UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Alexandre Lazaretti Zanatta

xScrum: uma proposta de extensão de um Método Ágil para Gerência e Desenvolvimento de Requisitos visando adequação ao CMMI

Dissertação de mestrado submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Orientadora
Profª. Drª. Patrícia Vilain

Florianópolis, dezembro de 2004

xScrum: uma proposta de extensão de um Método Ágil para Gerência e Desenvolvimento de Requisitos visando adequação ao CMMI

Alexandre Lazaretti Zanatta

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração (Sistemas de Computação) e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Raul Sidnei Wazlawick
Coordenador do Curso

Banca Examinadora:

Profª. Drª. Patrícia Vilain
Orientadora

Prof. Dr. Ricardo Pereira e Silva

Prof. Dr. Vitório Bruno Mazzola

Prof. Dr. Marcelo Soares Pimenta

continuamente vemos novidades...
tomando sempre novas qualidades...
Camões século XVI

Agradecimentos

A Deisi, minha esposa, pela confiança, pela palavra, pelo incentivo, pelo conforto nos momentos decisivos. A toda minha família pelo apoio.

Agradeço imensamente a minha orientadora professora Patrícia Vilain pela atenção, colaboração, dedicação, além de seus esclarecimentos e críticas fundamentais que me auxiliaram na elaboração deste trabalho.

Um agradecimento especial ao colega Alexandre Tagliari Lazzaretti, companheiro de inúmeras viagens, entre as cidades de Passo Fundo/RS e Florianópolis/SC, pelo estímulo e apoio. Também, ao pessoal do apartamento em Florianópolis: Luca, Bambam, Baiano e o Luiz, pelo acolhimento durante a realização deste trabalho.

A Universidade Federal de Santa Catarina (UFSC) pela estrutura, pelo corpo docente e a secretária da pós-graduação, carinhosamente conhecida por Verinha. Uma atenção ao grupo de estudo em Engenharia de Software/Banco de Dados, em particular ao professor Ronaldo S. Mello e a colega de mestrado Priscila Fagundes pelas importantes contribuições durante as discussões no grupo.

A direção, professores e funcionários do Curso de Ciência da Computação da Universidade de Passo Fundo (UPF). A equipe do Simuplan, em especial ao acadêmico Elias Müller e ao colega professor Willingthon Pavan, pela cooperação e disponibilização do local para realização do exemplo de aplicação. Também, um agradecimento ao grupo de estudo em Engenharia de Software da UPF pelas discussões sobre o tema do presente trabalho.

Aos professores Doutores Marcelo Soares Pimenta, Ricardo Pereira e Silva e Vitório Bruno Mazzola pela avaliação deste trabalho.

Aos amigos e a todos que contribuíram direta ou indiretamente neste trabalho. Obrigado a todos.

SUMÁRIO

Lista de Figuras

Lista de Abreviaturas e Siglas

ASD	*Agile Software Development*
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integration*
DoD	*Department of Defense*
DSDM	*Dynamic System Development Method*
EIA	*Electronic Industries Alliance*
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
FDD	*Feature-Driven Development*
IBM	*International Business Machine*
IEC	*International Electro Technical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
ISO	*International Organization for Standardization*
MA	Métodos ágeis
ML	*Maturity Level*
NBR	Normas Técnicas Brasileiras
NDIA	*National Defense Industrial Association*
OSD	*Office of the Under Secretary of Defense for Acquisition and Technology*
PA	*Process Area*
RAD	*Rapid Application Development*
RD	*Requirement Development*
REQM	*Requirement Management*
RUP	*Rational Unified Process*
SCAMPI	*Standard CMMI Assessment Method for Process Improvement*
SEI	*Software Engineering Institute*
SEPG	*Software Engineering Process Group*
SEPIN	Secretaria Especial de Informática
SG	*Specific Goal*
SP	*Specific Practice*
SPICE	*Software Process Improvement and Capability dEtermination*
TR	*Technical Report*
UFSC	Universidade Federal de Santa Catarina
UML	*Unified Markup Language*
UPF	Universidade de Passo Fundo
XP	*Extreme Programming*

RESUMO

Desenvolver software com qualidade não é uma tarefa trivial. Visando minimizar esta dificuldade, a engenharia de software dedica esforços no desenvolvimento de metodologias que definam e padronizem o ciclo de vida de um software, desde sua concepção até a entrega do produto final. Muitas vezes, estas metodologias são rotuladas de burocráticas e inadequadas quando aplicadas, principalmente, em organizações consideradas de menor porte. Neste sentido, surgem os métodos ágeis, que podem ser considerados como um meio termo entre a ausência e o exagero de detalhes no processo de desenvolvimento de software. Neste trabalho, inicialmente realizou-se uma análise comparativa dos métodos ágeis XP, Scrum, FDD e DSDM. Nesta análise, o Scrum obteve o melhor resultado, sendo, então, avaliado segundo as perspectivas do modelo CMMI, nas áreas de processo Gerenciamento de Requisitos e Desenvolvimento de Requisitos. Como algumas lacunas foram encontradas, fez-se uma proposta de extensão, que inclui diretrizes na busca da resolução destas lacunas existentes no método ágil Scrum. Esta extensão foi denominada de "xScrum". Por fim, as diretrizes foram aplicadas e validadas em um ambiente de desenvolvimento de software. Os dados levantados indicam que é possível utilizar métodos ágeis com o modelo CMMI, desde que a organização esteja disponível para aplicar novas perspectivas.

Palavras-chave: Scrum, CMMI, engenharia de requisitos.

ABSTRACT

Building quality software system is not a trivial task. To achieve this goal, the software engineering area is devoted to the development of new methodologies that define and standardize the software life cycle, from its conception to its final product. Often these methodologies are considered bureaucratic when applied in small organizations. Agile software development approaches have become more popular in the last few years, and can be considered a middle term between the absence and exaggeration of details in the software development process. Initially in this work, we elaborated a comparative analysis among the agile methods XP, Scrum, FDD and DSDM. In this analysis, Scrum obtained the best results, thus being evaluated according to perspectives of CMMI model, in the process areas of Requirements Management and Requirements Development. As we found some lacks in this analysis, we are proposing an extension to Scrum that includes guidelines in order to compensate for these lacks. We call this extension “xScrum”. Finally, we applied and validated the xScrum guidelines in a software development environment. The results show that is possible to use agile methods with CMMI model, as long as the organization is prepared to apply new approaches.

Key-words: Scrum, CMMI, requirements engineering.

1 Introdução

O interesse pela melhoria do processo de desenvolvimento de software tem aumentado significativamente na engenharia de software. A melhoria do processo significa compreendê-lo melhor a fim de modificá-lo, e, conseqüentemente melhorar a qualidade do produto e/ou reduzir custos e prazos no desenvolvimento. Espera-se, conforme demonstra Sommerville (2003, p.477), com a melhoria no processo de desenvolvimento de software, que a qualidade do produto final seja aperfeiçoada de modo correspondente, pois a "preocupação com a qualidade deixou de ser um diferencial competitivo e passou a ser um pré-requisito básico para participação no mercado", como destaca Côrtes (2001, p.20).

Neste sentido, adotar métodos, ferramentas ou modelos de processo que tenham sido utilizados em outra organização não significa que o processo irá melhorar. Embora organizações que desenvolvam aplicações de software tenham semelhanças, sempre existem fatores locais, assim como organizacionais e procedimentais, que influenciam no processo como um todo. Na verdade, a melhoria deve partir de uma política de gerenciamento da qualidade oriunda de gestores com comprometimento organizacional com aporte de recursos financeiros ou humanos.

Modelos de qualidade de processo de software, como o SW-CMM (*Capability Maturity Model for software)* e o CMMI (*Capability Maturity Model Integration),* preocupam-se com toda organização e tentam, de certa forma, aproximar conceitos da engenharia da produção apontados por Deming (1990) ao software, onde a habilidade e a experiência dos desenvolvedores afetam diretamente a qualidade do produto. O modelo CMMI é uma importante contribuição, mas se deve ter cuidados na sua adoção como um

modelo definitivo de capacitação de todos os processos de software, pois ele foi concebido para grandes organizações que normalmente apresentam padrões rígidos de qualidade.

Estes modelos visam fixar um padrão de desenvolvimento de software, para tentar garantir a qualidade final do sistema. Sem um padrão de desenvolvimento é difícil avaliar coerentemente um processo de software. Um exemplo da importância dos modelos de processo são as certificações, uma espécie de "atestado" que garantem a qualidade do processo de desenvolvimento de software para uma parte específica ou para o todo.

Para uma organização que busca obter algum tipo de certificação do seu processo de desenvolvimento, ter um modelo de processo bem definido é um requisito básico, pois os critérios de avaliação dessas certificações baseiam-se na padronização dos processos. Processos padronizados tendem a evitar imprevistos no desenvolvimento, facilitam na previsão dos prazos e na monitoração do andamento da construção do software.

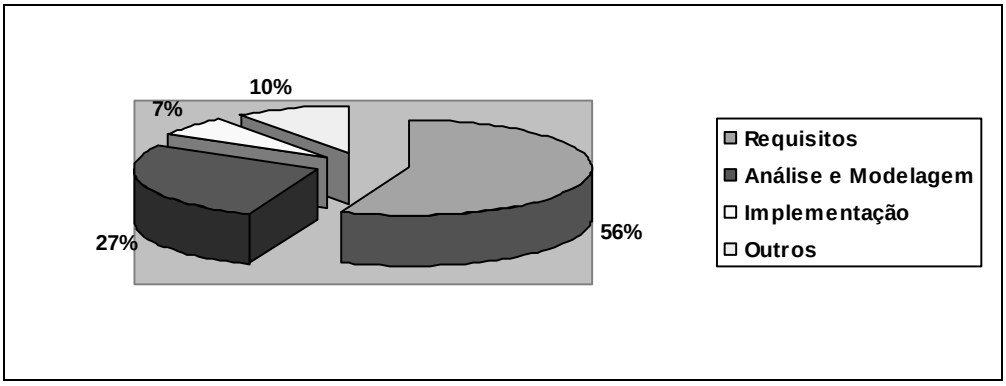
Por outro lado, os métodos ágeis surgiram como alternativa ao processo de desenvolvimento de software dirigido a pequenas organizações, abdicando, muitas vezes, de controles como, por exemplo, padronização dos processos, documentação excessiva e o gerenciamento de requisitos já presentes e consolidados em modelos como o SW-CMM. Boehm (2004, p.5) percebe uma diferenciação do entendimento da palavra qualidade utilizada nos métodos ágeis e nos modelos de qualidade. No SW-CMM, garantia da qualidade é definida como "conformidade nos processos e especificações", enquanto nos métodos ágeis, a qualidade é entendida como satisfação do cliente.

Dados do Cutter Consortium[1] apontam um crescimento de 50% na utilização dos métodos ágeis em organizações no desenvolvimento de software mundial. Boehm (2002, p.90) acredita também num significativo aumento na utilização destes métodos, mostrando que as práticas atuais da engenharia de software sofrerão mudanças estruturais, como aconteceu nos anos 90, com o desenvolvimento de software estruturado *versus* orientado a objeto.

[1] Mais informações em http://www.cutter.com/index.shtml

Segundo dados[2] da Secretaria de Política de Informática (SEPIN) do Ministério da Ciência e Tecnologia no Programa de Qualidade e Produtividade no Setor de Software no Brasil, modelos de melhoria de processo de software como o SW-CMM, apesar de serem conhecidos por 75% das organizações, apenas 21% o utilizam, demonstrando, que a maioria das organizações que desenvolvem softwares não utiliza normas ou modelos da qualidade de processos no desenvolvimento de software. Além disso, a mesma pesquisa aponta que práticas de engenharia de software adotadas no desenvolvimento e manutenção de software, como gerência de requisitos, são utilizadas por apenas 24,4% das organizações entrevistadas. Cabe ressaltar que estes dados referem-se ao SW-CMM, visto que, inexistem dados desta mesma secretaria sobre o modelo CMMI.

Bartié (2002, p.26) ressalta que a maioria dos defeitos oriundos dos projetos de softwares são resultados de análises imperfeitas de requisitos e especificações, reforçando, como mostra a Figura 1, a necessidade de investigações mais detalhadas em áreas relacionadas ao gerenciamento e desenvolvimento de requisitos. Conseguir obter os requisitos corretamente pode ser a mais importante e difícil parte de um projeto de software, como destacam Hofmann e Lehner (2001, p.1), confirmando, de certa forma, os levantamentos realizados por Bartié (2002, p.26).



Fonte: Bartié (2002, p.26)

Figura 1 Incidência de defeitos nas etapas de desenvolvimento

[2] Mais informações em http://www.mct.gov.br/Temas/info/Dsi/Quali2001/QualiProcessosSW2001.htm

O modelo CMMI também considera a área do Gerenciamento de Requisitos como uma das primeiras etapas para alcançar a maturidade organizacional, e que para ocorrer este gerenciamento é necessário que o processo de desenvolvimento de requisitos seja desenvolvido na organização.

Diante desse quadro, cabe uma investigação de como os métodos ágeis se comportam quando orientados a seguirem um modelo de melhoria no processo de software como o modelo CMMI em relação à Engenharia de Requisitos.

1.1 Objetivos

a) Objetivo Geral

Propor diretrizes a um método ágil na área de Gerenciamento e Desenvolvimento de Requisitos, visando a melhoria no seu processo de desenvolvimento de software, tendo como base o modelo CMMI.

b) Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Analisar o modelo de qualidade de processo de software CMMI;
2. Analisar e comparar métodos ágeis de desenvolvimento de software, selecionando um método para ser adequado ao modelo CMMI;
3. Definir uma proposta de extensão ao método ágil selecionado que seja compatível com o modelo CMMI exclusivamente nas áreas de processo Gerenciamento de Requisitos e Desenvolvimento de Requisitos;
4. Desenvolver um exemplo de aplicação utilizando a extensão proposta para o método selecionado em um projeto com uma pequena equipe de desenvolvimento de software.

1.2 Problematização

Orr (2002), Turner e Jain (2002), Paulk (2001), Boehm (2002), Paetshc (2003), em seus respectivos trabalhos, apontam para uma discussão recente: Desenvolvimento de software ágil é compatível com o modelo CMMI? Neste contexto, apresentam diferenças e semelhanças nas duas abordagens, acreditando que a área da engenharia de software está passando por mais uma nova fase denominada: Desenvolvimento tradicional de software *versus* Desenvolvimento de software ágil. Portanto, atualmente não existe consenso entre os principais autores se métodos ágeis são compatíveis com modelos de qualidade de software, como por exemplo o CMMI, sendo considerado um tema ainda bastante polêmico.

Turner e Jain (2002, p.161) comentam que, apesar da existência de características semelhantes entre os métodos ágeis e o modelo CMMI, ambos possuem planos específicos para o desenvolvimento de software e buscam o melhor para que a organização produza software com qualidade, algumas diferenças são apontadas e merecem destaque, conforme demonstra a Figura 2.

	Métodos ágeis	**CMMI**
Confiança do Cliente	Participação ativa durante o processo.	Infra-estrutura e maturidade do processo
Abrangência	Pequenos projetos, e focado na entrega conforme necessidade do cliente.	Pode ser aplicado em qualquer tipo de projeto e ambiente de desenvolvimento
Escalabilidade	Ideal para aplicação em pequenas organizações. Em organizações maiores, sua aplicação é questionada.	Organizações grandes e complexas. Não é indicado para pequenas organizações.
Metas	Rapidez e eficiência na entrega do software ao cliente.	Previsibilidade e estabilidade não importando o produto a ser desenvolvido ou equipe de trabalho.

Fonte: Adaptado de Turner (2002, p. 161)

Figura 2 Diferenças entre métodos ágeis e o modelo CMMI.

Por outro lado, Paulk (2001, p.26) acredita que o modelo CMMI pode ser utilizado em qualquer ambiente com qualquer projeto independente do tamanho, pois CMMI “diz o que deve ser feito em termos gerais, mas não diz como deve ser feito”, com isso, pode se

adaptar “facilmente” em projetos de desenvolvimento de software. Além disso, Paulk conclui que tanto o modelo CMMI quanto os métodos ágeis são “boas idéias” e podem trabalhar conjuntamente, uma completando a outra.

Os métodos ágeis no desenvolvimento de software, segundo Paetsch (2003, p.313) abdicam em parte da documentação e do gerenciamento dos requisitos. O autor indica, ainda, que as técnicas descritas nestes métodos são muitas vezes vagas ficando a implementação exclusivamente para o desenvolvedor.

Nos estudos de Ornburn (2002, p.1), os métodos ágeis são compatíveis com o modelo CMMI. Porém, são necessárias inclusões de algumas práticas nos métodos ágeis para total compatibilização, como, por exemplo, estes métodos satisfazem parcialmente as áreas de processo como Gerenciamento e Desenvolvimento de Requisitos do modelo CMMI. Neste estudo, o autor não apresenta que práticas poderiam ser incluídas, para total compatibilização com o modelo CMMI. Paetsch (2003, p.313) reforça que são superficiais e vagas as técnicas utilizadas no processo de desenvolvimento de software nos métodos ágeis, principalmente no que se refere ao Gerenciamento de Requisitos, devido principalmente a ausência de documentação. Emergem algumas questões ao tentar comparar modelos de qualidade de software e métodos ágeis:

1. Os modelos de qualidade e os métodos ágeis, podem coexistir em um mesmo ambiente obtendo os benefícios propostos por ambos?
2. Empresas que buscam ou que são certificadas pelos modelos de qualidade existentes, como o modelo CMMI, podem optar pelo uso de um método ágil de forma que não comprometa a certificação realizada?

1.3 Organização do presente trabalho

Este trabalho é composto por 7 capítulos organizados da seguinte forma. O primeiro capítulo apresenta a motivação, objetivos gerais e específicos do trabalho e finalmente a problematização.

No segundo capítulo, é apresentado o modelo CMMI, mostrando a sua importância e apresentando os seus conceitos. Logo a seguir, apresentam-se as duas representações do modelo CMMI, a Contínuo e Estágio, com suas definições e características peculiares. Também, apresenta os métodos ágeis, abordando conceitos fundamentais e características presentes, onde são apresentados, em detalhes, quatro métodos: Scrum, XP, FDD e DSDM. Realiza-se um estudo comparativo entre os métodos estudados, apontando semelhanças e diferenças. Ao final do capítulo são apresentados os trabalhos relacionados dos métodos ágeis quando comparados com modelos de qualidade como por exemplo, o CMMI.

O terceiro capítulo apresenta uma análise detalhada do Scrum segundo as perspectivas do modelo CMMI especificamente na categoria Engenharia com as áreas de processo Gerenciamento de Requisitos e Desenvolvimento de Requisitos.

Realiza-se, no quarto capítulo, o detalhamento da proposta de extensão do método Scrum. Para cada prática não atendida ou parcialmente atendida pelo Scrum na análise realizada é proposta uma diretriz. No capítulo 5, faz-se um relato do exemplo de aplicação da solução proposta no capítulo 3 em um local de desenvolvimento de software. Também são apresentados os resultados deste exemplo de aplicação.

Finalmente, no capítulo 6 apresentam-se as conclusões do trabalho, principais contribuições e alguns problemas encontrados, e no capítulo 7 as referências bibliográficas.

2 Áreas Relacionadas

Este capítulo apresenta as áreas relacionadas para o desenvolvimento do presente trabalho. Inicialmente fornece uma visão geral do modelo CMMI. Em seguida mostra os métodos ágeis XP, FDD, DSDM e o Scrum. Finalmente, apresenta os trabalhos relacionados.

2.1 Capability Maturity Model Integration

Esta seção primeiramente fornece uma visão geral do modelo CMM explicando, também, a origem do novo modelo CMMI. Em seguida aborda o modelo CMMI, apresentando suas características e suas duas representações, Estágio e Continuo. Finaliza com uma breve análise da relação entre os perfis de maturidade nas duas representações.

O CMM (*Capability Maturity Model for software*), ou em português Modelo de Maturidade da Capacitação para software, é, segundo Fiorini (1998, p.3) "um caminho gradual que leva organizações a se aprimorarem continuamente na busca da sua própria solução dos problemas inerentes ao desenvolvimento de software". O modelo CMM também deve atender as necessidades daqueles que realizam melhoria e avaliação do processo de software, refletindo as melhores práticas.

Ainda segundo Fiorini (1998, p. 15) citando Paulk, CMM é

> um *framework* que descreve os principais elementos de um processo de software efetivo. O CMM descreve os estágios das quais organizações de software evoluem quando elas definem, implementam, medem, controlam e melhoram seus processos de software. O CMM fornece uma diretriz para a seleção de estratégias de melhoria do processo, permitindo a determinação da capacitação dos processos correntes e a conseqüente

> identificação das questões mais críticas para a melhoria do processo e qualidade de software.

