodada 2
Adriano	20	8
Daniel	3	5
Mateus	5	8
Leonardo	13	8

Tabela 16. Planning Poker - Rodada Id 15

Equipe	Rodada 1	Rodada 2
Adriano	8	13
Daniel	5	8
Mateus	5	8
Leonardo	5	8

O Id 2 (Tabela 16) na primeira rodada a equipe ficou com duvida, já na segunda rodada, após a discussão sobre o requisito a equipe chegou um consenso que seria fácil e rápido de desenvolver. Com o requisito do Id 1 pronto, o Id 2 poderia ser implementado em cima do Id 1, assim tento reaproveitamento e o requisito valendo 8 pontos.

O Id 15 (Tabela 17) na primeira rodada alguns membros sugeriram pegar a consulta já desenvolvido de um projeto similar e adaptar para esse módulo e outros membros de desenvolver uma nova consulta para esse módulo, pois os argumentos era que poderia ser feito desenvolvido de uma forma que ficasse mais rápida a pesquisa e a tela mais limpa. Na segunda rodada a equipe em geral tomou a decisão de desenvolver uma consulta nova, assim, tomando a decisão que o requisito iria valer 8 pontos

APÊNDICE D – SPRINT BACKLOG

A Tabela 17 representa os itens do primeiro *Sprint* do projeto.

Tabela 17. Itens do primeiro *Sprint*

Id	Classe	Item	Responsável
1	Pedido	Cadastro de pedidos com campos: código, data emissão, data saída, produto(apenas um) com seu valor e quantidade.	Daniel
2	Pedido	Edição de pedidos utilizar o cadastro de pedidos de forma que busque o pedido (tela de consulta simples) ou pelo código. Campos a serem alterados: cliente, data de saída, produto e seu respectivo o valor e quantidade.	Daniel
3	Pedido	Criar uma tela de consulta simples que mostre os pedidos (código,emissão,cliente) cadastrados. Podendo ter cliente como parâmetro de entrada.	Daniel
4	Pedido	Consulta de pedido com campos de filtro: período de data de emissão, código do pedido, cliente e produto, listando: código, data de emissão, cliente, produto, valor, quantidade, valor total e link para espelho do pedido	Adriano
5	Pedido	Espelho do pedido deve mostrar todas as informações.	Adriano
6	NFRemessa	Cadastro de nota fiscal de remessa com campos: código, cliente, cidade do cliente, data de emissão, data de saída, seleção dos pedidos (tela de consulta simples pedido) do mesmo cliente com opção de alterar a quantidade e o valor do produto do respectivo pedido.	Daniel
7	NFRemessa	Edição nota fiscal de remessa utilizar o cadastro de nota fiscal de remessa de forma que busque a nota fiscal de remessa (tela de consulta simples) ou pelo código. Campos a ser alterados: data de saída e os pedidos com seus respectivos valores e quantidade do produto.	Daniel
8	NFRemessa	Criar uma tela de consulta simples que mostre as notas fiscais de remessa (código, emissão, cliente) cadastradas. Podendo ter como parâmetros de entrada: cliente e cidade.	Daniel
9	NFRemessa	Consulta da nota fiscal de remessa: período de data de emissão, código da nota fiscal de remessa, cliente e código do pedido, listando: código, data de emissão, cliente, cidade, valor total e link para espelho da nota fiscal de remessa.	Adriano
10	NFRemessa	Espelho da nota fiscal de remessa deve mostrar todas as informações.	Adriano
11	NFServico	Cadastro de nota fiscal de serviço com campos: código, cliente, cidade, data de emissão, data de saída e seleção das notas fiscais de remessa (tela de consulta simples nota fiscal de remessa) do mesmo cliente e da mesma cidade com opção de alterar a quantidade e o valor do produto das respectivas nota fiscais de remessa selecionadas.	Daniel