Heinz (2003) afirma que o modelo CMM vem sendo o padrão global para melhoramento de processo de software, sendo um dos modelos mais antigos relacionados à qualidade do processo de desenvolvimento de software. Porém, esse modelo, após uma década, sofreu melhorias que foram agrupadas, gerando o modelo SW-CMM (CMM software), mais popularmente conhecido apenas como CMM. Estas melhorias não foram suficientes para atender diversas áreas de uma organização, levando o *Software Engineering Institute*[3] (SEI) a desenvolver "outros" CMMs a seguir explicados:

a) *Software Acquisition* CMM (SA-CMM): é utilizado para avaliar a maturidade de uma organização em seus processos de seleção, compra e instalação de software desenvolvido por terceiros;

b) *Systems Engineering* CMM (SE-CMM): avalia a maturidade da organização em seus processos de engenharia de sistemas (hardware, software, etc), concebidos como algo maior que o software;

c) *Integrated Product Development* CMM (IPD-CMM): ainda mais abrangente que SE-CMM, inclui também outros processos necessários à produção e suporte ao produto;

d) *People* CMM (P-CMM): avalia a maturidade da organização em seus processos de administração de recursos humanos no que se refere a software, recrutamento e seleção de desenvolvedores, treinamento, desenvolvimento, remuneração, etc.

Com estes vários modelos SW-CMM, diferenças começaram a surgir. Diferenças como por exemplo, falta de padronização na utilização de terminologias ou, todos modelos tinham formas distintas de avaliação, gerando aumento de custos de treinamento.

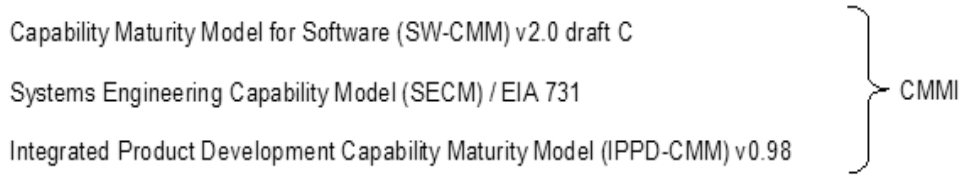
[3] Instituto responsável pelo desenvolvimento da primeira versão do CMM, mais informações em http://www.sei.cmu.edu

Outra diferença, ocorreu com o surgimento do projeto SPICE, acrônimo de *Software Process Improvement Capabitlity dEtermination,* atual norma ISO/IEC 15504, que apresenta um modelo de referência que permite que os diversos modelos de processo existentes e outros que venham ser desenvolvidos, possam ser definidos como modelos compatíveis, possibilitando que os resultados das avaliações, segundo cada um deles, possam ser comparados. Segundo Rocha, (2001, p.28), o projeto SPICE "presta-se à realização de avaliações de processos de software com dois objetivos: a melhoria do processo e a determinação da capacidade de processos de uma organização".

Se o objetivo da organização for, por exemplo, obter a melhoria de processos, ela realiza uma avaliação gerando um perfil de processos para elaborar um plano de ação de melhorias, definindo os objetivos e contextos para a avaliação, os modelos e métodos para avaliação e as melhorias desejadas. Caso o objetivo for avaliar um fornecedor, obtendo seu perfil de capacidade, ela deve definir quais são os objetivos e contextos para a avaliação, os modelos, métodos e os requisitos esperados.

Portanto, para resolver estas diferenças o SEI abandonou o projeto SW-CMM 2 que estava em andamento para atualização do CMM, e iniciou o chamado CMMI[4] (CMM Integration). Este projeto, contou com a colaboração de diversos segmentos da sociedade, como a indústria e o governo norte-americano, sendo patrocinado pelo DoD *(U.S. Department of Defense)*, mais especificamente pelo OSD (*Office of the Under Secretary of Defense for Acquisition and Technology*) e NDIA (*National Defense Industrial Association*), tendo como objetivo, gerar uma nova versão do CMM buscando resolver essas diferenças, ou seja, integrando os diversos CMMs numa estrutura única com mesma terminologia, processos de avaliação e estrutura, e tornando-o também compatível com a norma ISO/IEC 15504. Pode-se observar a integração conforme demonstra a Figura 3.

[4] CMM and Capability Maturity Model são marcas registradas no *U.S. Patent and Trademark Office.*
CMM Integration, CMMI, SCAMPI, and IDEAL são marcas de serviço da Carnegie Mellon University



Fonte: primária.

Figura 3 Integração dos vários modelos

Outra finalidade do CMMI é dar suporte ao processo e à melhoria do produto, reduzir redundâncias e eliminar inconsistências que são observadas pelos usuários que utilizam o modelo SW-CMM. Ainda conforme Heinz (2003), "seu propósito é de prover uma direção para uma organização aprimorar seus processos e sua habilidade de gerenciar o desenvolvimento, aquisição e manutenção de produtos e serviços". Para Chrissis *et al.* (2003, p. 18) "o propósito do CMMI é estabelecer um guia para melhorar o processo da organização e sua capacidade para gerenciar o desenvolvimento, aquisição e manutenção de produtos e serviços". Ainda segundo os autores, o CMMI traz alguns benefícios de melhoria de processo como:

a) Clareza de foco na melhoria facilitando a inclusão dos pontos críticos da organização;

b) Integração de processos: efeito da integração na organização;

c) Flexibilidade e extensão: a facilidade em adicionar extensões (explicadas a seguir) à medida que muda o ambiente de negócio.

Ahern *et al.* (2004, p.46), apresentam os principais objetivos do modelo CMMI a seguir destacados:

a) suprir as limitações do CMM, criando um *framework* comum eliminando inconsistências e permitindo a inclusão de novos modelos ao longo do tempo, sempre que surgirem necessidades específicas;

b) preservar os investimentos já realizados por órgãos governamentais, organizações privadas, fornecedores e indústria no processo de transição;

c) unificar os vários modelos CMM existentes;

d) implementar melhorias a partir das experiências adquiridas com projetos já implementados;

e) assegurar a integridade com a norma ISO/IEC 15504 permitindo a análise de áreas independentes do nível de maturidade;

f) reduzir custos de treinamento, da implementação de melhorias, da formação de avaliadores oficiais e das próprias avaliações oficiais.

O modelo CMMI possui 4 (quatro) disciplinas, também conhecidas como áreas do conhecimento, que auxiliam no planejamento da melhoria do processo de toda organização. Cada organização, deverá optar por uma ou mais disciplinas caso necessite iniciar o processo de melhoria. A seguir uma descrição das 4 (quatro) disciplinas.

a) Engenharia de Sistemas: refere-se ao desenvolvimento de sistemas em geral, incluindo ou não o software. Focaliza na transformação das necessidades, expectativas e restrições do cliente em soluções de produto, com o suporte durante o ciclo de vida do produto.

b) Engenharia de Software: trata especificamente da aplicação de abordagens sistemáticas, disciplinadas, quantificadas ao desenvolvimento, operação e manutenção de software.

c) Produto Integrado e Desenvolvimento de Processo: é um acompanhamento sistemático e colaborativo do ciclo de vida do produto para satisfazer as expectativas e requisitos dos usuários;

d) Aquisição: projetos podem necessitar de terceiros para executarem tarefas adicionais à medida que se tornam mais complexos. Esta disciplina tem como objetivo controlar a aquisição de produtos realizados por terceiros, principalmente quando estas atividades forem consideradas como críticas.

A implementação de uma ou mais destas disciplinas ao mesmo tempo com uma única terminologia e infra-estrutura de treinamento e avaliação, é considerada uma grande vantagem do modelo CMMI, pois a organização determina em quais disciplinas deseja

melhorar seu processo. Estas disciplinas são compostas por áreas de processo[5] que, quando executadas, determinam a melhoria do processo na disciplina escolhida.

Caso exista a necessidade da melhoria nos processos da disciplina de Engenharia de Sistemas é necessário satisfazer as 22 (vinte e dois) seguintes áreas de processo:

a) Definição do processo organizacional;
b) Desenvolvimento de Requisitos;
c) Estratégia e inovação organizacional;
d) Desempenho do processo da organização;
e) Foco no processo da organização;
f) Garantia da qualidade de processo e produto;
g) Gerenciamento de Requisitos;
h) Gerenciamento quantitativo do projeto;
i) Gerenciamento de configuração;
j) Gerenciamento de acordos de fornecimento;
k) Gerenciamento integrado de projeto;
l) Gerenciamento de risco;
m) Integração de produto;
n) Medições e análises;
o) Controle e monitoramento de projeto;
p) Planejamento de projeto;
q) Análises causais e resolução;
r) Análise de decisão e resolução;
s) Solução técnica;
t) Treinamento organizacional;
u) Validação;
v) Verificação;

No ANEXO 1, encontra-se uma descrição das áreas de processos Gerenciamento de Requisitos e Desenvolvimento de Requisito. As demais áreas não serão detalhadas devido ao enfoque do presente trabalho.

Se a necessidade for somente na disciplina de Engenharia de Software, a organização deverá satisfazer as mesmas áreas de processo da disciplina de Engenharia de Sistemas atendendo às amplificações das disciplinas. Amplificação de disciplina são modelos de

5 Área de Processo é um conjunto de práticas relacionadas em uma área que quando executadas coletivamente satisfazem um conjunto de objetivos importantes para a melhoria significante daquela área. (Fiorini, 1998, p.29)

componentes de caráter informacional, contendo dados relevantes para uma disciplina específica sendo associadas a práticas específicas. Por exemplo, na área de processo Gerenciamento de Requisitos, no SG-1 *(Specific Goal -1)* a amplificação da disciplina para engenharia de software é a seguinte: “os requisitos devem ser um subconjunto dos todos requisitos do produto, ou eles devem constituir por completo todos os requisitos”.

Caso a necessidade da organização seja na disciplina Produto Integrado e Desenvolvimento de Processo, ela deverá satisfazer as mesmas áreas de processo da disciplina de Engenharia de Sistemas com atenção as amplificações das disciplinas e acrescentando duas áreas de processo a seguir destacadas:

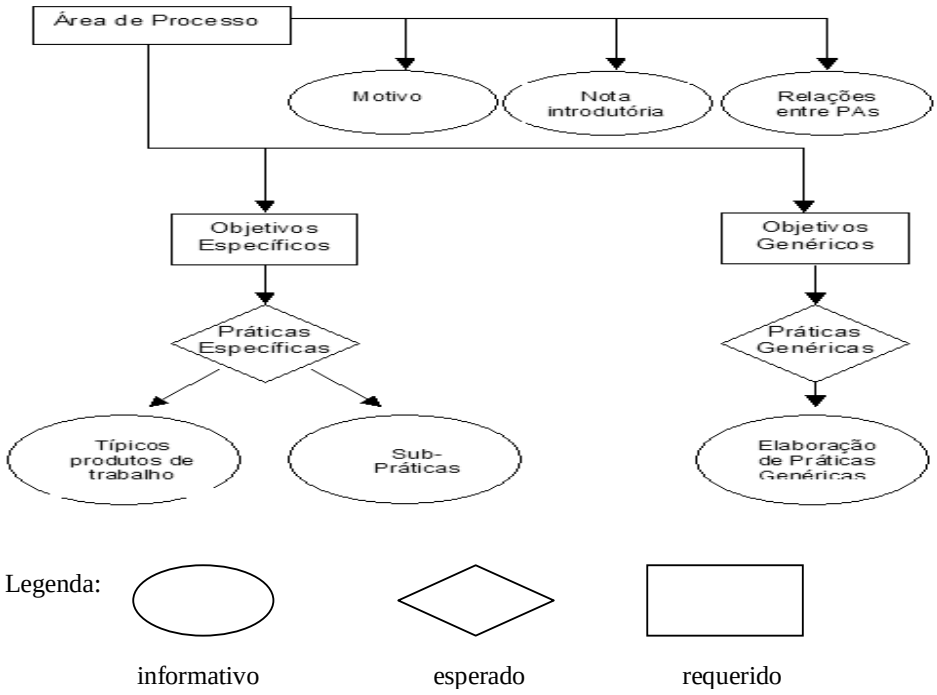
a) Ambiente organizacional para integração;
b) Equipes integradas.

Se a necessidade da organização é na disciplina Aquisição, deverá também satisfazer as mesmas áreas de processo da disciplina de Engenharia de Sistemas com atenção as amplificações das disciplinas e acrescentando a área de processo denominada Gerenciamento Integrado dos Fornecedores.

Caso a organização opte por todas disciplinas, ela deverá escolher as 25 (vinte e cinco) áreas de processo com especial atenção nas amplificações das disciplinas. Cabe ressaltar que a única diferenciação entre as disciplinas Engenharia de Sistemas e Engenharia de Software é justamente as amplificações. Nas demais disciplinas, ocorre a inclusão de outras áreas de processo.

Cada área de processo é composta por vários componentes separados em três categorias: Requeridos, Esperados e Informativos. (i) Os componentes requeridos descrevem o que uma organização deve satisfazer para alcançar determinada área. (ii) Os componentes esperados, descrevem tipicamente o que uma organização alcançará caso obtenha um componente requerido. (iii) Os informativos fornecem os detalhes que auxiliam

a organização no entendimento para obtenção dos componentes requeridos e esperados. A Figura 4 demonstra os componentes do modelo CMMI.



Fonte: Adaptado de Chrissis *et al.* (2003, p.22)

Figura 4 Componentes do Modelo CMMI.

Os componentes da categoria requeridos são os objetivos específicos e objetivos genéricos. Objetivos específicos descrevem características únicas que devem estar presentes para satisfazer determinada área de processo, por exemplo, um objetivo específico da área de processo "Gerenciamento de configuração" é "integração das *baselines*[6] é mantido e estabelecido". Os objetivos genéricos referem-se a todas as áreas de processo, não constando em áreas específicas, onde normalmente "aparecem" no final de cada área de processo. Um exemplo de um objetivo genérico é "Institucionalizar um processo gerenciado".

6 *Baseline* é um conjunto de requisitos aprovados e revisados que representam o embasamento de um acordo de desenvolvimento.

Os componentes da categoria esperados são: práticas específicas e práticas genéricas. Prática específica é uma descrição detalhada das atividades que são consideradas fundamentais para alcançar os objetivos específicos. Um exemplo de prática específica é a "Obter o entendimento dos requisitos", da área de processo Gerenciamento de Requisitos. Práticas genéricas se posicionam ao final de cada área de processo e são chamadas de genéricas, pois estão em várias áreas de processo. Uma prática genérica é a descrição de uma atividade fundamental para alcançar os objetivos genéricos. "Estabelecer uma política Organizacional", é um exemplo da prática genérica também da área de processo Gerenciamento de Requisitos.

E os componentes da categoria informativa são: típicos produtos de trabalho, sub-práticas e elaboração de práticas genéricas. O típico produto de trabalho é uma lista simples das saídas produzidas pelas práticas específicas. Por exemplo, um típico produto de trabalho da prática específica "Estabelecer e manter procedimentos e critérios para a seleção de produtos de trabalho" na área de processo Verificação é "critérios de verificação". Sub-prática é uma descrição detalhada para implementação e interpretação de uma prática específica, ou seja, contribuem efetivamente para o entendimento das práticas específicas. Cabe salientar que as sub-práticas não são obrigatórias. Um exemplo de uma sub-prática da prática específica "Agir Corretivamente na identificação dos assuntos" na área Monitoramento do Projeto e Controle do Processo é "Determinar e documentar ações apropriadas necessárias ao alcance na identificação dos assuntos". O componente, Elaboração de Práticas Genéricas situa-se logo após as práticas genéricas numa área de processo para fornecer um guia de como essas práticas devem ser aplicadas.

A categoria informativa possui outros três componentes: motivo, notas introdutórias e relação entre as áreas de processos, que basicamente descrevem características de cada área de processo.

O modelo CMMI, segundo o SEI, Chrissis *et al.* (2003, p. 18) e Ahern *et al.* (2004, p.46), ainda é dividido em duas representações: Estágio e Contínuo. A Representação Estágio fornece uma seqüência de melhoramentos com práticas de gerenciamento e

processos, além de proporcionar uma migração “facilitada” do SW-CMM para o modelo CMMI. Cada nível (estágio) possui diversas áreas de processos onde cada uma se encontra em um único nível. A Representação Contínuo, é baseada na norma ISO/IEC15504, possuindo 6 (seis) níveis de maturidade, onde qualquer área do processo pode ter sua maturidade avaliada em alguns desses níveis, além de facilitar também uma possível migração da norma EIA/IS-731 da SECM[7]. Esta norma, a EIA/IS-731, está organizada de forma muito similar ao CMMI, como por exemplo, na separação por categorias e práticas específicas.

Como ocorre nas disciplinas, se faz necessário à escolha por parte da organização de uma das representações sendo que esta decisão dependerá das necessidades que melhor convier a cada organização. A seguir, apresenta-se inicialmente o modelo CMMI na Representação Estágio, e logo após na Representação Contínuo.

2.1.1 CMMI – Representação Estágio

Esta representação, segundo o SEI TR-029 (2002), organiza as áreas do processo em 5 (cinco) níveis de maturidade, assim como o SW-CMM. Estes níveis representam um caminho para a melhoria do processo através da evolução dos processos da organização. Para Chrissis *et al.* (2003, p. 79) e Ahern *et al.* (2004, p.73), os níveis de maturidade da Representação Estágio são:

a) Nível 1: Inicial - neste nível o processo é improvisado, sua capacidade é imprevisível, e depende muito do esforço pessoal;

b) Nível 2: Gerenciado - assegura que os requisitos são gerenciados e os processos são planejados, executados, medidos e controlados;

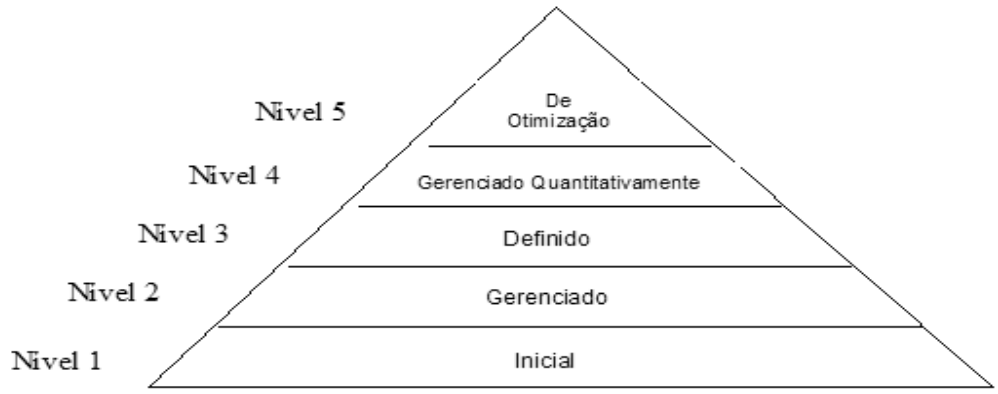
c) Nível 3: Definido - o processo operacional está definido, sendo capaz de atingir metas de funcionalidade, prazo e custo;

d) Nível 4: Gerenciado Quantitativamente - o processo é continuamente medido através de técnicas estatísticas;

7 Clouse, Aaron. Transition from EIA/IS-731 to CMMI
disponível em < http://www.stsc.hill.af.mil/crosstalk/2000/07/clouse.html >

e) Nível 5: De Otimização – o melhoramento contínuo do processo é implantado e baseado no entendimento quantitativo das causas da variação dos processos.

Os níveis de maturidade que normalmente são apresentados sob forma de uma pirâmide, como demonstra a Figura 5, indicam que para alcançar um nível superior, todas as áreas de processo no nível inferior deverão ser satisfeitas. Por exemplo, para que determinada organização possua nível de Maturidade 3, ela deverá ter cumprido todas as áreas de processo do nível 2, e assim por diante.



Fonte: Adaptado de Chrissis *et al.* (2003)

Figura 5 Níveis de maturidade na Representação Estágio.

O nível de maturidade um (1) ou Inicial é o único que não possui áreas de processos, pois normalmente nesse nível os processos dependem muito mais dos esforços individuais das pessoas envolvidas no processo do que um processo planejado.

O nível de maturidade 2 ou Gerenciado é composto pelas seguintes áreas do processo:

a) Controle e monitoramento do projeto;
b) Garantia da qualidade de processo e produto;
c) Gerenciamento da configuração;
d) Gerenciamento de acordos de fornecimento;
e) Gerenciamento de Requisitos;
f) Medições e análises;
g) Planejamento de projeto.

O nível de maturidade 3 ou Definido, que possui o maior número de áreas do processo, apresenta as seguintes áreas:

a) Ambiente organizacional para integração;
b) Análise de decisão e resolução;
c) Definição do Processo Organizacional;
d) Desenvolvimento de Requisitos;
e) Equipes integradas;
f) Foco no processo da organização;
g) Gerenciamento de risco;
h) Gerenciamento Integrado de fornecedores;
i) Gerenciamento integrado do projeto;
j) Integração de produto;
k) Solução técnica;
l) Treinamento organizacional;
m) Validação;
n) Verificação.

O nível de maturidade 4 ou Gerenciado Quantitativamente, é composto pelas seguintes áreas do processo:

a) Gerenciamento quantitativo do projeto;
b) Definição do processo organizacional.

Por fim, o nível de maturidade 5 ou De Otimização, possui as áreas de processo:

a) Análises causais e resolução;
b) Estratégia e inovação organizacional.

Além da utilização dos componentes do modelo CMMI para a efetiva aplicação de área de processo, a Representação Estágio utiliza mais 4 (quatro) características comuns para organizar as práticas genéricas. As características comuns, para Fiorini (1998, p.36), são “características que indicam se a implementação (atividades) ou a institucionalização (infra-estrutura) de uma área de processo são efetivas, repetíveis e duradouras”. Possuem como função básica, organizar as práticas-chave de uma área de processo e, segundo o SEI TR-029 (2002), são classificadas em:

a) compromisso em executar: descreve as ações que a organização deve adotar para garantir que o processo está estabelecido e irá continuar envolvendo políticas organizacionais e liderança.

b) habilidade para executar: descreve as pré-condições que devem existir no projeto ou organização para implementar o processo de software de forma competente. Normalmente envolve recursos, estruturas organizacionais e treinamento.

c) direção da implementação*:* descreve como será implementado cada processo dentro da organização.

d) verificação da implementação*:* descreve as etapas para assegurar que as atividades são executadas de acordo com o processo que foi estabelecido. Abrange revisões e auditorias pela gerência e garantia da qualidade de software.

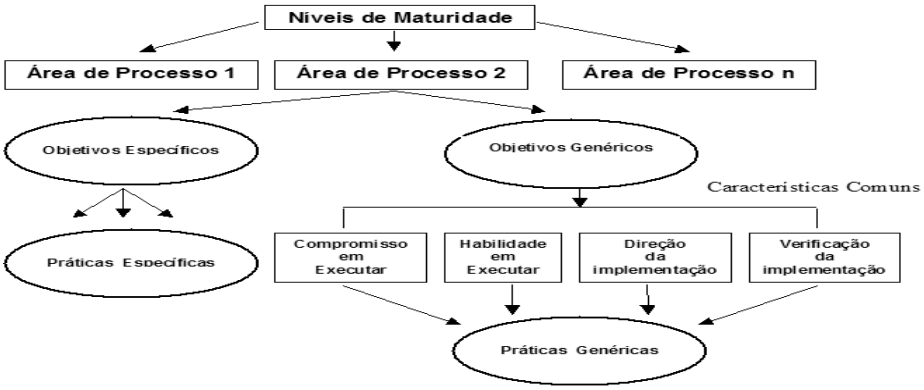
Ainda, as características comuns, exceto Direção da Implementação e Verificação da Implementação, são divididas por práticas genéricas. A seguir, são apresentadas as seguintes práticas genéricas das Características Comuns: Compromisso em Executar e Habilidade para Executar.

a) Compromisso em Executar:
    1. estabelecimento de uma política organizacional.

b) Habilidade para Executar:

1. plano de processos;
2. fornecimento de recursos;
3. determinação das responsabilidades;
4. treinamento de pessoas;
5. estabelecimento de um processo definido;
6. gerenciamento de configuração;
7. identificação e envolvimento dos supervisores;
8. monitoramento e controle dos processos;
9. melhoramento da coleção das informações;
10. avaliação objetiva do que foi planejado.

O modelo CMMI na Representação Estágio é ilustrado na Figura 6, onde os níveis de maturidade organizam as áreas de processo que contém os objetivos genéricos e específicos e as práticas genéricas e específicas. Cabe ressaltar que as características comuns organizam as práticas genéricas.



Fonte: Adaptado de SEI-TR029 (2002, p.10)

Figura 6 Componentes do Modelo na Representação Estágio

Um objetivo genérico descreve “o que” a organização deve executar para satisfazer uma área de processo na representação Estágio, sendo que cada área de processo possui somente um objetivo genérico. O objetivo genérico de cada área de processo depende do nível de maturidade que está colocado, por exemplo, o nível 2 possui o Objetivo Genérico – Instituir o Processo Gerenciado, enquanto o nível 3 possui o Objetivo Genérico - Instituir o Processo Definido.

2.1.2 CMMI - Representação Contínuo

O modelo CMMI na Representação Contínuo, conforme o SEI TR-028 (2002, p.33), foca a mensuração de melhoria de processos usando níveis de capacitação. Níveis de capacitação são aplicados à realização de melhoria do processo através das áreas de processos individuais, como por exemplo, a área de Gerenciamento de Requisitos. Segundo Fiorini (1998, p.16), capacitação do processo “é o intervalo ou faixa de tolerância dos resultados esperados, que podem ser alcançados seguindo um processo de software”.

De acordo com Chrissis *et al.* (2003, p. 76), Ahern *et al.* (2004, p.89), o modelo CMMI na representação Contínuo tem 6 (seis) níveis para dimensão da capacitação, de 0 (zero) a 5 (cinco), assim descritos:

a) Nível 0 : Incompleto – processo executado ou executado parcialmente;

b) Nível 1 : Executado – satisfaz metas específicas da área de processo;

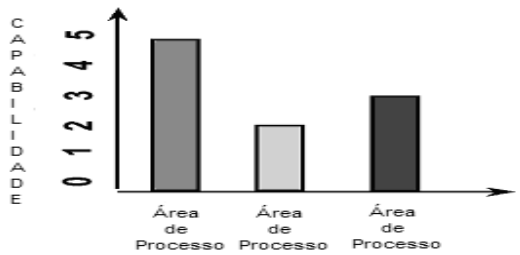
c) Nível 2 : Gerenciado – processo executado e também planejado, monitorado e controlado para atingir um objetivo;

d) Nível 3: Definido – processo gerenciado, adaptado de um conjunto de processos padrão da organização;

e) Nível 4 : Gerenciado Quantitativamente – processo definido, controlado utilizando estatísticas ou outras técnicas quantitativas;

f) Nível 5 : De Otimização – processo gerenciado quantitativamente para a melhoria contínua do desempenho do processo.

Diferentemente na representação Estágio, as áreas de processo na representação Contínuo são independentes dos níveis de maturidade, ficando relacionadas apenas com a capabilidade[8] do processo, ou seja, uma determinada área de processo em particular poderá ter sua capacidade avaliada independente das outras áreas de processo. Pode-se citar como exemplo, que a área de processo Gerenciamento de Requisitos poderá obter o nível de capacidade 3, e as demais áreas de processo poderão alcançar outros níveis de capacidade. A Figura 7, ilustra o modelo CMMI na representação Contínuo.



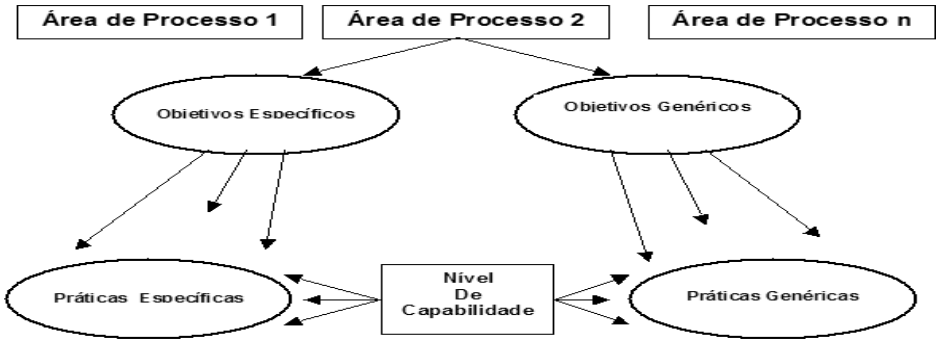
Fonte: Adaptado de http://www.sei.cmu.edu/cmmi

Figura 7 Representação Contínuo no modelo CMMI.

8 O termo Capabilidade não existe no vocabulário português, por esse motivo esse termo será substituído por capacidade.

Cabe ressaltar que o nível de capacidade é acumulativo, ou seja, nenhuma área de processo poderá receber um nível superior sem ter alcançado o nível inferior. Então, qualquer área de processo pode ter a sua capacidade avaliada em um desses seis níveis. Para permitir a independência entre essas duas dimensões[9], segundo Côrtes (2001, p.133), a representação Contínuo "tem objetivos e práticas genéricas, associados aos níveis e dissociados das áreas de processo, e objetivos e práticas específicas, associados às áreas de processo e dissociados dos níveis".

A Figura 8, ilustra a representação Contínuo, onde objetivos específicos organizam as práticas específicas e os objetivos genéricos organizam práticas genéricas. Cada prática genérica e específica corresponde ao nível de capacidade. Objetivos específicos e práticas específicas referem-se às áreas de processos individuais.



Fonte: Adaptado de SEI-TR028 (2002, p.10)

Figura 8 Modelo CMMI na Representação Contínuo

Objetivos genéricos e práticas genéricas referem-se a várias áreas de processo, e definem uma seqüência de níveis de capacidade que representam melhorias na implementação e efetividade para todos os processos definidos para a melhoria. No modelo CMMI na representação Contínuo, níveis de capacidade fornecem uma recomendação para melhorar os processos dentro de cada área de processo possuindo flexibilidade para determinar qual área de processo será abordada para a melhoria.

9 terminologia utilizada somente pela ISO/IEC 15504

2.1.2.1 Áreas de processo na representação Contínuo

As áreas de processo, de maneira semelhante a ISO/IEC 12207[10], conforme descritas por Rocha (2001, p.10), são separadas em 4 (quatro) categorias ou grupos: Gerenciamento de Processo, Gerenciamento de Projeto, Engenharia e Apoio. A seguir, estas categorias são apresentadas, de acordo com o SEI TR-028 (2002) e Ahern *et al.* (2004, p.113):

- **Gerenciamento de Processo**

Contém atividades que são aplicadas entre os projetos para a definição, planejamento, desenvolvimento, implementação, monitoração, controle, avaliação, medidas e melhoria do processo. As áreas de processo da categoria Gerenciamento de Processo, conforme Ahern *et al.* (2004, p.114), são:

a) Definição no processo organizacional;
b) Foco no processo da organização;
c) Desempenho do processo da organização.
d) Estratégia e inovação organizacional;
e) Treinamento organizacional;

- **Gerenciamento de Projeto**

São responsáveis pelo gerenciamento das atividades de projeto relacionadas ao planejamento, monitoria e controle do projeto. As áreas de processo da categoria Gerenciamento de Projeto conforme Ahern *et al.* (2004, p.114), são destacadas a seguir:

a) Planejamento de projeto;
b) Controle e monitoramento do projeto;
c) Gerenciamento integrado de projeto;
d) Gerenciamento quantitativo do projeto;
e) Gerenciamento de acordos de fornecimento;
f) Gerenciamento de risco.

---

10 International Organization for Standardization and International Electro-technical Commission. ISO/IEC 12207 Information Technology-Software Life Cycle Process, 1995. Disponível em <http://www.iso.ch>

- **Engenharia**

As áreas de processo da categoria Engenharia cobrem o desenvolvimento e manutenção das atividades da engenharia. As áreas de processo desta categoria, conforme Ahern *et al*. (2004, p.125), são as seguintes:

a) Gerenciamento de Requisitos;
b) Desenvolvimento de Requisitos;
c) Integração do produto;
d) Solução técnica;
e) Verificação;
f) Validação;

- **Apoio**

Essa categoria baseia-se principalmente no suporte às atividades de manutenção e desenvolvimento de produtos. As áreas de processo da categoria apoio, conforme Ahern *et al*. (2004, p.134) são:

a) Gerenciamento da configuração;
b) Garantia da qualidade de processo e produto;
c) Medições e análises;
d) Análise de decisão e resolução;
e) Análises causais e resolução;
f) Ambiente organizacional para integração.

2.1.2.2 Práticas genéricas na representação Contínuo

Segundo Côrtes (2001, p.134), “as práticas genéricas do modelo CMMI na representação Contínuo estão associadas aos níveis de maturidade e são ortogonais às áreas de processo”. Para exemplificação, as práticas genéricas, que possuem sub-práticas, do nível 2 são apresentadas a seguir:

a) Definir e atribuir responsabilidades.
b) Estabelecer uma política organizacional para o planejamento e execução do processo.
c) Executar e gerenciar o processo.
d) Gerenciar configurações e versões de produtos de trabalho selecionados.
e) Monitorar e controlar a execução do processo.

f) Planejar o processo.
g) Prover os recursos necessários para a execução do processo.
h) Providenciar o treinamento necessário para as pessoas executarem o processo.
i) Submeter à análise gerencial os resultados e atividades do processo.
j) Verificar a conformidade da execução do processo e dos produtos de trabalho com os procedimentos e padrões.

2.1.2.3 Práticas específicas na representação Contínuo

Ainda segundo Côrtes (2001, p.135), "as práticas específicas na representação Contínuo não são agrupadas por classe como na representação Estágio, mas agrupadas por objetivo". Para exemplificação, as práticas específicas (PE), ou em inglês *Specific Practice (SP)*, da área de Processo Planejamento de Projeto são apresentadas a seguir:

- Objetivo 1: Estabelecer estimativas

  - PE 1: Estabelecer as tarefas do projeto e responsabilidades associadas.
  - PE 2: Estimar atributos do projeto.
  - PE 3: Determinar esforço e custo.

- Objetivo 2: Elaborar planos de projeto

  - PE1: Definir o ciclo de vida do projeto.
  - PE2: Estabelecer e manter cronogramas.
  - PE3: Estabelecer planos subordinados.
  - PE4: Identificar os riscos do projeto.
  - PE5: Planejar os conhecimentos e capacitações necessárias para o projeto.
  - PE6: Planejar a coleta de dados da execução do projeto.
  - PE7: Estabelecer e manter os planos.

- Objetivo 3: Obter comprometimento com os planos

  - PE1: Conciliar o plano para acomodá-lo aos recursos disponíveis.
  - PE2: Coordenar o comprometimento de indivíduos e organizações aos planos.
  - PE3: Coordenar e analisar os planos com as partes interessadas.

2.1.3 Relação entre os perfis de maturidade do Contínuo e o Estágio

Para atingir um determinado nível na representação estágio, é necessário que todas as áreas de processo associadas àquele nível e aos níveis inferiores satisfaçam as práticas genéricas daquele nível e dos níveis inferiores. Desta forma, Côrtes (2001, p.135)

recomenda que para se determinar qual é o perfil de maturidade necessário para a organização atingir um determinado nível, deve-se, na representação Contínuo, observar o seguinte:

a) Para que a organização alcance o nível 2 da representação Estágio, é necessário que as 7 (sete) áreas de processo associadas ao nível 2 na representação Estágio satisfaçam as práticas genéricas dos níveis 1 e 2.

b) Para que a organização alcance o nível 3 na representação Estágio é necessário que as primeiras 7 (sete) áreas de processo associadas ao nível 2 mais as 14 (quatorze) áreas de processo associadas ao nível 3 na representação Estágio satisfaçam as práticas genéricas dos níveis 1, 2 e 3.

c) Para alcançar o nível 4 na representação Estágio , é necessário que as 7 (sete) áreas de processo associadas ao nível 2, mais as 14 (quatorze) áreas de processo associadas ao nível 3, mais as 2 (dois) associadas ao nível 4 na representação Estágio satisfaçam as práticas genéricas dos níveis 1, 2 e 3.

d) Para que a organização alcance o nível 5 na representação Estágio, é necessário que todas as áreas de processo descritas na representação Estágio satisfaçam somente as práticas genéricas dos níveis 1, 2 e 3.

Ainda segundo Côrtes (2001, p.135), “pode-se comentar, também, que essa visão talvez não seja compatível com as visões apresentadas na literatura para se mapear um perfil de maturidade de processos construídos a partir da norma ISO/IEC 15504 a um nível de maturidade de uma organização, do ponto de vista do SW-CMM”.

É possível que o SEI promova novas revisões na representação Contínuo, dado o caráter de novidade que introduziu na abordagem tradicional do SW-CMM. A escolha da representação Estágio ou Contínuo depende das características de cada organização. A dupla representação apresenta vantagens, pois possibilita uma adequação na implementação do SW-CMM. Organizações que possuem histórico de utilização do SW-CMM,

provavelmente utilizarão o modelo CMMI na representação Estágio, pela facilidade de migração e simplicidade do uso, visto estarem adaptadas ao processo de capacitação do processo. Por outro lado, organizações que tiveram dificuldades na adoção de algum modelo de qualidade de software, terão na representação Contínuo uma facilidade, pois nesta representação o foco inicial é trabalhar em áreas de processos isoladas, não estando alocadas a nenhum nível de maturidade em particular, como ocorre na representação Estágio.

Vale ressaltar que na representação Contínuo algumas organizações não buscam "estar no nível de maturidade X" e sim, apenas melhorar "o processo Y" da organização, seguindo sempre suas prioridades, por isso, a representação Contínuo é a mais adequada nestes casos.

2.1.4 Considerações finais sobre o CMMI

Nesta seção foi realizada uma exposição do CMMI, que segundo Côrtes (2001, p.96) é o modelo de qualidade de software mais utilizado. Mostra também que este modelo mesmo sendo considerado "burocrático" em algumas críticas, pode ser uma alternativa como modelo de qualidade no processo de desenvolvimento de software.

Apesar de Orr (2002, p.7) demonstrar alguns pontos negativos do modelo CMMI como "arbitrário" e "rigoroso", ter o foco somente no gerenciamento do processo, ter muito investimento em marketing e propaganda e ser voltado somente para grandes organizações, o modelo CMMI agrega, várias práticas da Engenharia de Software para a melhoria de processo de software.

O modelo CMMI é, uma iniciativa na busca de padrão de qualidade, o que é bastante comum hoje em dia em outras áreas, como engenharia da produção de uma fábrica qualquer que não seja de software. Nesse sentido, ações da ISO contribuíram de forma significativa para a melhoria do processo em desenvolvimento de software. Pressman *apud* Paulk (2002, p.488) analisou o CMM, e não CMMI, com o padrão ISO 9000 para o gerenciamento de qualidade, examinando as áreas chave do processo de cada nível do

CMM, e as comparou com os requisitos da ISO 9000 para os processos de gerenciamento de qualidade e concluiu que “para a grande maioria das áreas, há uma clara correlação entre os processos chave e o padrão ISO 9000”.

Com a agregação dos vários modelos CMM e a ISO/IEC 15504, acredita-se que o CMMI e seus produtos derivados, sejam um dos principais modelos para melhoria e avaliação do processo de desenvolvimento de software na atualidade. Isto, de certa forma, justifica a opção por este modelo para o desenvolvimento do presente trabalho.

Para finalizar, estima-se que até dezembro de 2005, o modelo CMMI entre no cenário mundial substituindo de forma gradativa o “antigo” SW-CMM.

2.2 Métodos ágeis

Esta seção apresenta inicialmente os métodos ágeis, sua origem e principais características. Em seguida, aborda detalhadamente algum destes métodos como a XP, FDD, DSDM e o Scrum. Finalmente, realiza uma comparação destes métodos.

A entrega de software dentro do prazo estabelecido, no orçamento previsto e com todos requisitos atendidos, é um dos grandes desafios da atual engenharia de software. Durante anos a “indústria do software” foi baseada no “*code and fix*”, expressão utilizada para codifica e conserta. Isto até poderia atender a pequenas organizações que desenvolvem software mais simples e com um pequeno número de desenvolvedores. Mas em grandes produtoras de software, existe a necessidade da atividade de software ser mais disciplinada, eficiente e previsível. Para isso, surgiram métodos de desenvolvimento de software. Ambler (2004, p.19) afirma que estes métodos são acusados de burocráticos, pois necessitam de considerável quantidade de documentação, recebendo um rótulo de “pesado”.

Os “pesados” são considerados eficientes para grandes projetos com grandes equipes, tendo a disposição todos recursos disponíveis, inclusive os humanos. Projetos menores em pequenas organizações não necessitam deste rigor e formalismo, podendo inclusive ter

projetos mais simplificados, pois a adaptação e os custos destes métodos podem ser um entrave para os pequenos desenvolvedores.

Em 2001, um grupo de pesquisadores auto denominados de Aliança Ágil *(agile alliance)*, entre eles Martin Fowler, Alistair Cockburn, Jim Highsmith, Kent Beck, Mike Beedle e outros, motivados pela suas experiências em desenvolvimento de software, iniciaram uma discussão sobre como desenvolver software de forma mais rápida e eficaz, orientado principalmente à simplicidade. Esta discussão resultou no chamado Manifesto Ágil[11], que em síntese aponta:

> desejamos descobrir melhores caminhos para desenvolver software fazendo e ajudando outros a fazerem. Valorizamos os indivíduos e interações através de processos e ferramentas; O desenvolvimento de software deve possuir uma documentação compreensiva; A colaboração do cliente e respostas às mudanças através de um plano específico.

Assim, como em todos os modelos de processo de desenvolvimento de software, os métodos ágeis também possuem alguns princípios, dentre os quais destacam-se:

1. A prioridade é satisfazer o cliente através da entrega contínua e rápida de uma versão do software com valor agregado.
2. Alterações sobre os requisitos são bem vindas, mesmo que ela ocorra tarde, pois isto dá uma vantagem competitiva ao cliente.
3. Entrega de software freqüente, a cada semana ou a cada mês, sempre visando o menor tempo possível.
4. Os especialistas no negócio e os desenvolvedores trabalham juntos diariamente no projeto.
5. Manter uma equipe motivada fornecendo ambiente e confiança necessários para o trabalho a ser executado.
6. A maneira mais eficiente de comunicação da equipe é através de uma conversa face-a-face.
7. A simplicidade é essencial.