Id	Classe	Item	Responsável
12	NFServico	Edição de nota fiscal de serviço utilizar o cadastro de nota fiscal de serviço de forma que busque a nota fiscal de serviço (tela de consulta simples nota fiscal de serviço) ou pelo código. Campos a serem alterados: data de saída e as notas fiscais de remessa e seus respectivos valores e quantidade do produto.	Daniel
13	NFServico	Criar uma tela de consulta simples que mostre as notas fiscais de saída (código,emissão,cliente) cadastrados.	Daniel
14	NFServico	Consulta da nota fiscal de serviço: período de data de emissão, código da nota fiscal de serviço, cliente e código da nota fiscal de remessa, listando: código, data de emissão, cliente, cidade, valor total e link para espelho da nota fiscal de serviço.	Adriano
15	NFServico	Espelho da nota fiscal de serviço deve mostrar todas as informações.	Adriano
16	ContasReceber	Cadastro do contas a receber com campos: código, cliente, portador, tipo de cobrança, data de emissão, data de vencimento e seleção de pedidos do mesmo cliente. Criar uma função que busque o valor total do contas a receber a partir da soma dos pedidos.	Adriano
17	ContasReceber	Edição do contas a receber utilizar o cadastro de contas a receber de forma que busque o contas a receber pela tela de consulta simples do contas a receber ou pelo código. Campos a serem alterados: portador, tipo de cobrança e a seleção de pedidos do mesmo cliente. Para poder alterar é necessário que não tenha ocorrido nenhum movimento.	Adriano
18	ContasReceber	Criar uma tela de consulta simples que mostre as as contas a receber (código,emissão,cliente) cadastrados.	Adriano
19	ContasReceber	Consulta do contas a receber: período de data de emissão e vencimento, código do contas a receber, cliente, pedido, listando: código, data de emissão, data de vencimento, cliente, valor total e link para impressão do contas a receber.	Daniel
20	ContasReceber	Impressão do contas a receber deve mostrar todas as informações.	Daniel
21	Pedido	Criar propriedade Status. Ao salvar o pedido preencher automático com Finalizado.	Adriano
22	Pedido	Ao salvar nota fiscal de remessa, os pedidos inseridos devem mudar status para Nota Fiscal de Remessa. Ao remover mudar status para Finalizado	Adriano
23	Pedido	Ao salvar nota fiscal de serviço, os pedidos inseridos nas notas fiscais de remessa devem mudar status para Nota Fiscal de Serviço. Ao remover mudar status para Nota Fiscal de Remessa	Adriano

APÊNDICE E – PONTOS DE FUNÇÃO

Para aplicar PF necessitou se de um diagrama classes, representado na Figura 10 que veio a facilitar a coleta dos ALI e AIE.