11 Disponível em http://www.agilemanifesto.org/

8. Software funcionando é a medida primária de progresso.
9. A atenção contínua à excelência técnica e ao bom projeto melhoram a agilidade.
10. As melhores arquiteturas, requisitos, e projetos emergem de equipes auto-organizadas.
11. A equipe reflete, a intervalos regulares, sobre como se tornar mais efetivo, então se ajusta e otimiza o seu comportamento.
12. Promover desenvolvimento sustentável. Todos *stakeholders* devem ser capazes de manter um ritmo constante indefinidamente.

Os princípios apresentados pelo manifesto ágil e discutidos por Fowler (2003, p.1), mostram que certos valores continuam sendo importantes, como a modelagem de dados. A documentação continua fundamental, mas com a preocupação da não geração enorme de papéis que provavelmente nunca serão lidos. O planejamento deve ser utilizado, mas com a consciência de que existem limitações sobre qualquer plano.

Neste sentido, surgem os métodos ágeis que podem ser considerados como um meio termo entre a ausência de processo e o processo exagerado, diferenciando-se dos métodos tradicionais basicamente em 2 (dois) aspectos:

a) são adaptativos ao invés de preditivos;
b) são orientados às pessoas ao invés dos processos.

Outra discussão é com relação à denominação, *método* ou *metodologia* ágil. Cockburn (2001, p.10) define metodologia como “uma série de métodos ou técnicas relacionadas”, e método seria: “um procedimento sistemático, similar a uma técnica”. Ou seja, quando se discute um procedimento ou uma técnica utiliza-se o termo método, enquanto o termo metodologia pode ser empregado de forma mais ampla, como por exemplo, gerenciar uma equipe. Para finalizar, Boehm (2004, p. 16) utiliza o termo "métodos ágeis" e não "metodologias ágeis".

Segundo Beck (2000), métodos ágeis são “métodos leves para pequenas e médias equipes, desenvolvendo software com requisitos vagos ou com mudanças rápidas nos requisitos”. E, ainda conforme Beck (2000), “apresentam uma alternativa para documentação no desenvolvimento de software” surgindo, como uma reação aos modelos tradicionais de desenvolvimento de software, como o RUP[12] (*Rational Unified Process*).

Os métodos ágeis fornecem uma mudança nas formas de desenvolver software, dentre as quais destacam-se:

a) priorizar a satisfação do cliente, através da entrega rápida e contínua de software;
b) alterar os requisitos de forma rápida;
c) buscar um desenvolvimento sustentável;
d) orientar a simplicidade.

Astels *et al.* (2002) apontam que “consistem em uma nova forma de construir software, mudando conceitos de métodos tradicionais que inquestionavelmente funcionaram no passado”. Por consistir de uma mudança considerável, os autores, comentam que métodos chamados popularmente de “pesados” atuam como uma restrição a novos negócios, inseridos em um mundo em constante mudança, reforçando o conceito de agilidade para permanência no mercado tão disputado. Cabe destacar, que os métodos ágeis não são contra os modelos de processo de desenvolvimento de existentes, sendo apenas uma proposta diferente que busca também a melhoria do processo, tornando-os mais ágeis e menos burocráticos.

Os métodos ágeis podem ser considerados novos no contexto da engenharia de software, pois datam de 2001. Porém, várias iniciativas estão sendo desenvolvidas numa tentativa de consolidar estes métodos na área da engenharia de software. Um exemplo é a recente inclusão da IEEE (*Institute of Electrical and Electronics Engineers)* em seu portal na internet, um diretório dedicado exclusivamente para discussões e troca de informações

[12] Mais informações em http://www.rational.com

sobre os referidos métodos. Outra iniciativa é que algumas universidades, como por exemplo, a *Calgary University* do Canadá e a universidade de Kaiserslautern na Alemanha, organizam congressos e seminários sobre métodos ágeis tendo como principal objetivo divulgar e promover a difusão destes métodos no cenário mundial. Estas ações podem, de certa forma, contribuir para um melhor reconhecimento destes métodos.

Existem vários métodos que podem ser classificados como ágeis, pois possuem em comum o fato de trabalharem em pequenos projetos onde os requisitos são instáveis, a priorização é com as pessoas e não com processos, possui retroalimentação constante, intimidade com o cliente, guiado por característica e fundamentalmente nos quais as pessoas trabalham juntas com boa vontade. Mesmo com semelhanças, existem diferenças na maneira como tratam, por exemplo, o ciclo de vida do desenvolvimento de software.

Highsmith (2002, xvii) apresenta vários métodos ágeis, entre os quais destacam-se: *Adaptive Software Development* (ASD), *Dynamic Systems Development Method (DSDM), Crystal Methods (CM), Extreme Programming (XP), Feature Driven Development (FDD), Lean Development (LD),* e Scrum.

Dentre todos métodos ágeis destacados anteriormente, o *Lean Development* não será detalhado neste trabalho pela limitada bibliografia sobre este tema. Os métodos ágeis ASD e CM serão brevemente explicados devido a escassez de pesquisas científicas sobre o assunto e principalmente também pela falta de literatura. Por outro lado, os métodos ágeis Scrum, DSDM, FDD e XP serão detalhados neste trabalho porque contrariamente ao que ocorre com o LD, ASD e CM, estes métodos possuem uma documentação suficiente que permita um estudo mais aprofundado.

O método ágil ASD (*Adaptive Software Development)* tem como principais características, segundo Highsmith (2002), o desenvolvimento baseado em um processo iterativo e incremental, voltado para sistemas considerados teoricamente de grande porte e muitas vezes complexos. Vale ressaltar, que este método é um dos poucos indicados para sistemas considerados de “grande porte”. A presença constante do cliente e a utilização da

técnica conhecida como JAD *(Joint Application Development*), são outras características deste método.

Ainda segundo Highsmith (2002), o ASD possui ciclos que duram de 4 a 8 semanas, divididos em 3 fases: *Especular, Colaborar* e *Aprender*. A fase conhecida como *Especular* tem como função principal fixar os prazos, determinar os objetivos e definir um plano para a execução dos trabalhos. A fase denominada de *Colaborar* tem como objetivo básico a construção ou desenvolvimento do projeto. Finalmente, a fase chamada de *Aprender* objetiva conhecer e melhorar o que foi desenvolvido durante a fase *Colaborar*.

O método ágil CM (*Crystal Methods)*, desenvolvido pela IBM através de Alistar Cockburn, não é considerado apenas um método ágil, mas sim, um conjunto de métodos. Isto porque seu desenvolvedor acredita que diferentes tipos de projeto requerem diferentes métodos. Ou seja, não é um *kit* pronto com métodos, mas sim um conjunto de exemplos ajustáveis de acordo com as circunstâncias. Estes métodos são interligados por princípios, nos quais são centrados na pessoa e na comunicação. A filosofia "da família Crystal", segundo o autor, é baseada em quatro pontos principais:

a) *Human-powered* (poder humano): o foco é alcançar o sucesso do projeto através da melhoria do trabalho das pessoas envolvidas, isto é, centralizada na pessoa e não no processo, ou arquitetura ou ferramenta.

b) *Ultralight* (super leve): não importa o tamanho do projeto ou as prioridades, a metodologia trabalhará para reduzir "os papéis" e a burocracia.

c) *Shrink-to-fit*: "encolher para manter saudável, caber": começa com alguma possibilidade pequena (suficiente) e trabalha para tornar ela menor e mais adequada.

d) *Non-jealous* (não ciumento, invejoso): os métodos *Crystal* permitem substituição de elementos por similares de outras tecnologias.

A seguir, faz-se então um detalhamento dos métodos ágeis Scrum, FDD, DSDM e XP.

2.2.1 Scrum

O método ágil Scrum tem como objetivo, segundo Schwaber (2002, p.2), definir um processo para projeto e desenvolvimento de software orientado a objeto, focado nas pessoas, indicado para ambientes em que os requisitos mudam e surgem rapidamente. Foi desenvolvido, inicialmente, por duas organizações produtoras de software, a *Advanced Development Methods*[13] e a VMARK Software[14] , nos anos 90, por Ken Schawber e Jeff Sutherland, para gerenciar projetos de software conforme relata (Highsmith, 2002, p.20). A origem do termo Scrum é uma metáfora a uma tática utilizada no Rugby[15] para recuperar uma bola perdida.

O principal objetivo do Scrum é

> entregar um *software* com a maior qualidade possível dentro de séries, compostas por pequenos intervalos de tempo definidos chamados ***Sprints***, que tem aproximadamente um mês de duração. (Beedle, 2001).

O Scrum é um método para gerenciar o processo de desenvolvimento de software não definindo técnicas nem ferramentas, apenas define como equipes devem trabalhar em ambientes onde requisitos mudam constantemente, sofrendo alterações constantes, oposto ao que ocorre normalmente em uma indústria de manufatura qualquer, onde os processos são repetíveis e muitas vezes bem definidos.

O método baseia-se de acordo com Linda (2000, p.8) e Schwaber (2002, p.7) em princípios como: equipes pequenas de, no máximo, 7 pessoas; requisitos que são pouco estáveis ou desconhecidos e iterações curtas, não definindo técnicas específicas de desenvolvimento de software para fase do desenvolvimento. Divide o desenvolvimento em intervalos de tempos de, no máximo 30 dias, também chamados de *Sprints*.

---

13 Organização cujo proprietário é um dos idealizadores do SCRUM, Ken Schwaber.
14 A VMARK software foi recentemente adquirida pela Informix.
15 Rugby é um esporte tendo como objetivo que duas equipes de quinze, dez ou sete jogadores cada uma, jogando de forma leal e de acordo com as Leis e o espírito desportivo, portando, passando, chutando ou apoiando a bola, marquem a maior quantidade de pontos possível. A equipe que marcar mais pontos será a vencedora da partida. E, SCRUM é o nome dado a disputa da bola. Disponível em :< http://www.rugbynews.com.br>.

O Scrum concentra-se em “como” as equipes devem realizar suas tarefas de forma ordenada para produzir um sistema flexível num ambiente com constantes mudanças.

Nesse sentido,

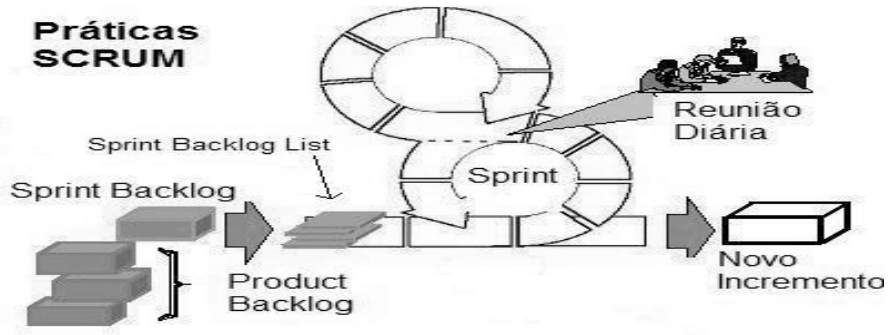
> O método Scrum é uma maneira de fazer coisas que é completamente diferente daquilo que a maioria das pessoas está acostumada a fazer no desenvolvimento de software e produto. (Schwaber, 2002, p.25).

2.2.1.1 Funcionamento Básico

Abrahamsson (2002, p.30) afirma que no Scrum a etapa de desenvolvimento inclui as fases tradicionais do desenvolvimento de software: requisitos, análise, projeto e entrega. Ressalta que, ao final do projeto, uma arquitetura conceitual da solução deve estar preparada para mudanças durante a etapa do desenvolvimento.

Para que isto ocorra, é elaborada uma lista das funcionalidades que deverá conter o produto final chamada de *Product Backlog*. O projeto acabará quando todas as funcionalidades descritas nesta lista forem realizadas.

O sucesso do projeto depende diretamente do *Scrum Master*, pessoa responsável por todo projeto, desde o levantamento dos requisitos até a entrega do produto. O desenvolvimento ocorre num período máximo de 30 dias (também chamado de *Sprint*). As funcionalidades são desmembradas em pequenas tarefas que não poderão exceder a uma semana para execução. Ressalta-se que durante o *Sprint* não são permitidas mudanças nas tarefas que estão em desenvolvimento. A Figura 9 ilustra o processo Scrum, desde a elaboração do *Product Backlog* até a entrega do incremento, conforme descrito a seguir.



Fonte: Adaptado de SCRUM (2002).

Figura 9 Estrutura do processo do SCRUM.

2.2.1.2 Estrutura do funcionamento do Scrum

A estrutura de funcionamento do Scrum é separada em três fases: *PreGame*, *Game* e *PostGame.*

A fase *PreGame* é a fase inicial do Scrum e é dividida em dois estágios: planejamento e arquitetura. O estágio do planejamento consiste no desenvolvimento dos requisitos apontados no *Product Backlog* para o sistema, na definição da entrega das versões e das datas para realização das tarefas. No projeto da arquitetura os itens do *Product Backlog* são revistos e são executados refinamentos no sistema da arquitetura necessários para os próximos domínios de aplicação.

A fase do desenvolvimento, também conhecida como *Game*, é um ciclo iterativo do desenvolvimento do trabalho, que consiste basicamente em encontros da equipe para revisar os planos das versões. Realiza também o desenvolvimento incremental até que o produto a ser entregue esteja finalizado. Nessa fase, todo e qualquer risco é avaliado periodicamente. Novos itens são incluídos na lista do *Product Backlog* se necessário.

A etapa final, denominada de *PostGame*, é a preparação para entrega do resultado final. Possui atividades como integração, documentação ao usuário e preparação de material para treinamento, etc.

2.2.1.3 Papéis e Responsabilidades no Scrum

Existem diferentes papéis e responsabilidades no Scrum que desempenham tarefas durante o processo na utilização das práticas. Os papéis descritos por Schwaber (2002, p.31) são destacados a seguir:

- ***Scrum Master***

É o responsável pela realização das tarefas (valores e práticas) e pelo sucesso direto do Scrum, conforme aponta Schwaber (2002, p.31). É considerado também, como um interlocutor entre a tecnologia e os negócios, e deve realizar as seguintes atividades:

a) organizar as reuniões diárias;
b) representar o gerente do projeto e o técnico;
c) gerenciar todo processo Scrum na organização;
d) remover os impedimentos do projeto.

O *Scrum Master* trabalha constantemente para reduzir a não conformidade do produto entregue ao cliente, sendo que isto pode ser feito através de rápidas respostas aos prováveis impedimentos que poderão surgir durante a execução do projeto.

- ***Product Owner***

É representada por uma pessoa e não um comitê, que além de defender os interesses do negócio tem como objetivo manter e priorizar o desenvolvimento os itens do *Product Backlog*. Desempenha o papel formal para assumir as responsabilidades do projeto, sendo responsável pela liberação e escolha dos itens que passarão à outra fase (conhecida como *Sprint Backlog)* para o desenvolvimento do produto de software.

- **Equipe Scrum**

A Equipe Scrum, através de encontros com o *Scrum Master*, é responsável pelas ações para atingir os objetivos de cada *Sprint*. Determina por exemplo, a criação do *Sprint Backlog* e revisa os itens da lista do *Product Backlog* verificando os impedimentos que precisam ser retirados para o sucesso do projeto. A equipe deve ter, no máximo, 7 pessoas, sendo conhecida também apenas pela expressão “time”.

- **Cliente**

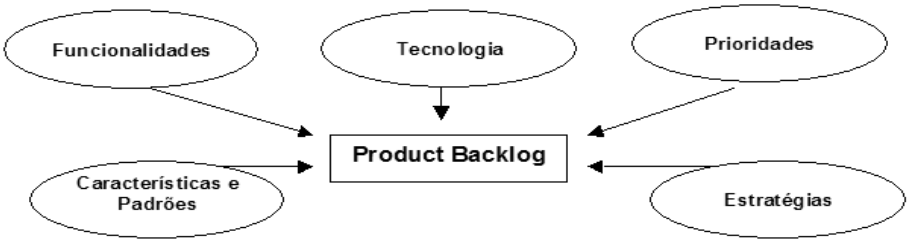
Como na maioria dos métodos ágeis, o cliente está presente diretamente em todo projeto, participando ativamente principalmente na fase inicial de elaboração dos itens que irão compor o produto a ser desenvolvido.

2.2.1.4 As Práticas do Scrum

O método Scrum não requer ou fornece qualquer método específico para desenvolvimento de software, apenas estabelece conjuntos de regras e práticas gerenciais que devem ser adotadas para o sucesso de um projeto. A seguir, um detalhamento das seguintes práticas adotadas pelo método Scrum: *Product Backlog, Daily Scrum, Sprint, Sprint Planning Meeting, Sprint Backlog* e *Sprint Review Meeting*.

- ***Product Backlog***

É o ponto inicial do Scrum, sendo considerada a fase responsável pela coleta dos requisitos, conforme aponta Schwaber (2002, p.33). Nesta fase, através de reuniões com todos *stakeholders* no projeto, são apontados os itens com todas as necessidades do negócio e os requisitos técnicos a serem desenvolvidos, como as funcionalidades, características e padrões, prioridades, tecnologia e estratégias (tecnologia da informação e requisitos), ou seja, o *Product Backlog*, é uma lista de atividades que provavelmente serão desenvolvidas durante o projeto. A Figura 10, ilustra as diversas origens para composição dos itens do *Product Backlog*.



Fonte: Adaptado de Schwaber (2002, p.33)

Figura 10 Origens para confecção do *Product Backlog*

Tem-se como regra, que nenhum item é adicionado fora da reunião ao *Product Backlog*, garantindo com isso que novos itens não sejam agregados ao projeto sem o comum acordo de todos *stakeholders* no projeto. Porém, itens internos resultantes do *Product Backlog* original podem ser incluídos livremente para desenvolvimento.

Um exemplo do *Product Backlog* é apresentado na Figura 11, onde os itens são descritos e separados por categorias que indicam prioridades (muito alta, alta e média), o responsável e o tempo estimado, em horas, para execução das tarefas.

Prioridade	**Item**	**Descrição**	**Tempo Estimado**	**Responsável**
Muito Alta				
	1	Finalizar versão do Banco de Dados	16	Paulo
	2	Legalizar licenças de uso dos softwares	24	Pedro
Alta				
	3	Desenvolver programa de usuários	5	José
	4	Software para consulta de clientes	400	José e Paulo
Media				
	5	Controle dos padrões	4	Maria

Fonte: Adaptado de http://www.controlchaos.com

Figura 11 Exemplo de itens do *Product Backlog*

Outras descrições dos itens do *Product Backlog* são:

a) melhorar a portabilidade do produto;
b) simplificar o processo de instalação quando existem vários Bancos de Dados em uso;

c) determinar como *workflows*[16] podem ser adicionados ao produto.

- ***Daily Scrum***

De acordo com Linda (2003, p.1), as reuniões do Scrum são realizadas, na maioria das vezes, diariamente ou em dias alternados, com duração de aproximadamente quinze minutos, e tempo limite de no máximo trinta minutos. Este tempo alocado para as reuniões possibilita a identificação dos obstáculos ou impedimentos ao projeto. Exemplos de impedimentos são apresentados por Schwaber (2002, p.44) como: "A Rede de computadores ou os servidores estão lentos", ou "Não sei como proceder em determinada situação". Ressalta-se que essas reuniões não objetivam a resolução dos problemas, pois os mesmos são tratados a posterior com somente efetiva participação dos *stakeholders* no problema.

Rotineiramente, o *Scrum Master* durante a reunião, levanta três questões para cada membro da equipe assim destacada por Schwaber (2002, p.43):

a) O que foi finalizado desde a última reunião do grupo? O *Scrum Master* registra quais tarefas foram completadas e quais ainda estão pendentes.
b) Quais foram as dificuldades encontradas durante o trabalho? O *Scrum Master* registra todas as dificuldades encontradas para posteriormente encontrar uma maneira de resolvê-las.
c) Quais são as atividades específicas que a pessoa planeja finalizar para o próximo encontro? O *Scrum Master* ajuda os integrantes da equipe a escolher as tarefas mais importantes. Devido ao curto espaço de tempo, em média 24 horas, as tarefas são geralmente pequenas.

A reunião realizada pelo grupo tem vários objetivos sendo que a maior parte dos esforços está concentrada no *Product Backlog*. Cada item da lista que foi trabalhado, automaticamente é retirado da lista, ou seja, quanto menos itens na lista, melhor para o

[16] KOBIELUS, James G. *Workflow strategies*. Foster city, IDG Books, 1997, diz que *"Workflow é o fluxo de controle e informação num processo de negócio"*.

Scrum Master, por isso, existe uma grande preocupação do *Scrum Master* em concentrar o trabalho nas tarefas mais importantes, que reduzirão o *Product Backlog*, proporcionando um maior progresso para a equipe. A reunião também possibilita que todas as pessoas fiquem informadas sobre o progresso e as dificuldades encontradas.

- ***Sprint***

É considerada a principal fase do Scrum, onde são executados os itens de trabalho definidos no *Product Backlog* pela equipe Scrum, sendo essa execução não superior a 30 dias (pode durar de uma a quatro semanas). Conforme Abrahamsson (2002, p.30), o *Sprint* inclui as fases tradicionais do desenvolvimento de software: requisitos, análise, projeto e entrega. O resultado do item do trabalho a ser desenvolvido é chamado de *Product Increment*. Aqui não existe o "como fazer", e sim o "vamos fazer".

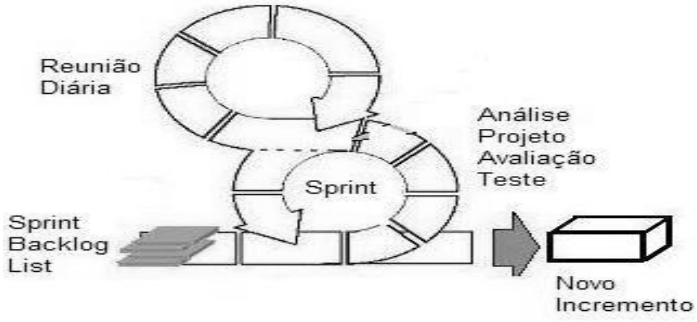
Observa-se que cada estágio presente no ciclo de desenvolvimento é mapeado para um *Sprint* ou uma série de *Sprints*, variando de acordo com a duração e dificuldade de cada fase. O processo de levantamento de requisitos pode durar um *Sprint*, enquanto o processo de implementação do sistema possa consumir mais de um *Sprint*, por exemplo.

Diferentemente de uma abordagem repetível e definida, no Scrum não existem processos pré-definidos dentro de um *Sprint*. Ao invés disso, existem reuniões periódicas, denominadas *Daily Scrum*, que coordenam a realização das atividades existentes.

Ao final de cada *Sprint*, segundo Beedle (2001), é criado uma pequena demonstração para ilustrar ao cliente o que está acontecendo, proporcionando ao desenvolvedor uma sensação que cumpriu a tarefa com êxito. Outra finalidade é integrar e testar uma parte significante do *software* que está sendo desenvolvido garantindo um progresso real das atividades, significando redução dos itens do *Product Backlog*.

Assim que a etapa de coleta e re-organização das tarefas incompletas e das novas tarefas é finalizada, um novo *Product Backlog* é criado sucedido da inicialização de um novo *Sprint*. A qualidade poderá ser garantida com aplicação de práticas de engenharia de

software como a XP (Programação Extrema). A Figura 12, ilustra somente a fase do desenvolvimento, onde os requisitos são apresentados no *Sprint Backlog List*, e desenvolvidos no *Sprint* gerando um novo incremento que será entregue para a fase final do processo.



Fonte: Adaptado de Schwaber (2002, p.8).

Figura 12 O *Sprint* no Scrum.

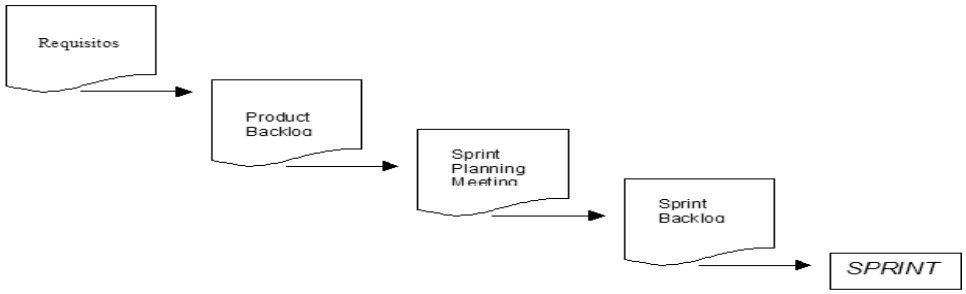
- ***Sprint Planning Meeting***

Cada *Sprint* inicia com uma sessão chamada de *Sprint Planning Meeting*, tendo como objetivo analisar os itens do *Product Backlog* juntamente com o *Product Owner* a fim de priorizar os itens a serem desenvolvidos. Após a definição, os itens escolhidos são repassados à equipe Scrum para desenvolvimento, essa etapa é chamada de *Sprint Goal*, que é o objetivo do primeiro *Sprint*. Durante a *Daily Scrum* do primeiro *Sprint*, os impedimentos ou problemas do projeto são identificados e removidos para a continuidade do mesmo. Ao final, o Cliente e o *Scrum Master* decidem na *Sprint Review* a próxima etapa a ser desenvolvida.

- ***Sprint Backlog***

Sprint Backlog é o ponto inicial de cada *Sprint*, sendo considerado como uma coleção de características e funções que foram extraídas do *Product Backlog* e serão desenvolvidas conforme definidas no *Sprint Goal*, selecionado pela equipe Scrum, *Scrum Master* e *Product Owner*, para o desenvolvimento no *Sprint*.

A Figura 13 ilustra o processo desde o levantamento dos requisitos até o *Sprint*.



Fonte: primária

Figura 13 Processos do Scrum até o Sprint.

Um gráfico poderá ser utilizado para gerenciar todas as atividades e todos os requisitos do software que devem ser desenvolvidos durante o ciclo de desenvolvimento. Este gráfico de acompanhamento é criado somando-se diariamente a quantidade de horas trabalhadas restantes para que o *Sprint* seja finalizado. Ressalta-se que no Scrum não é utilizado nenhum tipo de acompanhamento de quanto tempo foi gasto para obtenção de resultados, e sim, apenas quanto tempo falta para atingir a meta definida.

- ***Sprint Review Meeting***

No último dia do *Sprint*, a equipe *Scrum* e o *Scrum Master* apresentam os resultados do incremento numa reunião informal ao cliente, ao gerente e ao *Product Owner*, que posteriormente, analisarão estes resultados decidindo sobre as novas atividades que poderão integrar o *Product Backlog*. Ressalta-se que o Scrum tem como premissa básica não exigir todos os pré-requisitos logo no início do projeto, pois estes são "descobertos" na medida em que o projeto evolui.

2.2.1.5 O Scrum em projetos maiores

Schwaber (2002, p.128) garante que não existem limitações como tamanho e complexidade de projetos desenvolvidos com o método ágil Scrum. Todavia, o autor adverte que em projetos maiores, problemas como gerenciamento das mudanças dos requisitos, relações entre as equipes de desenvolvimento e trocas de pessoal, devem ser cuidados. Indica que não se deve iniciar o projeto com múltiplas equipes ao mesmo tempo,

e sim, com uma única equipe. Explica ainda, que esta equipe terá como objetivo principal criar uma primeira aplicação a ser entregue ao cliente, para posteriormente preparar alguns componentes que serão utilizados por outras aplicações. A aplicação entregue, deverá estar estruturada em “camadas lógicas” para facilitar o entendimento. Uma vez produzida esta aplicação, novas equipes são agregadas ao projeto compartilhando os recursos existentes. Desta forma, o autor acredita que projetos maiores podem ser desenvolvidos com o método Scrum.

2.2.1.6 Considerações finais sobre o Scrum

Os métodos de desenvolvimento de software utilizados atualmente parecem um pouco restritivos, no sentido de exigirem pré-requisitos bem definidos para um produto que ainda nem foi criado. Por exemplo, é difícil que o responsável perceba todas as necessidades do cliente no período de coleta de requisitos, visto que existem muitas incertezas, além de constantes modificações do projeto.

Busca-se então, um método mais “adaptativo”, que permita constantes modificações no projeto. É justamente aqui que se encaixa o método ágil de desenvolvimento Scrum, com a premissa básica de não exigir todos os pré-requisitos logo no início do projeto. O Scrum define em seu escopo reuniões curtas e freqüentes entre os membros da equipe, a fim de que os problemas sejam rapidamente identificados e resolvidos.

Por se tratar de um método que não necessita de grande investimento financeiro, ser um dos poucos métodos ágeis com capacidade de escalabilidade, como destaca (Schwaber 2002, p.8), além de propor uma nova abordagem para o desenvolvimento de software, este método, pode ser uma alternativa à escolha de um método ágil.

2.2.2 Feature Driven Development (FDD)

Feature Driven Development (FDD), ou em português, Desenvolvimento Dirigido à Característica, como outros métodos ágeis, foca na entrega de pequenas iterações com a presença de alguma característica tangível. Em geral, os desenvolvedores “gostam” deste

método porque recebem constantemente novas atividades, em geral a cada duas semanas, com iterações também curtas conforme apresenta Coad (1999, p.184). O FDD foi desenvolvido por Peter Coad e Jeff De Luca, e uma das principais obras[17] foi publicada recentemente em 2002, após a utilização deste método com sucesso em uma instituição bancária, segundo Highsmith (2002, p.270).

Para Coad (1999, p.185), uma característica "é uma função com valor para o cliente que pode ser desenvolvida em duas ou menos semanas", pois este método é baseado em modelos e é guiado (dirigido) por característica. Toda característica poderia ser descrita com a utilização do seguinte padrão:

<ação> <artigo> <resultado> <preposição> <artigo> <objeto>

A seguir, são apresentadas características de um software que gerencia vendas de produtos para exemplificação:

a) calcular o total de vendas de produtos da organização;

b) calcular o total de compras de um cliente.

Highsmith (2002, p.273) afirma que, por ser iterativo e com ciclo de vida curto, o FDD é indicado para o desenvolvimento de software onde os requisitos mudam rapidamente, como acontece no método ágil SCRUM. Afirma, ainda, que "trabalha melhor, para processos simples e bem definidos".

Coad (1999) indica a utilização do FDD em projetos de tamanho médio, que envolvam entre 10 e 30 pessoas na equipe de desenvolvimento do projeto.

[17] Palmer, S.R, Felsing J.M. Feature-Driven Development, Upper Saddle River, NJ, Prentice Hall, 2002

2.2.2.1 O Processo do FDD

Ainda segundo Coad (1999), o FDD prima pela clara definição das etapas do processo, pois acredita que contribuem sistematicamente para o sucesso do projeto, fazendo que desenvolvedores e *stakeholders* no projeto o sigam, evitando, com isso, que busquem alternativas próprias que dificultam o andamento do projeto. Os produtos de saída de cada etapa devem ser tangíveis, ou seja, devem definir o que será produzido, o formato que terá, e qual resultado deverá produzir.

O FDD possui o foco no projeto e construção e é composto por cinco etapas: três delas (desenvolver um modelo, construir uma lista de características e planejar cada uma delas) são realizadas no início de cada projeto. As duas últimas, (projeto de cada característica e sua construção) são completadas dentro de cada iteração. Para De Luca (2003, p.1), cada uma dessas cinco etapas, possui um conjunto de padrões presentes que devem ser seguidos obrigatoriamente. Esses padrões são chamados de ETVX, acrônimo de, Entrada (*Entry*), Tarefa (*Task*), Verificação (*Verification*) e saída (*eXit*). A seguir, uma pequena descrição de cada padrão, ainda conforme De Luca:

a) Entrada (*Entry*): especifica e define os critérios de entradas para as etapas;

b) Tarefa (*Task*): é composto por uma lista com as tarefas que deverão ser realizadas. Cada tarefa possui: um título; que papel desempenhará; quem é o responsável (equipe ou pessoa), se é opcional ou obrigatória e uma pequena descrição;

c) Verificação (*Verification*): especifica tipos de avaliações (internas e externas) e inspeções de projeto e código;

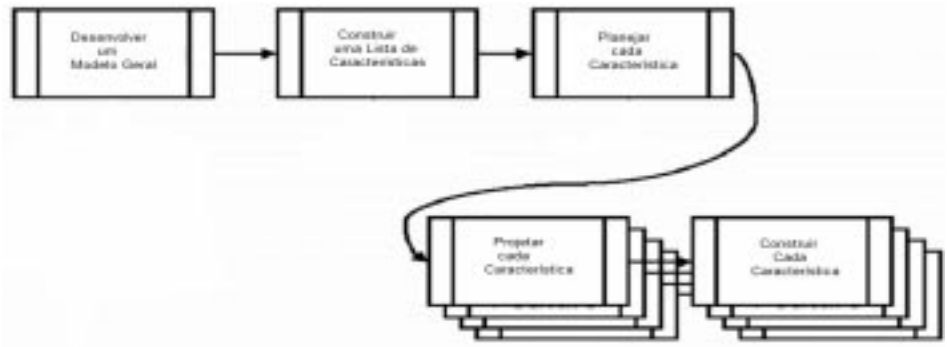
d) Saída (eXit): especifica os critérios de saída, definindo os produtos tangíveis.

Os autores, Highsmith (2002, p.73), De Luca (2003) e Palmer (2004) apresentam as cinco etapas no FDD assim:

a) desenvolver um modelo global;
b) construir uma lista de características;

c) planejar a construção por características;
d) projetar cada característica;
e) construir cada característica.

A Figura 14, demonstra a estrutura de todo o processo no FDD, com a descrição de cada etapa que são executadas seqüencialmente dentro do processo no FDD. Ressalta-se que a iteração e o incremento estão presentes mais fortemente nas duas últimas etapas.



Fonte: Adaptado de De Luca (1999).

Figura 14 Estrutura do processo no FDD.

A seguir, uma descrição de cada etapa do FDD, conforme os autores Coad (1999, p.190), Highsmith (2002, p.270) e Abrahamsson (2002, p.48).

- **Desenvolver um modelo global**

É primeira etapa do processo no FDD, onde são definidos a abrangência (escopo), o contexto e os requisitos do software que será construído conforme demonstra Abrahamsson (2002, p.48). São elaboradas as documentações dos requisitos tais como: casos de uso e especificações funcionais. Cabe ressaltar que no FDD não se trata do Gerenciamento de Requisitos, e não se abordam questões dos requisitos não funcionais. As principais pessoas envolvidas são o Especialista do Domínio e o Projetista. A seguir, as tarefas requeridas nesta etapa:

a) construir e modelar o domínio da aplicação;

b) documentar, se necessário, requisitos funcionais e modelo de dados.

Nesta etapa, têm-se como resultados ou saídas: o diagrama de classe, o diagrama de seqüência, e uma lista informal de características com possíveis alternativas de modelagem.

- **Construir uma lista de características.**

O objetivo da segunda etapa do processo no FDD, conforme apresenta Highsmith (2002, p.274), é construir uma lista completa de todas as características do produto a ser desenvolvido (similar as Estórias dos usuários da XP). Cada característica apresenta um peso, uma prioridade e uma hierarquia, que determinam a ordem/importância no desenvolvimento. Isso é possível devido à documentação existente dos requisitos e o *walkthroughs*[18], pois fornecem uma base para construção dessa lista de características que poderá ser desenvolvida, além da definição das classes e dos métodos.

Na lista, a equipe de desenvolvimento apresenta cada função esperada pelo cliente que deverá estar presente no software a ser desenvolvido. A lista é dividida em conjuntos menores conforme suas características para melhor detalhamento, sendo que a execução de cada divisão (detalhes do conjunto) não poderá exceder a duas semanas.

Por fim, a lista é revisada pelos usuários e mantenedores do software para validação e completude para entrega a próxima etapa.

- **Planejar a construção por características.**

Nesta etapa, é construído um plano onde um conjunto de características é agrupado seqüencialmente de acordo com prioridade e dependência definidas pelo programador chefe. As classes identificadas na primeira etapa são encaminhadas aos desenvolvedores. A equipe de planejamento formada pelo gerente do projeto, programador chefe, e gerente de desenvolvimento é a responsável pela elaboração do plano de quais características serão desenvolvidas.

[18] Esforço conjunto de revisão com finalidade de melhorar a qualidade do produto de software. Disponível em http://www.babylon.com. Acesso em Jan 2004.

Planejar no FDD implica, conforme Highsmith (2002, p. 277), em verificar vários fatores que poderão auxiliar ou prejudicar no desenvolvimento de software como determinação dos riscos do projeto, complexidades, balanceamento dos trabalhos, etc.

São exemplos de saída dessa etapa: a designação dos proprietários de cada classe e um conjunto de características que serão repassados ao programador chefe.

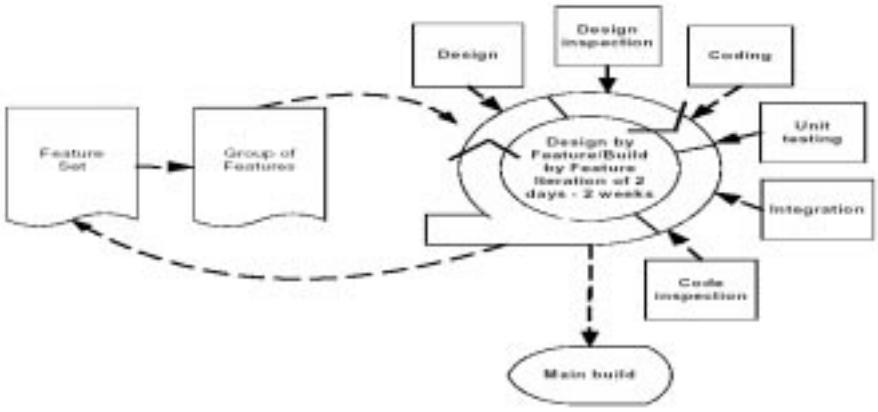
- **Projetar cada característica.**

É nessa etapa que cada classe é identificada. São verificadas características requeridas no planejamento através do estudo da documentação entregue. É elaborado um plano no qual um conjunto de características é agrupado seqüencialmente de acordo com sua prioridade e dependência definidas pelo programador chefe. As classes identificadas na primeira etapa são encaminhadas aos desenvolvedores para construção.

Algumas tarefas requeridas nesse processo, executadas pela equipe e o programador chefe:

a) estudar a documentação existente;
b) desenvolver o diagrama de seqüência;
c) refinar o modelo do objeto;
d) escrever as classes e os métodos;
e) inspecionar o projeto.

São resultados desse processo: o diagrama de seqüência detalhado, o diagrama de classe, classes e métodos atualizados, além das características com toda documentação. A Figura 15 apresenta a interação entre as etapas de projeto e construção de cada característica.



Fonte: Abrahamsson (2002, p.50)

Figura 15 Projeto e Construção de cada característica no FDD.

- **Construir cada característica.**

É a última etapa do processo no FDD, onde um pequeno grupo de características são selecionadas do conjunto definidos no processo de construção da lista para início do processo de desenvolvimento. A seguir, algumas tarefas requeridas nessa etapa que são executadas pela equipe e o programador chefe:

a) implementação de classes e métodos;
b) inspeção de código;
c) teste de unidade;
d) verificação;
e) inspeção de código e teste de unidade.

De Luca (2004, p.11) demonstra que 85% do tempo gasto nos dois últimos processos do FDD referem-se em especial ao projeto e codificação.

São resultados dessa etapa a implementação das classes e métodos, inspeção do código e testes de unidade, e a efetiva entrega da versão ao usuário. Após a conclusão de cada característica nesta etapa retorna a anterior para iniciar novo ciclo.

Ressalta-se que o programador chefe tem papel decisivo nesta etapa do processo mantendo a iteração e a entrega dos produtos finais.

2.2.2.2 Considerações finais sobre o FDD

Por se tratar de um método novo, carecem estudos avançados em universidades e, por conseguinte, publicação de artigos científicos em revistas e ou seminários especializados em engenharia de software que validem sua efetiva aplicação no desenvolvimento de software.

Percebe-se, forte apelo comercial para utilização deste método, pois Peter Coad, o principal autor do FDD, comercializa uma ferramenta desenvolvida por ele baseada no FDD. Para Highsmith (2002, p.284), este método, juntamente com a XP, é considerado "um dos melhores métodos ágeis existentes na atualidade no mercado internacional".

Por fim, o FDD na definição dos processos e suas etapas com tarefas associadas, se aproxima dos conceitos adotados por Deming (1990) nos princípios da Qualidade Total, como por exemplo, o princípio da total satisfação do cliente. Este princípio tem como objetivo não apenas agradar o cliente, mas superar suas expectativas.

2.2.3 Dynamic Systems Development Method (DSDM)

Dynamic Systems Development Method (DSDM) ou, Método de Desenvolvimento de Sistemas Dinâmicos originou-se em 1994 na Inglaterra pela formação de um consórcio de organizações britânicas, tendo como objetivo fomentar a utilização de práticas alternativas no desenvolvimento de software, com atuação em diversos países como: EUA, Suécia, Holanda, França e Índia. Dane Falkner[19] e Arie van Bennekum[20], são alguns dos vários membros desse consórcio que divulgam as práticas do DSDM[21].

19 Dane Falkner é presidente da organização Surgeworks nos EUA, especializada em treinamento e consultoria em DSDM. Foi responsável pelo consórcio DSDM nos EUA em 2001.
20 Arie van Bennekum é sócio proprietário da Solvision. Mais informações em http://www.thevisionweb.nl/
21 Disponível em http://www.dsdm.org.