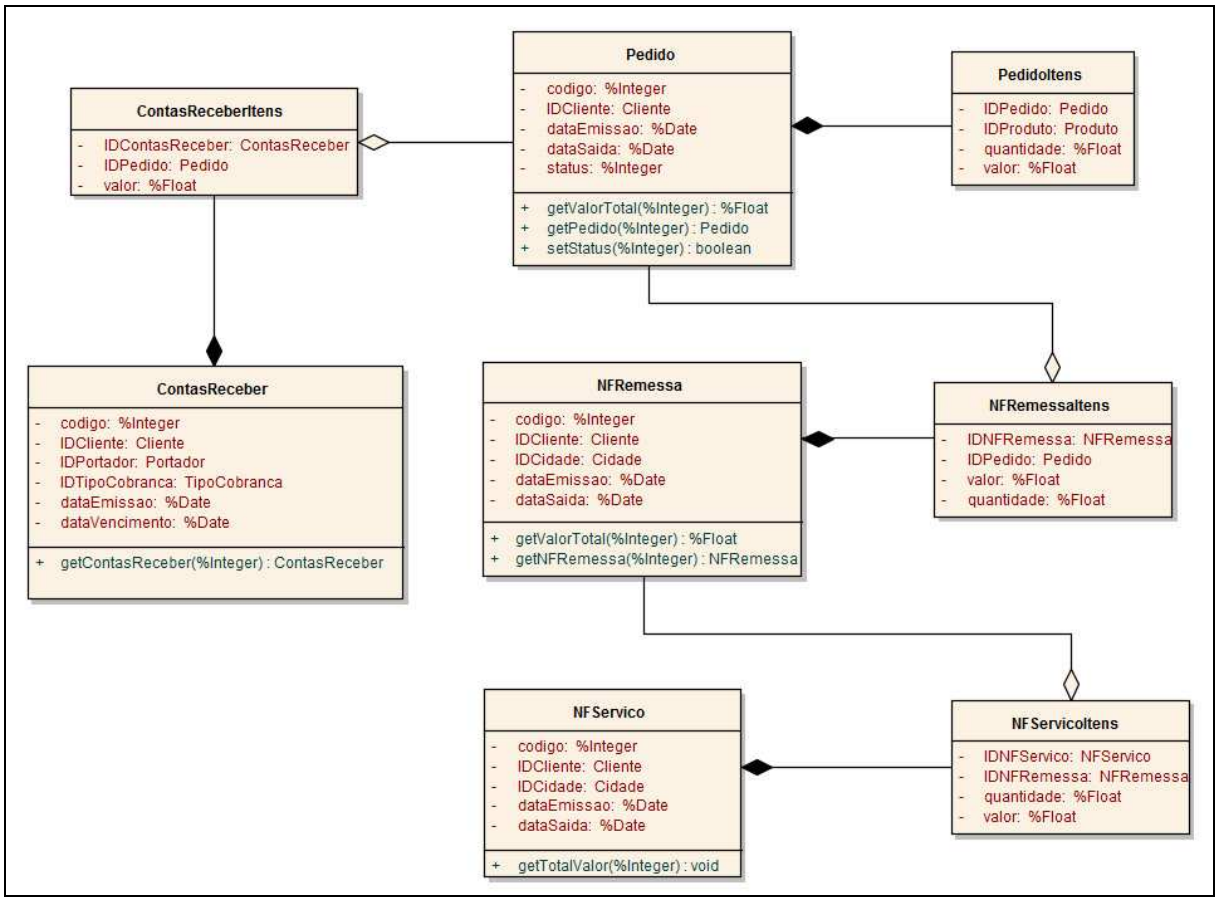


Figura 10. Diagrama de Classes

As EE, CE e SE foi utilizado as informações coletadas durante o *Sprint Planning Meeting* e o *Sprint Backlog*. A Tabela 18, a Tabela 19, a Tabela 20 e a Tabela 21 representam a contagem dos PF, onde cada tabela representa uma classe, assim, facilitando a contagem.

Tabela 18. Contagem dos PF da classe Pedido

Processo Elementar ou Grupo de Dados	Tipo	TD	AR/TR	Complex.	PF
Classe Pedido	ALI	5	1	Simples	7
Classe PedidoItens	ALI	4	2	Simples	7
Métodos Classe Pedido	CE	2	0	Simples	3
Métodos Classe Pedido	EE	1	0	Simples	3
Cadastro de Pedido	EE	4	1	Simples	3

Processo Elementar ou Grupo de Dados	Tipo	TD	AR/TR	Complex.	PF
Edição de Pedido	EE	2	1	Simples	3
Consulta de Pedido	CE	5	2	Média	4
Consulta de Pedido – Lista	SE	6	2	Média	5
Espelho do Pedido	SE	7	2	Média	5
Cadastro do Item do Pedido	EE	4	2	Simples	3
Edição do Item do Pedido	EE	3	1	Simples	3
Exclusão do Itens do Pedido	EE	4	2	Simples	3
Consulta Rápida de Pedido	EE	1	1	Simples	3
Consulta Rápida de Pedido - Lista	CE	3	1	Simples	3
Total					**55**

Tabela 19. Contagem dos PF da classe NFRemessa

Processo Elementar ou Grupo de Dados	Tipo	TD	AR/TR	Complex.	PF
Classe NFRemessa	ALI	5	2	Simples	7
Classe NFRemessaItens	ALI	4	2	Simples	7
Métodos Classe NFRemessa	CE	2	0	Simples	3
Cadastro de Nota Fiscal de Remessa	EE	5	2	Média	4
Edição de Nota Fiscal de Remessa	EE	1	0	Simples	3
Consulta de Nota Fiscal de Remessa	CE	5	2	Média	4
Consulta de Nota Fiscal de Remessa - Lista	SE	5	2	Média	5
Espelho de Nota Fiscal de Remessa	SE	9	4	Complexa	7
Cadastro de Itens da Nota Fiscal de Remessa	EE	4	2	Simples	3
Edição de Itens da Nota Fiscal de Remessa	EE	3	1	Simples	3
Exclusão de Itens da Nota Fiscal de Remessa	EE	4	2	Simples	3
Consulta Rápida de Nota Fiscal de Remessa	EE	2	2	Simples	3
Consulta Rápida de Nota Fiscal de Remessa - Lista	CE	5	2	Simples	3
Total					**55**