A idéia principal do DSDM é formalizar as práticas adotadas pelo RAD *(Rapid Application Development)* ou, Desenvolvimento Rápido de Aplicações. O termo RAD, se aplica a projetos que têm prazos curtos, e que em geral envolvem o uso de prototipagem com auxílio de ferramentas CASE e linguagens de quarta geração. O RAD, conforme destaca Coleman

> contribui no desenvolvimento de aplicações com esta característica, mas sofre de problemas semelhantes a outras formas de desenvolvimento rápido: a falta de prazo pode implicar em qualidade reduzida, e há necessidade de habilidade maior dos desenvolvedores, e suporte maior da gerência e dos clientes (2004).

O DSDM é um método que aplica prototipagem a fim de garantir a entrega contínua de um produto tangível. A entrega dos produtos intermediários é determinada por faixas fixas de tempo. Highsmith (2002, p.254), afirma que "nada é construído perfeitamente na primeira vez". Apresenta ainda, que os princípios que norteiam o DSDM são muito próximos com os apresentados no manifesto ágil, por isso, o DSDM pode ser considerado como um método ágil.

Considerando o domínio de aplicação, o DSDM é mais bem indicado em organizações do tipo comerciais/industriais do que em aplicações científicas. É utilizado para qualquer tamanho de projeto, onde a única condição para grandes projetos, é que seja dividida em pequenas partes para melhor execução das atividades, chamada de "*timeboxes"*.

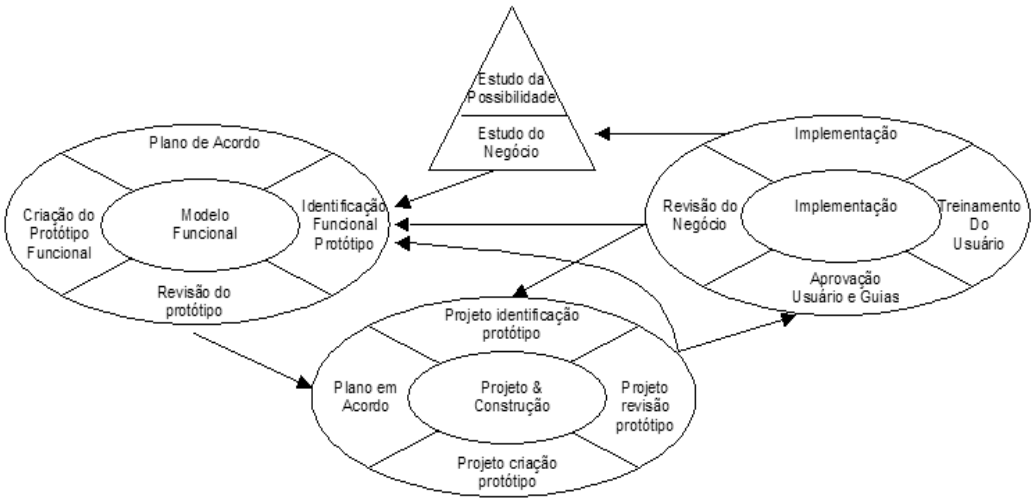
2.2.3.3 O Processo no DSDM

Abrahamsson (2002, p. 62), apresenta as 5 fases do DSDM, a seguir destacadas:

1) estudo de viabilidade;
2) estudo do negócio;
3) iteração para o modelo funcional;
4) iteração para projeto e desenvolvimento;
5) implementação.

As duas primeiras fases, Estudo da Viabilidade e Estudo do Negócio são seqüenciais. As três últimas ocorrem enquanto houver o desenvolvimento do projeto, sendo consideradas iterativas e incrementais.

As fases ocorrem em ciclos. No primeiro ciclo ocorre o estudo da viabilidade e do negócio, e depois em paralelo, o ciclo do modelo funcional (análise e protótipos), ciclo de projeto e construção (engenharia do produto) e, por fim, o ciclo da implementação (implantação operacional). A Figura 16, apresenta o processo no DSDM e as iterações que ocorrem durante todo o desenvolvimento do projeto.



Fonte: Adaptado de Highsmith (2002, p.255)

Figura 16 O processo no DSDM.

A seguir, uma descrição das fases do processo no DSDM.

- **Estudo de Viabilidade**

Esta fase determina se o projeto é factível ou não, e se o DSDM é o método adequado. A entrada para o estudo da viabilidade é uma descrição geral do sistema e de como ele será utilizado dentro de uma organização. Opcionalmente, um rápido protótipo poderá ser construído caso não se tenha amplo conhecimento do negócio ou da tecnologia envolvida no projeto. Esta fase não poderá ultrapassar duas semanas.

São apresentadas duas saídas nesta primeira fase: um relatório da viabilidade que recomenda se vale a pena ou não realizar o processo de engenharia de requisitos e o processo de desenvolvimento de software, e um plano para o desenvolvimento do projeto.

- **Estudo do Negócio**

É nesta fase onde a característica essencial do negócio e as tecnologias são analisadas. Esta análise é elaborada a partir de reuniões, encontros informais e *workshops*, com os participantes do projeto, a fim de discutirem todos aspectos relativos ao projeto, para que, ao final, entrem num comum acordo sob o que será desenvolvido. São exemplos de saídas desta fase a definição da arquitetura do sistema e um plano mais detalhado do protótipo a ser desenvolvido.

- **Iteração para o Modelo Funcional**

É a primeira fase do DSDM considerada iterativa e incremental. Um modelo funcional é produzido como saída, contendo a codificação do protótipo e o modelo da análise. O teste é também considerado como essencial nesta fase. O protótipo vai sendo gradativamente substituído pelo produto final, conforme o conceito apresentado por Sommerville (2003, p.150), conhecido por “Prototipação Descartável”. Nesta fase, ocorre o desenvolvimento em si, sendo que na primeira aprimora-se o levantamento de requisitos, funcionais e não funcionais, e na segunda, assegura-se a qualidade dos protótipos gerados. Conforme demonstra DSDM (2004), os são produtos finais desta fase: protótipo funcional; lista dos requisitos não-funcionais; plano de implementação; plano *timebox*; registros da revisão do modelo funcional e a análise de riscos.

- **Iteração para Projeto e Desenvolvimento**

Nesta fase, são adequados os protótipos em conformidade com o levantamento dos requisitos não funcionais realizados na fase anterior, para atender as necessidades dos usuários. Os produtos finais desta fase, ainda conforme DSDM (2004) são: plano *timebox*; projeto do protótipo; revisão dos registros do projeto do protótipo e testes.

- **Implementação**

A fase final da Implementação é onde ocorre a transferência do ambiente de desenvolvimento para o de produção Cabe ressaltar que somente após ocorrer o treinamento aos usuários, o sistema entrará em utilização. É nesta fase, que o processo no DSDM define se os requisitos são preenchidos, caso isso não ocorra, uma nova iteração com as etapas do Modelo Funcional e Projeto e Construção é realizada para a devida correção, sendo que isto poderá ocorrer diversas vezes até que a não conformidade seja resolvida. São resultados dessa fase, além da entrega do produto final, o manual do usuário e um relatório com a revisão do projeto.

Vale ressaltar que o DSDM foca o trabalho conjunto de forma colaborativa com constante comunicação entre a equipe, como ocorre em todo método ágil.

2.2.3.4 Considerações finais sobre o DSDM

Conclui-se, que o DSDM inclui vários processos gerenciais em sua estrutura, reconhecendo explicitamente a necessidade de tais processos para garantir o desenvolvimento ordenado de software. Embora existam técnicas para a garantia de qualidade, o DSDM não inclui indicações para o uso de técnicas durante o processo. Ressalta-se, que o DSDM tem como pré-requisito que a organização possua um mínimo de organização em seu processo de desenvolvimento.

Segundo dados da Cutter Consortium, o DSDM situa-se entre os quatro métodos ágeis mais utilizados e, é um dos métodos ágeis mais bem definidos e organizados, apresentando características similares ao RUP, além de “contar com fácil adequação a outros métodos ágeis como a XP”. Porém, o DSDM pertence a um consórcio, e como tal, para ter acesso à documentação mais detalhada se faz necessário o cadastramento neste consórcio com pagamento de taxas mensais. Este é um dos grandes problemas do DSDM, pois dificulta o acesso e sua disseminação na comunidade de desenvolvimento de software.

2.2.4 Extreme Programming (XP)

> XP veio para ficar. Ela é real e certamente não é uma novidade passageira. A XP se concentra basicamente na criação de software com alta qualidade e abandona todo tipo de *overhead* de processo que não suporte diretamente esse objetivo. Astels *et al.* apud Ambler (2002, p. XIII).

Desta forma, Ambler comenta que a XP "veio para ficar", e Beck (2000), um dos principais autores e fundadores da XP, a define como "um método ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante mudança". Ainda segundo Beck, a XP "representa uma disciplina e não um método de desenvolvimento de software", pois XP baseia-se em regras que devem ser usadas todo tempo e não em uma seqüência de procedimentos estáticos. As regras são conhecidas como práticas e tornam possível a aplicabilidade da programação extrema no software.

"A XP vem crescendo na utilização pelo mercado de desenvolvimento de softwares frente às metodologias tradicionais", conforme Highsmith (2002, p.297). O menor custo, a rapidez do desenvolvimento e a adequação a requisitos mutáveis contribuem para isso. As práticas da XP estão sendo usadas com sucesso em organizações como Ford, Symantec, BMW, Borland, entre outras, como aponta Astels *et al.* (2002, p. XVII).

Beck (2000), afirma que o estado natural da especificação do sistema é aquele na qual ocorram mudanças, e para que isto se efetive, é necessário que se tenha um custo acessível para alterar o software, mesmo em estágios avançados de desenvolvimento. Ainda segundo Beck, a XP possui as seguintes características:

a) Mudanças de requisitos são aceitas a qualquer momento durante o desenvolvimento de um projeto;
b) Todo código de produção é desenvolvido em duplas que se revezam mais de uma vez por dia;
c) Todo código é testado através de *script*s de teste automatizado;
d) Todo *script* de teste é escrito antes do respectivo código;

e) A medição de progresso de um projeto é feita através de funcionalidades que passam nos testes de aceitação definidos pelo cliente;
f) O código deve estar sempre na sua forma mais manutenível;
g) Não existe um único dono do código, a propriedade do código é coletiva;
h) Clientes podem testar o produto geralmente a partir da terceira semana de desenvolvimento;
i) Nenhuma documentação é necessária durante o desenvolvimento;
j) O *design* é trabalhado depois da codificação.

Segundo Mesquita (2002), a XP é indicada para equipes pequenas pois, este método, supõe que as equipes de desenvolvimento de um projeto possuam de 2 a 10 desenvolvedores. Também, é indicada para organizações que possuam ambientes instáveis, ou seja, quando a mudança de requisitos é uma constante e exigem respostas rápidas para atender aos novos requisitos. A seguir uma descrição dos valores na XP.

2.2.4.1 Valores na XP

A XP enfoca e prioriza a melhoria do projeto de software em quatro valores diferentes: simplicidade, comunicação, coragem e realimentação, a seguir detalhadas:

- **Simplicidade**

O software dever ser simplificado continuamente, buscando ser mais simples e inteligível. Segundo Astels *apud* Beck (2002, p. 9), “O melhor estado do software no momento da mudança é simples". Um código simples deve ser claro, fácil de compreender, com nomes mnemônicos, com problemas complexos quebrados em partes menores e sem duplicação.

- **Comunicação**

A XP prioriza o diálogo a outras formas de comunicação como *chats*, *emails*. Segundo Mesquita (2002), a comunicação é um dos pontos fortes da XP, pois busca

dinamismo em um ambiente que a mudança de requisitos pode ser uma constante. A criação de relatórios que podem se tornar obsoletos em pouco tempo é discutida.

- **Coragem**

Ainda conforme Mesquita (2002), é preciso coragem para apontar um problema no projeto, pedir ajuda quando necessário, simplificar o código e buscar um relacionamento direto com o cliente para, quando necessário, informar que não será possível implementar um requisito no prazo determinado. Ou seja, fazer a coisa certa mesmo que não seja a coisa mais popular naquele momento.

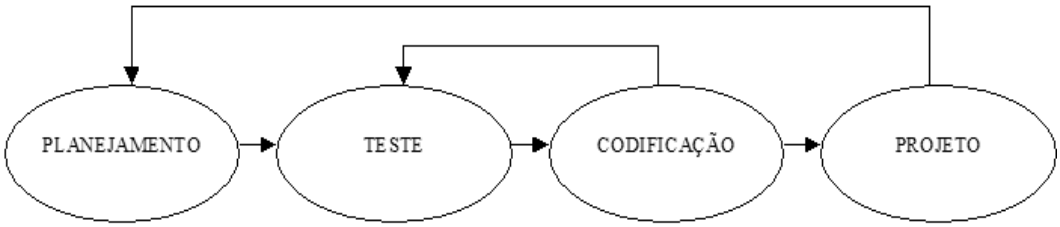
- **Realimentação**

O *feedback* (realimentação) deve ser feito a cada iteração (de cinco até quinze minutos) ou a cada *release*, ocupando pouco tempo no desenvolvimento de requisitos falsos ou em funções que provavelmente não terão utilidade. Quanto mais rápido os problemas forem descobertos, mais rápida será a solução dos mesmos. Da mesma forma, toda a oportunidade descoberta logo será mais rapidamente aproveitada.

2.2.4.2 Práticas da XP

A XP possui regras para as atividades de Planejamento, *Design*, Testes e Codificação. Uma relação entre as atividades da XP é apresentada na Figura 17. Cada uma destas atividades é subdividida em práticas que devem ser seguidas mesmo que algumas destas sejam conhecidas, mas muitas vezes não utilizadas.

As práticas segundo Astels *et al.* (2002, p. 3) "...são criadas para funcionar juntas e fornecer mais valor do que cada uma poderia fornecer individualmente".



Fonte: Adaptada de Beck (2000)

Figura 17 Relação entre as atividades da XP.

A seguir, uma descrição das atividades da XP com o detalhamento das práticas adotadas, conforme descrevem os autores Beck (2000) e Astels *et al.* (2002).

- **Planejamento**

Desenvolver software necessita de planejamento independente do tamanho, conforme afirma Beck (2000). O planejamento serve para fornecer uma compreensão mútua para todas as partes de quanto tempo, por exemplo, levará o projeto. A seguir, uma descrição das práticas adotadas na atividade do Planejamento da XP.

a) Estórias de Usuário: o cliente escreve, em cartões, pequenas estórias sobre as funcionalidades do sistema, com poucas frases em uma “linguagem de fácil entendimento”. Estes cartões são utilizados para documentação que descreve os requisitos e para a realização de estimativas de tempo de desenvolvimento do software

b) Planejamento da Versão: consiste em uma reunião onde são definidos o cronograma, a prioridade, as medidas de desempenho e quantas Estórias de Usuário podem ser feitas em cada iteração. O resultado deste trabalho gerará o “Plano de Iteração”.

c) Pequenas Versões: o projeto é feito em pequenas etapas e em cada etapa um protótipo é entregue ao cliente para que este possa utilizar e ao mesmo tempo, acompanhar o desenvolvimento do software. Para Astels *et al.* (2002), “ao usar as primeiras *releases* do software, o cliente pode adquirir mais conhecimento sobre como tornar o software mais eficiente na solução do problema”.

d) Velocidade do Projeto: a velocidade do projeto pode ser definida por tempo ou escopo. Utilizando o tempo como parâmetro é preciso que seja recalculada a quantidade de Estórias dos Usuários que serão implementadas na próxima iteração, para que todas as iterações utilizem o mesmo tempo de desenvolvimento, se possível.

e) Rumo do Projeto: o rumo do projeto pode necessitar de alterações que reflitam a realidade, uma vez que o projeto avança o cliente entende mais sobre o sistema e o desenvolvedor mais sobre o negócio.

f) Plano de Iteração: são escolhidas as Estórias dos Usuários que foram definidas no Planejamento da Versão e priorizadas pelo cliente. As Estórias dos Usuários da iteração anterior que não funcionaram, devem constar novamente da iteração. As tarefas que serão realizadas devem durar, no máximo, três dias, caso superem esse prazo, devem ser divididas em tarefas menores, e as que durarem menos de um dia devem ser agrupadas.

g) Mudança de Pessoas: significa não deixar os desenvolvedores em pares fixos ou em uma única parte do código.

h) Reuniões Rápidas: tem como objetivo a comunicação entre os desenvolvedores. São realizadas todos os dias em pé de forma rápida e direta, objetivando fornecer uma visão geral do projeto. Nessas reuniões, são discutidas como as regras da XP devem ser cumpridas pela equipe.

- **Projeto**

Astels *et al.* (2002, p.121) afirmam que a XP defende uma abordagem evolucionária para o projeto, ou seja,

> ... a XP não é contra o projeto, mas está relacionada ao projeto contínuo. A XP é contra os grandes projetos iniciais. A XP faz pequenos projetos durante todo o tempo e uma tarefa de cada vez. (2002, p. 120)

A seguir uma breve descrição das práticas adotadas na atividade do Projeto de desenvolvimento de software pela XP:

a) Simplicidade: deve-se utilizar uma codificação simples, clara e inteligível. Esta prática, é, um dos temas polêmicos da XP, pois, termos como “simples”, “clara” e “inteligível” são subjetivos.

b) Metáforas: segundo Mesquita (2002), uma metáfora é utilizada ao invés de uma arquitetura formal para descrever como o sistema funciona. A metáfora serve como uma linguagem comum entre cliente e desenvolvedores, facilitando a compreensão do sistema que está sendo desenvolvido.

c) Cartões CRC: CRC é o acrônimo de Classe, Responsabilidade e Colaboração. Permitem que as equipes do projeto contribuam com o *design*, além de facilitar o entendimento de conceitos, como por exemplo, da orientação a objetos.

d) Soluções: são soluções rápidas objetivando reduzir os riscos e deixar as Estórias dos Usuários possíveis de serem implementadas. É uma codificação para sanar algum problema tecnológico ou testar alguma solução.

e) *Refactoring*: consiste em melhorar o código que já está funcionando sem comprometer o comportamento externo do software. Conforme Mesquita (2002), *refactoring* não deve acrescentar funcionalidade, apenas “limpá-la”.

- **Teste**

É considerado um fator central da “filosofia” XP, e possui como característica fundamental testar primeiro antes de codificar. Astels *et al.* (2002, p.7), apontam uma vantagem na adoção da abordagem de testar primeiro pois fornece a possibilidade de uma definição ou documentação do comportamento desejado. A seguir uma descrição das práticas da atividade de Teste conforme Mesquita (2002).

a) Testes de Unidade: são testes escritos anteriormente ao processo da codificação. Tem como função, obrigar o desenvolvedor a analisar melhor o que efetivamente será codificado, permitindo com isso, um *feedback* imediato sobre o código gerado.

b) Integração Contínua: utiliza-se um *framework* para criação de testes para executá-los de forma simples na busca de erros na integração, permitindo com isso, que os

desenvolvedores trabalhem sempre com a última versão, evitando a duplicidade de alguma funcionalidade.

c) Testes de Aceitação: as funcionalidades do sistema são validadas através dos testes executáveis. São realizados com freqüência, e os resultados devem ser publicados a rotineiramente a toda equipe.

d) Propriedade Coletiva do Código Gerado: não existem desenvolvedores “donos” do código escrito, ou seja, toda equipe deve saber sobre o código e pode alterá-lo a qualquer momento. Para que isto seja possível, se faz necessário ter testes de unidade bem elaborados, permitindo segurança ao desenvolvedores que trabalham neste código.

e) Integração Freqüente: faz com que desenvolvedores que trabalham em um ambiente com Propriedade Coletiva do Código Gerado nem percebam se o código no qual trabalham foi alterado. Um fator fundamental para viabilizar a manipulação do código por vários desenvolvedores é a adoção de padrões de codificação.

- **Codificação**

A codificação também é uma atividade central do projeto na XP, deve ser padronizada e organizada de tal forma que qualquer pessoa envolvida no projeto não reconheça quem codificou determinado código, conforme afirma Beck (2000). A seguir, uma descrição das práticas adotadas na atividade da Codificação da XP.

a) Cliente Sempre Disponível: o cliente está presente e a disposição em todas as fases do desenvolvimento, de preferência no mesmo local ou sala onde estão os desenvolvedores. Esta prática pode ser considerada uma barreira na aceitação da XP conforme aponta Beck (2000), pois muitas vezes o cliente não dispõe de todo tempo disponível para acompanhar as fases de desenvolvimento do software.

b) Padrões de Codificação: é obrigatória a adoção de padrões e formato dentro dos códigos, facilitando com isso a integração da equipe. A XP não define qual é o padrão, apenas indica a necessidade. Normalmente surge como resultado das práticas: Propriedade Coletiva do Código, Programação em Par e Mudanças de Pessoas.

c) Teste de Unidade Primeiro: são escritos antes da codificação da unidade em si. Além de obrigar o desenvolvedor a analisar o que vai ser codificado, permite ainda receber *feedback* imediato sobre a qualidade do seu código. Os testes podem indicar ao desenvolvedor que seu trabalho está finalizado.

d) Programação em Par: toda a construção do código deve ser desenvolvida por duas pessoas trabalhando em conjunto utilizando o mesmo computador. Para que isto ocorra, a XP determina que ambos devam trabalhar em perfeita harmonia e sintonia. Juntamente com a prática Cliente Sempre Disponível, é considerada também uma das maiores barreiras para a aceitação de XP, visto que a maioria dos desenvolvedores codifica solitariamente.

e) Excesso de Tempo: A XP não recomenda o trabalho semanal acima de 40 horas, pois acredita que um funcionário não é produtivo se ultrapassar esta carga horária.