Tabela 20. Contagem dos PF da classe NFServiço

Processo Elementar ou Grupo de Dados	Tipo	TD	AR/TR	Complex.	PF
Classe NFServico	ALI	5	2	Simples	7
Classe NFServicoItens	ALI	4	2	Simples	7
Métodos Classe NFServico	CE	1	0	Simples	3
Cadastro de Nota Fiscal de Serviço	EE	5	2	Média	4
Edição de Nota Fiscal de Serviço	EE	1	0	Simples	3
Consulta de Nota Fiscal de Serviço	CE	5	2	Média	4
Consulta de Nota Fiscal de Serviço - Lista	SE	5	2	Média	5
Espelho de Nota Fiscal de Serviço	SE	9	4	Complexa	7
Cadastro de Itens da Nota Fiscal de Serviço	EE	4	2	Simples	3
Edição de Itens da Nota Fiscal de Serviço	EE	3	1	Simples	3
Exclusão de Itens da Nota Fiscal de Serviço	EE	4	2	Simples	3
Consulta Rápida de Nota Fiscal de Serviço	CE	3	0	Simples	3
Total					**52**

Tabela 21. Contagem dos PF da classe ContasReceber

Processo Elementar ou Grupo de Dados	Tipo	TD	AR/TR	Complex.	PF
Classe ContasReceber	ALI	6	3	Simples	7
Classe ContasReceberItens	ALI	3	2	Simples	7
Métodos Classe ContasReceber	CE	1	0	Simples	3
Cadastro de Contas a Receber	EE	6	3	Complexa	6
Edição de Contas a Receber	EE	2	2	Simples	3
Consulta de Contas a Receber	CE	7	1	Simples	3
Consulta de Contas a Receber - Lista	SE	5	1	Simples	4
Impressão de Contas a Receber	SE	9	5	Complexa	7
Cadastro de Itens do Contas a Receber	EE	3	2	Simples	3
Exclusão de Itens do Contas a Receber	EE	3	2	Simples	3
Consulta Rápida de Contas a Receber	CE	3	0	Simples	3
Total					**49**

Como o *Sprint Backlog* e o *Product Backlog* foram agrupados em classes, os PF foram separados por classes, pois posteriormente seria mais fácil de fazer uma contagem de PF de projeto de melhoria.

Os pontos e a complexidade foram realizados de acordo com o padrão da IFPUG que é descrito no Capítulo 4.4.

Esse projeto por ser um novo módulo foi considerado um projeto de desenvolvimento. A estimativa é realizada foi separada por TD e por tipo de transação, pois para cada tipo poderia ser representado um prazo diferente, como pode ser representada na Tabela 22.

Tabela 22. Estimativa de PF

Tipo	Total	Horas/PF	Total Horas
ALI	56	0,02	1,12
EE	71	0,3	21,3
CE	39	0,4	15,6
SE	45	0,05	2,25
Total	**211**		**40,27**

O fator de ajuste já foi considerado nos TD e tipo de transação, assim não tendo a necessidade nesse projeto de utilizar os fatores de ajustes.