2.2.4.3 Considerações finais sobre a XP

Durante o desenvolvimento do projeto, o cliente tem fundamental importância no rumo dado ao software, uma vez que o cliente fornece os dados sobre o que deseja obter com o software. A cada *release* o cliente consegue vislumbrar os resultados gerados e pode mudar o que não considerar adequado. Deste modo, os requisitos do cliente e a qualidade serão alcançados de acordo como a visão do cliente.

No que se refere à qualidade do desenvolvimento em um âmbito global são utilizados os testes para certificar os resultados. Esses testes são a base da XP e estão inclusos principalmente na codificação. Portanto, a XP, apesar de não possuir algumas características para uma certificação de qualidade oficial como a ISO, como por exemplo, uma documentação de todos seu processo, permite, de certa forma, o desenvolvimento de software com qualidade.

De modo geral, os limites para se implementar um processo como a XP não estão totalmente definidos, e possui benefícios e limitações que necessitam de uma análise mais aprofundada antes de sua aplicação como afirma Sommerville (2003, p.44). O autor

acredita que ainda “é muito cedo para dizer se a XP se tornará uma abordagem de uso dominante para o desenvolvimento de software”.

2.3 Comparação entre os métodos ágeis

Nesta seção é realizada uma análise comparativa entre os métodos ágeis estudados: Scrum, FDD, DSDM e a XP. Para realizar a análise dos resultados, baseada na verificação do grau de satisfação dos princípios ágeis pelos métodos ágeis, propõe-se uma escala ordenada de três categorias: *Não atendido*, *Parcialmente Atendido* e *Atendido*. Os princípios ágeis são a base do pensamento ágil servindo como uma guia para todos os métodos ágeis. Eles auxiliam na definição do quão ágil é um método, baseado, principalmente, na forma em que são usados pelos métodos. As três categorias são:

a) Não atendido (N): Há pouca evidência de que o princípio foi satisfeito, ou o método propõe diferentes propostas ao manifesto ágil.

b) Parcialmente atendido (P): Existem evidências de uma prática sistemática no método na satisfação do princípio.

c) Atendido (A): Existe uma prática na satisfação do princípio existindo evidências significativas no método.

A seguir, um detalhamento das justificativas e as notas atribuídas aos métodos ágeis quando comparados aos princípios ágeis.

A Figura 18 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 1: A prioridade é satisfazer o cliente através da entrega contínua e rápida de uma versão do software com valor agregado.

MA	Categoria	Justificativa da categoria atribuída.
XP	A	Atende completamente este princípio, pois tem como uma de suas práticas a entrega freqüente; além de pregar a iteração e satisfação total do cliente.
SCRUM	A	Através de ciclos rápidos (*sprint*), são desenvolvidas as funcionalidades com valor para o cliente. Visa trabalhar para o cliente.
FDD	N	Não há especificações diretas para este princípio.
DSDM	A	Objetiva o rápido desenvolvimento de partes da aplicação, sempre orientado aos requisitos e clientes.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 18 Justificativa para o princípio 1

A Figura 19 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 2: Alterações sobre os requisitos são bem vindas, mesmo que ela ocorra tarde, pois isto dá uma vantagem competitiva ao cliente.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Apresenta princípios e técnicas, como *feedback* rápido, mudanças incrementais e integração contínua que focalizam e acolhem a mudança.
SCRUM	A	Os requisitos vão sendo definidos durante a evolução do projeto em forma de funcionalidades, suprindo as novas necessidades.
FDD	P	Através do desenvolvimento por característica, visa suprir as novas necessidades, sem entretanto, especificar tarefas específicas para o tratamento das mudanças de requisitos.
DSDM	A	Entende que deve haver especificações freqüentes para os requisitos, que o desenvolvimento deve ser dinâmico e adaptar-se às mudanças.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 19 Justificativa para o princípio 2

A Figura 20 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados Princípio 3: Entrega de software freqüentemente, a cada semana ou a cada mês, sempre visando o menor tempo possível.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Atende completamente ao princípio, pois além de estabelecer um *feedback* rápido, apresenta como prática a entrega freqüente.
SCRUM	A	Estipula ciclos de geralmente 30 dias para a entrega de novas funcionalidades.
FDD	A	Através de ciclos curtos, de no máximo duas semanas, especifica que as características desenvolvidas devem ser agregadas ao *software*.
DSDM	A	Define ciclos curtos que variam de poucos dias a poucas semanas e que resultam em protótipos referentes às freqüentes iterações e incrementos.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 20 Justificativa para o princípio 3

A Figura 21 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 4: Os especialistas no negócio e os desenvolvedores trabalham juntos diariamente no projeto.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Atendido completamente pela prática do cliente presente e sua filosofia de freqüente interação entre a equipe.
SCRUM	A	Os *stakeholders* em algum projeto Scrum tem uma integração contínua e diária, trabalham com um senso de equipe muito intenso.
FDD	N	Não existem práticas específicas para este princípio.
DSDM	N	Nada especificando a efetivação deste princípio.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 21 Justificativa para o princípio 4

A Figura 22 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 5: Manter uma equipe motivada fornecendo ambiente e confiança necessários para o trabalho a ser executado.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Possui como valor a comunicação entre a equipe, servido de motivação, além de fornecer um ambiente integrado onde se concentra a equipe.
SCRUM	A	Equipes são encorajadas e possuem autoridade no processo, trabalham juntas em um ambiente onde encontram apoio e recursos.
FDD	N	Não há especificações diretas para este princípio.
DSDM	A	Procura encorajar a equipe, visando a colaboração e a cooperação.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 22 Justificativa para o princípio 5

A Figura 23 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 6: A maneira mais eficiente de comunicação da equipe é através de uma conversa face-a-face.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Um dos valores, é a comunicação entre a equipe, incentivando a conversa cara-a-cara.
SCRUM	A	Trabalhando no mesmo ambiente e através de reuniões diárias.
FDD	N	Não há especificação clara deste princípio.
DSDM	A	Encoraja e cooperação entre a equipe. Uma prática referente a isso é a utilização de *workshops* que inclui os *stakeholders*.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 23 Justificativa para o princípio 6

A Figura 24 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 7: A simplicidade é essencial.

MA	Categoria	Justificativa da categoria atribuída
XP	A	O projeto simples é uma das práticas da XP.
SCRUM	A	O desenvolvimento simplificado é um dos valores, que procura trabalhar com as soluções mais simples.
FDD	N	Não há especificações diretas para este princípio.
DSDM	N	Não há especificação quanto a este princípio.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 24 Justificativa para o princípio 7

A Figura 25 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 8: Software funcionando é a medida primária de progresso.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Além de princípio ágil, *software* funcionado é um princípio da XP, completado pelas entregas freqüentes e integração contínua.
SCRUM	A	Busca a entrega do incremento.
FDD	N	Possui uma técnica própria, chamada de ETVX, para medir o progresso do projeto, bem diferente do proposto por este princípio.
DSDM	A	O desenvolvimento é voltado para iterações e incrementos, gerando protótipos que fornecem uma base de progresso.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 25 Justificativa para o princípio 8

A Figura 26 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 9: A atenção contínua à excelência técnica e ao bom projeto melhoram a agilidade.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Atendida por práticas como o projeto simples, testes e programação em pares, que visam melhorar continuamente o projeto.
SCRUM	P	Não há especificações diretas para este princípio, contudo, não existem barreiras para sua aplicação.
FDD	P	Práticas como a modelagem dos objetos de domínio e inspeções, demonstram a intenção de atingir uma excelência técnica, entretanto, não existem atividades específicas para tanto.
DSDM	A	Especificações, validação do propósito do negócio e projeto, além de testes integrados para analisar o andamento do projeto, são práticas abrangidas pelo DSDM e que justificam essa qualificação.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 26 Justificativa para o princípio 9

A Figura 27 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 10: As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.

MA	Categoria	Justificativa da categoria atribuída
XP	A	A comunicação entre a equipe, papéis bem definidos, padrões e planejamento são práticas que englobam o conceito de auto-organização.
SCRUM	P	A organização é um ponto central no Scrum, as equipes são auto gerenciadas através de práticas e papéis responsáveis por ela, como as reuniões diárias e o SCRUM Master.
FDD	N	Não há especificações diretas para este princípio.
DSDM	A	Não há práticas definidas para este princípio, porém, o DSDM manifesta interesse neste sentido quando estabelece a colaboração e aproximação cooperativa entre a equipe.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 27 Justificativa para o princípio 10

A Figura 28 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 11: A equipe reflete, a intervalos regulares, sobre como se tornar mais efetivo, então se ajusta e otimiza o seu comportamento.

MA	Categoria	Justificativa da categoria atribuída
XP	P	Não existe prática ou conceito especificando este princípio, porém, deixa aberta esta possibilidade. Um exemplo desta abertura são as "reuniões em pé" que acontecem em alguns projetos.
SCRUM	A	Além das reuniões de planejamento antes dos *sprints*, o SCRUM estabelece a prática de reuniões diárias para acompanhar e refletir sobre o andamento do projeto.
FDD	N	Não há especificações diretas para este princípio.
DSDM	P	Através de *workshops*, procura promover o debate dentro do projeto.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 28 Justificativa para o princípio 11

A Figura 29 apresenta uma justificativa das categorias atribuídas ao métodos ágeis quando comparados ao Princípio 12: Promovem desenvolvimento sustentável. Todos *stakeholders* devem ser capazes de manter um ritmo constante indefinidamente.

MA	Categoria	Justificativa da categoria atribuída
XP	A	Cliente presente, projeto simples, mudanças incrementais auxiliam o desenvolvimento a manter um ritmo sustentável.
SCRUM	N	Não há especificações diretas para este princípio.
FDD	N	Não há especificações diretas para este princípio.
DSDM	A	A equipe deve ser completamente dedicada para o sucesso do projeto. Isso também inclui a participação do cliente.

Fonte: Primária
Legenda: N:Não Atendido - P: Parcialmente Atendido - A:Atendido

Figura 29 Justificativa para o princípio 12

Na Figura 30 apresenta-se um resumo com todos os princípios do manifesto ágil (Manifesto, 2003) que foram utilizados nesta comparação entre os métodos ágeis estudados e as categorias atribuídas.

Nº	Princípio ágil	Scrum	FDD	DSDM	XP
1	A prioridade é satisfazer o cliente através da entrega contínua e rápida de uma versão do software com valor agregado.	A	N	A	A
2	Alterações sobre os requisitos são bem vindas, mesmo que ela ocorra tarde, pois isto dá uma vantagem competitiva ao cliente.	A	P	A	A
3	Entrega de software freqüentemente, a cada semana ou a cada mês, sempre visando o menor tempo possível.	A	A	A	A
4	Os especialistas no negócio e os desenvolvedores trabalham juntos diariamente no projeto.	A	N	N	A
5	Manter uma equipe motivada fornecendo ambiente e confiança necessários para o trabalho a ser executado.	A	N	A	A
6	A maneira mais eficiente de comunicação da equipe é através de uma conversa face-a-face.	A	N	A	A
7	A simplicidade é essencial.	A	N	N	A
8	Software funcionando é a medida primária de progresso.	A	N	A	A
9	A atenção contínua à excelência técnica e ao bom projeto melhoram a agilidade.	P	P	A	A
10	As melhores arquiteturas, requisitos, e projetos emergem de equipes auto-organizadas.	P	N	A	A
11	A equipe reflete, a intervalos regulares, sobre como se tornar mais efetivo, então se ajusta e otimiza o seu comportamento.	A	N	A	P
12	Promovem desenvolvimento sustentável. Todos *stakeholders* devem ser capazes de manter um ritmo constante indefinidamente.	N	N	P	A

Fonte: Adaptada de Hartmann (2003)

Figura 30 Comparação dos métodos ágeis com os princípios do manifesto ágil.

A Figura 30, mostra que 92% das notas atribuídas ao XP foram da categoria *Atendido*, com isso, pode-se concluir que possui um maior atendimento em relação ao atendimento dos princípios propostos no manifesto ágil em comparação com os demais métodos estudados.

O método Scrum obteve o segundo lugar, com uma diferença mínima sobre a XP. Esta diferença se deu basicamente, porque o Scrum não descreve em seu método a necessidade de atenção à excelência técnica (princípio 9) bem como da ausência da auto organização da equipe - (o método Scrum exige a presença de uma pessoa externa à equipe para realização do princípio 10, no caso, o *Scrum Master*). O princípio 12, também influenciou nesta diferença, porque o Scrum não especifica que as equipes devem manter um ritmo constante no trabalho.

O método FDD utiliza algumas características presentes nos demais métodos ágeis como o desenvolvimento incremental e iterativo, sendo fortemente baseado na modelagem dos dados. Highsmith (2002, p.284), indica que este método pode ser classificado como intermediário, entre os ágeis e os “pesados”. Este método obteve um alto índice na categoria *Não Atendido*, aproximadamente 75%, porque não atende várias das características, ao contrário do Scrum e XP. Por exemplo, o princípio 8, software funcionando é medida primária de progresso, não é atendido, pois o método FDD propõe sua própria técnica para que o progresso do projeto seja medido e divulgado. Pela busca da subdivisão do projeto em características que serão implementadas incrementalmente, o método FDD não define que, por exemplo, o software com valor agregado, deva ser entregue o mais rápido possível, isto, justifica porque o princípio 8 está na categoria de *Não atendido*.

Como o FDD procura seguir uma rotina de projeto e construção mais rígida, definindo uma seqüência de passos os quais devem ser seguidos, o princípio 10 não é atendido pelo FDD, pois este princípio defende que deve-se priorizar o trabalho de tal forma que as principais funcionalidades sejam realizadas, mesmo em detrimento das funcionalidades menos relevantes, como destaca Hartmann (2003, p.61).

O DSDM é um dos métodos mais bem definidos e organizados, apresentando características muito similares ao RUP. Devido à esta proximidade com o RUP, alguns autores, como Boehm (2004, p.21), não o classificam como um método ágil. Por outro lado, Highsmith (2002, p.254) entende que os princípios que norteiam o DSDM são muito próximos aos apresentados no manifesto ágil, gerando, de certa forma, uma oposição à idéia de Boehm sobre o método DSDM, com relação ao atendimento aos princípios do manifesto ágil. Na análise realizada, percebe-se uma carência no atendimento aos princípios 4 e 7. Como no FDD, o método DSDM não possui equipes auto-organizadas, pois estas equipes necessitam de pessoas responsáveis pela gestão, como afirma Abrahamsson (2002, p.257). Por se tratar de um método extremamente organizado e disciplinado, o princípio da simplicidade (item 7) não é atendido no DSDM.

Cabe ressaltar que os autores da XP e Scrum participaram do manifesto ágil, enquanto os autores do FDD e DSDM não participaram.

Outra análise é com relação ao estágio de maturidade dos métodos ágeis. Entende-se por estágio de maturidade o tempo que método está sendo utilizado e sua efetiva utilização pelas organizações. A Figura 31 apresenta um resumo dos métodos ágeis estudados em relação ao estágio de maturidade em que cada um se encontra e um breve resumo conforme estudos do Cutter Consortium[22].

Método ágil	Resumo	Estágio de Maturidade
FDD	Foca no projeto e implementação.	Em desenvolvimento
Scrum	Detalha o gerenciamento do ciclo de vida do projeto.	Em desenvolvimento
XP	É postura. Práticas de gerenciamento de engenharia de software são descartadas.	Maduro
DSDM	Baseia-se no RAD e prototipação. É proprietário	Maduro

Fonte: adaptada de Cutter Consortium.

Figura 31 Maturidade dos métodos ágeis.

[22] Mais informações em http://www.cutter.com/index.shtml

Para finalizar, os métodos ágeis *Crystal Method*, ASD (*Adaptative Software Development*), *e Lean Development* (LD) não foram estudados porque carecem de estudos mais avançados, especialmente realizados por instituições de ensino superior. O que se encontra são materiais disponibilizados pelos próprios desenvolvedores destes métodos. Além disso, há bastante referência a esses métodos, porém nada muito explicativo ou detalhado. A Modelagem Ágil (MA) também não foi selecionada, para o presente estudo, por não apresentar um processo e ter que, por isso, ser agregada a outro método ágil, como destaca Ambler (2004).

Após análises realizadas, o método ágil Scrum, foi escolhido para o desenvolvimento do presente trabalho devido a:

- É um dos métodos ágeis com maior possibilidade de escalabilidade, ou seja, capacidade de se adaptar a projetos maiores, como destaca Boehm (2004, p.169);
- Por estar em "fase de desenvolvimento", este método carece de mecanismos para controle no processo de gerenciamento, tão fundamentais em áreas como a engenharia de requisitos.
- Destacou-se, conforme demonstrado na Figura 30, como o mais ágil dentre os métodos que gerenciam o processo de desenvolvimento de software. Vale destacar, que a XP, não é considerada como um método para gerenciamento de processo.

2.4 Métodos Ágeis e Modelos de Qualidade de Software

Nesta seção, serão apresentados trabalhos relacionados que analisam o processo de desenvolvimento de software dos métodos ágeis tendo como base os modelos SW-CMM e CMMI.

Na verdade, a maioria destes trabalhos, mostra uma comparação de alguns métodos apenas com o modelo SW-CMM e não o CMMI, isto ocorre, porque o modelo CMMI está a apenas aproximadamente 2 anos no mercado, justificando, de certa forma, a pouca

existência de estudos com este modelo. Até a data de hoje, segundo dados da ISDBrasil[23], somente a IBM possui certificação CMMI nível 3. Observa-se, que não foram encontrados na literatura, trabalhos sugerindo adaptações, extensões nos métodos ágeis para adequação as necessidades do CMMI.

Cabe salientar, que as análises detalhadas no método ágil Scrum, em relação às práticas específicas do modelo CMMI não foram encontradas na literatura. Alguns autores, como Mark Paulk, Richard Turner, Barry Boehm, Nawrocki e Frank Maurer, discutem a utilização dos métodos ágeis quando comparados com modelos de qualidade de processo de software. A seguir, alguns trabalhos relacionados.

2.4.1 Métodos ágeis e CMMI

Estudos realizados por Turner e Jain (2002, p.153), verificaram como o modelo CMMI pode se adequar aos métodos ágeis. Perceberam que existem diferenças significativas entre os métodos ágeis, constatando que 42% das áreas de processo do modelo CMMI, estão em conflito com práticas adotadas pelos métodos ágeis.

Os autores Turner e Jain acreditam que estas diferenças podem ser minimizadas através de adoção de políticas organizacionais no processo de desenvolvimento de software, com isso, justificam que os métodos ágeis e o modelo CMMI podem ser aplicados em conjunto numa organização. Neste trabalho, os autores não avaliaram em específico nenhum método ágil.

2.4.2 Scrum e SW-CMM

Como citado no início desta seção, existem poucos trabalhos relacionados analisando o Scrum conforme as áreas de processo do modelo SW-CMM. Paulk (2001,

---

[23] É uma empresa internacional, atuando na América do Sul, focada exclusivamente em qualidade de processos baseada em modelos, credenciada pelo SEI. Mais informações em http://www.isdbrasil.com.br

p.19) em seus estudos, aponta que método ágil Scrum atende a apenas 22,22 % , ou seja, 4 de todas 18 áreas de processo são atendidas pelo Scrum.

O mesmo percentual, 22,22%, ocorre quando a situação passa para Parcialmente Atendido. Já, na situação de Não Atendido este percentual eleva-se para 55,55%, pois o método ágil Scrum não atende 10 das 18 áreas de processo do modelo SW-CMM. A seguir, a Figura 32 mostra as áreas de processo do modelo SW-CMM com relação ao método ágil Scrum.

Nível	Áreas de Processo do modelo CMM	Situação
2	Gerenciamento de Requisitos	Atendido
2	Planejamento do projeto	Atendido
2	Visão geral e acompanhamento do projeto	Atendido
2	Gerenciamento de sub-contratos	Não Atendido
2	Garantia da qualidade do software	Não Atendido
2	Gerenciamento de configuração	Parcialmente Atendido
3	Foco do processo organizacional	Parcialmente Atendido
3	Definição do processo organizacional	Parcialmente Atendido
3	Programa de treinamento	Não Atendido
3	Gerenciamento de software integrado	Não Atendido
3	Engenharia de produto de software	Atendido
3	Coordenação intergrupos	Parcialmente Atendido
3	Revisão conjunta	Não Atendido
4	Gerenciamento quantitativo dos processos	Não Atendido
4	Gerenciamento da qualidade de software	Não Atendido
5	Prevenção de defeitos	Não Atendido
5	Gerenciamento de mudanças tecnológicas	Não Atendido
5	Gerenciamento de mudanças no processo	Não Atendido

Fonte: SEI

Figura 32 Scrum *versus* SW-CMM.