ANEXO A – SPRINT BACKLOG

ID	Itens	Horas Ideal Day	Horas Planning Poker	1	2	3	4	5	6	7	8	9	Total Horas Item
1	Cadastro Pedido...	9	8	0	0	0	0	0	0	0	0	0	0
2	Edição Pedido...	5	4,5	0	0	0	0	0	0	0	0	0	0
3	Consulta Rapida Pedido...	1,7	1,5	0	0	0	0	0	0	0	0	0	0
4	Consulta Pedido...	6,7	6	0	0	0	0	0	0	0	0	0	0
5	Espelho Pedido...	3,5	4	0	0	0	0	0	0	0	0	0	0
6	Cadastro NFR...	11,2	12	0	0	0	0	0	0	0	0	0	0
7	Edição NFR...	7	6	0	0	0	0	0	0	0	0	0	0
8	Consulta Rapida NFR...	2	2	0	0	0	0	0	0	0	0	0	0
9	Consulta NFR...	6,7	8	0	0	0	0	0	0	0	0	0	0
10	Espelho NFR...	3,5	4	0	0	0	0	0	0	0	0	0	0
11	Cadastro NFS...	11,2	12	0	0	0	0	0	0	0	0	0	0
12	Edição NFS...	5	4,5	0	0	0	0	0	0	0	0	0	0
13	Consulta Rapida NFS...	1,7	1,5	0	0	0	0	0	0	0	0	0	0
14	Consulta NFS...	6,7	6	0	0	0	0	0	0	0	0	0	0
15	Espelho NFS...	3,5	4	0	0	0	0	0	0	0	0	0	0
16	Cadastro CR...	11,2	12	0	0	0	0	0	0	0	0	0	0
17	Edição CR...	7,5	6,5	0	0	0	0	0	0	0	0	0	0
18	Consulta Rapida CR...	1,5	1,5	0	0	0	0	0	0	0	0	0	0
19	Consulta CR...	6,7	6	0	0	0	0	0	0	0	0	0	0
20	Impressao CR...	3,5	6	0	0	0	0	0	0	0	0	0	0
21	Status Pedido...	0,8	0,5	0	0	0	0	0	0	0	0	0	0
22	Status Pedido...	0,8	0,5	0	0	0	0	0	0	0	0	0	0
23	Status Pedido...	0,8	0,5	0	0	0	0	0	0	0	0	0	0
Total		**117,2**	**117,5**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ANEXO B – TASKBOARD

ANEXO C – SPRINT BAKCLOG REALIZADO

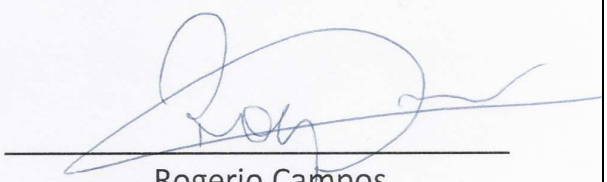
Id	Horas Ideal Day	Horas Planning Poker	1	2	3	4	5	6	7	8	Total
1	9	8	8	1	0	0	0	0	0	0	9
2	5	4,5	0	4	0	0	0	0	0	0	4
3	1,7	1,5	0	2	0	0	0	0	0	0	2
4	6,7	6	6	0	0	0	0	0	0	0	6
5	3,5	4	2	0,5	0	0	0	0	0	0	2,5
6	11,2	12	0	1	8,5	3	0	0	0	0	12,5
7	7	6	0	0	0	5	0	0	0	0	5
8	2	2	0	0	0	0	2,2	0	0	0	2,2
9	6,7	8	0	7	0	0	0	0	0	0	7
10	3,5	4	0	0	3	0	0	0	0	0	3
11	11,2	12	0	0	0	0	6,5	5,5	0	0	12
12	5	4,5	0	0	0	0	0	2,5	1,5	0	4
13	1,7	1,5	0	0	0	0	0	0	1,5	0	1,5
14	6,7	6	0	0	4,5	0	0	0	0	0	4,5
15	3,5	4	0	0	0	3	0	0	0	0	3
16	11,2	12	0	0	0	6	5,5	0	0	0	11,5
17	7,5	6,5	0	0	0	0	2,5	2,5	0	0	5
18	1,5	1,5	0	0	0	0	0	1,5	0	0	1,5
19	6,7	6	0	0	0	0	0	0	5	0,5	5,5
20	3,5	6	0	0	0	0	0	0	0	4	4
21	0,8	0,5	0	0	0	0	0	0,5	0	0	0,5
22	0,8	0,5	0	0	0	0	0	0,5	0	0	0,5
23	0,8	0,5	0	0	0	0	0	0,5	0	0	0,5
Total	**117,2**	**117,5**	**16**	**15,5**	**16**	**17**	**16,7**	**13,5**	**8**	**4,5**	107,2

ANEXO D – DECLARAÇÃO

Criciúma, 24 de novembro de 2009

DECLARAÇÃO

Declaro, para os devidos fins, que concordo em disponibilizar um estudo de caso realizado na Empresa WEBMAIS Sistemas Ltda, para elaboração de Trabalho de Conclusão de Curso **Aplicação de Métricas de Software no Método Scrum da Metodologia Ágil**, do Bacharelado em Ciência da Computação da Universidade do Extremo Sul Catarinense - UNESC, do acadêmico Mateus Luiz Gamba.



Rogerio Campos
WEBMAIS Sistemas Ltda