Pela análise, percebe-se que o método ágil Scrum não atende, por completo, nenhum dos níveis de maturidade do modelo SW-CMM. Ressalta-se ainda, que o método ágil Scrum não atende principalmente as áreas de processo dos níveis 4 e 5. Isto se justifica, de certa forma, porque estas áreas referem-se basicamente ao controle dos processos através de medidas, buscando a melhoria continua, visto que o método Scrum preocupa-se basicamente no gerenciamento do processo de desenvolvimento de software.

Uma instituição que está dedicando atenção, com pesquisas e estudos sobre os métodos ágeis e suas aplicações é a Universidade de Calgary no Canadá. Nesta universidade, professores pesquisadores como Amy Law e Frank Maurer, concentram-se na avaliação e aplicação do método ágil Scrum no mercado. Além da avaliação da utilização deste método no mercado, verificam, também, suas relações com modelos de qualidade de software como o SW-CMM.

Em um estudo[24], Amy Law analisou se o método ágil Scrum estava de acordo com as exigências do nível 2 do SW-CMM. Constatou que este método não atende as áreas de processo Gerenciamento de Sub-Contratos e Revisão conjunta, confirmando, de certa forma, as análises realizadas por Paulk (2001). As áreas de processo como Garantia da Qualidade do Software e Gerenciamento de Configuração atendem parcialmente. Por outro lado, as áreas de processo Gerenciamento de Requisitos, Visão geral e Acompanhamento do Projeto e Planejamento do Projeto são plenamente atendidas por este método.

2.4.3 XP e SW-CMM

Paulk (2001, p.19) aponta que XP tem "boas" práticas da engenharia de software, logo, pode trabalhar tanto com o SW-CMM como qualquer outro modelo. Porém, o autor destaca ainda, que nem todas práticas da XP estão de acordo com o modelo SW-CMM. Ainda conforme Paulk (2001, p.24), a XP não atende as áreas de processo do SW-CMM, Gerenciamento de Contrato, Programa de Treinamento, Gerenciamento de Integração de Software, Gerenciamento Quantitativo do Processo e Gerenciamento da Qualidade. Outras áreas de processo do modelo SW-CMM, como por exemplo, Gerenciamento de Requisitos, Planejamento e Projeto de Software, estão plenamente atendidas pelas práticas da XP.

Reifer (2003) comenta que as práticas da XP e o SW-CMM também são compatíveis, mesmo que inexistam orientações de "como" realizar esta compatibilização. As organizações que utilizam práticas da XP podem adequar-se ao SW-CMM desde que, estas práticas, sejam executadas de forma "racional". Entende ainda, que os métodos ágeis estão

[24] Disponível em http://sern.ucalgary.ca/~bowen/623/scrum/ScrumAssessment2.htm

de acordo com as exigências das práticas existentes nos níveis 2 e 3 do SW-CMM. O autor sugere que seja utilizado o SW-CMM como uma estrutura na adoção das práticas propostas pela XP.

A seguir, uma análise realizada por Reifer (2003), detalhando se a XP atende aos níveis de maturidade do SW-CMM.

a) Nível 1: Neste nível, o processo é improvisado, sua capacidade é imprevisível. A XP, segundo Reifer (2003), através da fase do Planejamento, supera as exigências esperadas por este nível.
b) Nível 2: A XP especifica claramente um número das práticas referentes a este nível, como treinamento e medidas. A equipe é treinada no começo do projeto. As práticas, geralmente são medidas e analisadas pela equipe. A programação em par, por exemplo, assegura que todos os colaboradores saibam o que está acontecendo no projeto, com isso, é possível repetir processos bem sucedidos. Desta forma, o nível 2 do modelo SW-CMM, segundo o autor, é satisfeito pela XP.
c) Nível 3: Neste nível, o processo operacional está definido, ele é capaz de atingir metas de funcionalidade, prazo e custo. Devido à existência de um processo de gerência na XP que desempenha a função de verificação e definição de padrões, o autor acredita que ocorre a análise do desempenho real do projeto. Desta forma, a XP está de acordo com o nível 3 no SW-CMM.
d) Nível 4: O processo é continuamente medido, sua capacidade é administrada. A XP requer que os objetivos da qualidade dos produtos sejam ajustados através dos testes funcionais e teste de unidade. Com os testes, obtêm-se valores quantitativos, que poderão ser utilizados para a melhoria do processo. O autor acredita que este nível não está plenamente atendido, porque não existem práticas na XP que meçam a capacidade do processo.
e) Nível 5: O melhoramento contínuo do processo está implantado, sua capacidade aumenta continuamente, novas tecnologias são testadas e introduzidas. Este nível não está atendido pela XP porque não existem práticas na XP que priorizem o melhoramento contínuo.

Opperthauser (2003), mostra que “alguns autores entendem que as práticas da XP são pensadas como uma aproximação indisciplinada do desenvolvimento de software, faltando alguns aspectos como a gerência eficaz. Entretanto, o desenvolvimento ágil não exclui o uso de processos da gerência formal”.

Em outro trabalho, Nawrocki *et al.* (2002, p.288), através de um exemplo de aplicação realizado com estudantes do curso de computação da *Poznan University of Technology*, verificaram que é válida a utilização do método ágil XP quando comparado ao modelo SW-CMM em projetos de desenvolvimento de software. Comparou-se, neste trabalho, especificamente o nível 2 (repetível) do SW-CMM. Os relatos indicam que as áreas de processo do nível 2 do SW-CMM, Gerenciamento de Requisitos, Planejamento do projeto, Visão geral e Acompanhamento do Projeto, Garantia da Qualidade do Software e Gerenciamento de Configuração, são atendidas plenamente pelas práticas da XP. Exceção, ocorreu na área de processo Gerenciamento de sub-contratados, isto deve-se porque nos estudos realizados os softwares desenvolvidos foram solicitados pelos próprios estudantes e não por terceiros.

Relatam ainda, que encontraram resistências pelos alunos principalmente em algumas práticas propostas pela XP, como programação em pares e *refactoring*. Os autores afirmam ainda que nos experimentos realizados, a programação em pares exigiu um acréscimo de 50% de esforço comparando com a programação individual. Concluíram também, que o método ágil XP pode ser uma alternativa ao processo de desenvolvimento de software tradicional, principalmente quando comparado a modelos de qualidade de software como o SW-CMM.

Como o método ágil Scrum não define as atividades e as técnicas para todo processo de Gerenciamento de Requisitos, esta lacuna, é um ponto a ser investigado neste método. Boehm (2004, p.169), considera o Scrum como um dos poucos métodos ágeis aptos a trabalhar também em grandes projetos. Ressalta-se que, conforme Paulk (2002), o modelo CMMI também é indicado para grandes projetos.

3 Uma análise do Scrum conforme abordagem CMMI

Neste capítulo, o método ágil Scrum é avaliado segundo as perspectivas do modelo CMMI, nas áreas de processo Gerenciamento de Requisitos e Desenvolvimento de Requisitos pertencentes à categoria Engenharia, As demais áreas[25] desta e de outras categorias[26], não serão exploradas devido à delimitação do trabalho.

Como visto anteriormente, os componentes do modelo CMMI são informativos, esperados e requeridos. Estes componentes contribuem significativamente na definição da capacidade de uma organização. Práticas e sub-práticas são componentes informativos que auxiliam no atendimento ao modelo CMMI. Cada prática está relacionada a uma ou mais metas, de forma que elas ajudam a alcançá-las. Cabe ressaltar que apesar de cada prática não ser considerada obrigatória para alcance de um nível, são, em geral, aplicáveis. Avaliações, como o SCAMPI V1.1, fazem observações relacionadas às práticas e sub-práticas. Segundo Paulk (2001), "Práticas não são requeridas e devem existir implementações alternativas, mas isto não tira a responsabilidade de realizar julgamentos profissionais sobre cada prática e meta associada"

3.1 Metodologia para realização da análise

Devido à significativa importância das práticas para entendimento do modelo CMMI e alcance das metas, cada produto de trabalho resultante das práticas específicas das áreas de processo Gerenciamento de Requisitos e Desenvolvimento de Requisitos do modelo

[25] Demais Áreas de Processo da categoria Engenharia: Integração do produto, Solução Técnica, Validação, Verificação.
[26] Além desta categoria, o modelo CMMI possui as categorias Gerenciamento de Processo, Gerenciamento de Projeto e Apoio.

CMMI é analisado e é verificado se o Scrum atende este produto conforme o modelo CMMI. Para isso, elaborou-se uma escala ordenada de três categorias, com a determinação de classificação do atendimento ao Scrum. As três categorias são:

N: Não atendido. Há pouca evidência de que o atributo foi satisfeito.

P: Parcialmente atendido. Existem evidências de uma prática sistemática no Scrum na satisfação do atributo.

A: Atendido. Existe uma prática na satisfação do atributo, e há evidências significativas em todo o Scrum.

Para auxiliar na análise e verificação no atendimento do Scrum às exigências das práticas específicas, utilizou-se também como referência o Método Quicklocus. Este método, dentre outras características, consiste basicamente na elaboração de uma pergunta ao processo que será analisado e verificado. Esta forma de questionamento é muito utilizada no método ARC (*Appraisal Requirements for CMMI)*, que combina as características do CBA-IPI *(CMM – Base Appraisal for Internal Process Improvement)* e SCE *(Software Capability Evaluation).*

Caso a resposta for negativa ao questionamento, isto significará que o Scrum não atende a exigência desta prática, posicionando-se então na categoria “Não Atendido”. Se ocorrer algum tipo de dúvida na resposta ao questionamento no Scrum com relação a esta prática, então se posicionará na categoria “Parcialmente Atendido”. E, por fim, caso a resposta seja positiva ao questionamento elaborado, esta prática específica em relação a determinação de classificação do atendimento ao Scrum se posicionará na categoria de “Atendido”.

Por exemplo, na prática REQM SP1.1-1, o questionamento será: *Os requisitos são esclarecidos com quem define os requisitos?* A idéia principal é identificar, através da análise, pontos fortes e pontos fracos no Scrum quando relacionado ao modelo CMMI, a fim de realizar um plano de melhoria para corrigir principalmente os pontos fracos e conhecer melhor os pontos fortes.

Além das práticas específicas, o modelo CMMI possui também as sub-práticas que, não serão analisadas neste trabalho, pois Chrissis *et al.* (2003, p.25), comentam que estas práticas apenas auxiliam no entendimento e interpretação das práticas específicas.

3.2 Análise da área de processo Gerenciamento de Requisitos

A seguir, faz-se uma análise do Scrum conforme as práticas específicas da área de processo Gerenciamento de Requisitos (REQM) da categoria Engenharia do modelo CMMI.

3.2.1 Obter o entendimento dos requisitos - SP 1.1-1

No Scrum, os requisitos são levantados diretamente com os clientes, e os critérios de seleção dos requisitos são alocados para cada *Sprint* durante a divisão de tarefas. Os requisitos são entendidos pelos participantes do projeto e colocados numa lista de itens denominada de *Product Backlog*, que representa "o que" deverá ser analisado pela equipe para posterior desenvolvimento. Depois do entendimento dos requisitos, o *Product Backlog* será dividido em tarefas que serão designadas por itens e descrições, tempo e a definição de quem a realizará. O resultado desta etapa é chamado de *Sprint Backlog List*.

Diante do exposto, conforme sugestão do método Quicklocus, faz-se então o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *Os requisitos são esclarecidos com quem os define?* Para responder esta pergunta remete-se ao Scrum, e se percebe que neste método os requisitos são definidos juntamente com os clientes, isto ocorre devido à presença constante do cliente durante o entendimento de suas necessidades, desta forma, esta prática está atendida no Scrum.

A seguir, a Figura 33 apresenta os produtos de trabalho resultantes da prática específica REQM SP1.1-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Lista de critérios para distinguir requisitos fornecidos	-	-	X
2) Critério para avaliação e aceitação dos requisitos	-	-	X
3) Análise dos resultados conforme lista de critérios estabelecidos	-	-	X
4) Elaboração de um conjunto de requisitos aceitos	-	-	X

Fonte: primária

Figura 33 Produtos de trabalho da REQM SP1.1-1

Na reunião inicial no Scrum, o *Sprint Planning Backlog* são apresentados os critérios para a elaboração do *Product Backlog*. Percebe-se, então, que os itens 1 e 2 da Figura 33 são Atendidos pelo Scrum. Como a elaboração de um conjunto de requisitos aceitos e a análise dos resultados, itens 3 e 4 da Figura 33 respectivamente, são justamente os resultados finais da análise do *Product Backlog,* conhecido como *Sprint Backlog,* todos os produtos de trabalho resultantes da REQM SP1.2-2 do modelo CMMI estão Atendidos no Scrum.

3.2.2 Obter o compromisso com os requisitos: SP 1.2-2

O *Scrum Master* e o *Product Owner*, trabalham para que todos participantes do projeto entrem num acordo comum sobre o entendimento dos requisitos que serão analisados para o projeto.

Então, conforme sugestão do método Quicklocus, faz-se o questionamento para verificar se o Scrum atende a exigência desta prática, com a seguinte pergunta: *Os participantes do projeto se comprometem com os requisitos?* Para responder esta pergunta remete-se ao Scrum, e se percebe que neste método os participantes estão sempre presentes durante o desenvolvimento do projeto. Pelo fato da constante presença dos participantes pode-se concluir que esta prática está atendida no Scrum.

A Figura 34 apresenta os produtos de trabalho resultantes da prática específica REQM SP 1.2-2 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Avaliação dos impactos dos requisitos	-	-	X
2) Compromisso na documentação dos requisitos e suas mudanças	-	-	X

Fonte: primária

Figura 34 Produtos de trabalho da REQM SP1.2-2

A avaliação dos prováveis impactos que os requisitos terão sobre o projeto é realizada pelo *Product Owner*, e as documentações sobre o impacto e as mudanças são registradas entre as atividades realizadas durante a definição do *Product Backlog* e o *Sprint Backlog*. Com isso, tanto a avaliação dos impactos e o compromisso na documentação são Atendidos na prática específica REQM SP1.2-2 do modelo CMMI no Scrum.

3.2.3 Gerenciar as mudanças dos requisitos: SP 1.3-1

Todo processo de Gerenciamento de Requisitos no Scrum é controlado durante o *Sprint* que, ao final do desenvolvimento, no *Sprint Review Meeting*, resulta em um incremento do *Sprint* que é analisado pelos *stakeholders* no projeto, desde clientes até fornecedores. Caso novos requisitos apareçam, ou alguma mudança é solicitada, estes são acrescentados ao *Product Backlog*, que reiniciará o processo Scrum. Vale ressaltar, que um projeto Scrum encerra quando o desenvolvimento de todos os itens da lista das funcionalidades que estão presentes do *Product Backlog* são finalizados.

Diante disto, conforme sugestão do método Quicklocus, faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São gerenciadas as mudanças nos requisitos durante o projeto?* Para responder esta pergunta remete-se ao Scrum, e se percebe que neste método os requisitos não são gerenciados, com isso, esta prática não está atendida no Scrum.

A seguir, a Figura 35 apresenta os produtos de trabalho resultantes da prática específica REQM SP 1.3-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Status dos requisitos	-	-	X
2) Banco de Dados dos requisitos	X	-	-
3) Banco de Dados para Tomada de decisão dos requisitos	X	-	-

Fonte: primária

Figura 35 Produtos de trabalho da REQM SP1.3-1

Durante o *Sprint*, mantêm-se através da *Daily Scrum*, um controle *(status)* de qual requisito está sendo desenvolvido, o que satisfaz o item 1 desta prática específica. O Scrum não determina explicitamente que utiliza banco de dados dos requisitos, muito menos que o utiliza para auxílio nas tomadas de decisões. Portanto, os itens 2 e 3 da prática específica REQM SP 1.3-1 do modelo CMMI, não estão atendidos no Scrum.

3.2.4 Manter a rastreabilidade bidirecional dos requisitos: SP 1.4-2

Para o SCRUM a rastreabilidade não é importante, pois o que interessa é o resultado do produto final e não a técnica envolvida no desenvolvimento do produto. Os integrantes da equipe em suas reuniões diárias chegam a um consenso dos requisitos apresentados para seguir ou não o seu desenvolvimento. O Scrum não relaciona os requisitos aos *stakeholders*, nem menciona a relação dos requisitos entre si ou os requisitos aos modelos resultantes do projeto.

Faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *Existe rastreabilidade bidirecional entre os requisitos e os planos de projeto e os produtos de trabalho?* Para responder esta pergunta remete-se ao Scrum, e se percebe que neste método não existem controles de rastreabilidade, logo esta prática específica não está atendida no Scrum. A Figura 36

apresenta os produtos de trabalho resultantes da prática específica REQM SP 1.4-2 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Matriz da rastreabilidade dos requisitos	X	-	-
2) Sistema de acompanhamento dos requisitos	X	-	-

Fonte: primária

Figura 36 Produtos de trabalho da REQM SP1.4-2

O Scrum não possui uma matriz de rastreabilidade, conforme mostrado no item 1 da Figura 36, e não implementa um sistema de acompanhamento de requisitos, como o item 2 da Figura 36, apesar de ser indicado para ambientes em que os requisitos são poucos estáveis ou em muitas vezes desconhecidos. Então, a prática específica REQM SP 1.4-2 do modelo CMMI não é atendida no método Scrum.

3.2.5 Identificar inconsistências entre Projeto de Trabalho e Req. SP 1.5-1

Como o método Scrum baseia-se em princípios onde os requisitos são poucos estáveis e o projeto não está totalmente definido, as prováveis inconsistências são verificadas pela equipe Scrum somente durante o *Sprint*, através das reuniões diárias e do *Sprint Review*. O projeto somente chegará ao final quando todas as funcionalidades descritas no *Product Backlog* forem realizadas. As ações corretivas são realizadas no *Sprint Review* em consonância com o *Scrum Master* e o Cliente. Desta forma, o Scrum minimiza a existência de inconsistências entre o Projeto de Trabalho e os requisitos apresentados.

Então, conforme sugestão do método Quicklocus, faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *As inconsistências entre os planos de projeto, produtos de trabalho e requisitos são identificadas?* Para responder este questionamento remete-se ao Scrum, e se percebe que neste método são identificadas as inconsistências entre os planos de projeto e produtos de trabalhos durante o *Sprint Review*.

A seguir, a Figura 37 apresenta os produtos de trabalho resultantes da prática específica REQM SP 1.5-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Documentação das inconsistências: fontes, condições e lógica	X	-	-
2) Ações Corretivas	-	-	X

Fonte: primária

Figura 37 Produtos de trabalho da REQM SP1.5-1

O Scrum não determina explicitamente a geração de documentação das inconsistências do projeto, como indica o item 1 da Figura 37. Por outro lado, as ações corretivas, item 2 da Figura 37, são executadas após a entrega do incremento que é o resultado final do *Sprint,* também conhecido como *Product Increment.* Desta forma, esta prática específica REQM SP 1.5-1 do modelo CMMI está Atendida.

3.3 Análise da área de processo Desenvolvimento de Requisitos

A seguir, será apresentada uma análise do Scrum conforme as práticas específicas da área de processo Desenvolvimento de Requisitos (RD) da categoria engenharia do modelo CMMI.

3.3.1 Coletar as necessidades dos *stakeholders*: SP 1.1-1

No *Product Backlog* são colocadas todas as necessidades dos *stakeholders*, sendo o *Product Owner* o responsável pela criação e execução da lista no *Release Backlog* e *Sprint Backlog*. Conforme sugestão do método Quicklocus, faz-se então o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São identificadas e coletadas as necessidades dos stakeholders*? Para responder esta pergunta remete-se ao Scrum, e nota-se que neste método, através da geração do *Product Backlog* todas as necessidades dos *stakeholders* são identificadas e coletadas.

Assim, como todas as necessidades dos *stakeholders* no projeto são coletadas e identificadas, a prática específica RD SP 1.1-1 do modelo CMMI está totalmente atendida pelo Scrum. Cabe ressaltar que esta prática específica não define produtos de trabalhos e é aplicada somente na Representação Contínuo do modelo CMMI.

3.3.2 Eliciar as necessidades: SP 1.1-2

O SCRUM não define técnicas para eliciar os requisitos, como casos de uso, *brainstorming*, protótipos, JAD, etc. Faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *Utiliza-se de alguma técnica para elicitação das necessidades dos stakeholders?* Para tentar responder esta pergunta remete-se ao Scrum, e se percebe que neste método não são definidas técnicas para elicitação de requisitos.

Portanto, o único produto de trabalho resultante da prática específica RD SP 1.1-2 do modelo CMMI não é atendido no Scrum, conforme mostra a Figura 38.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Utilizar métodos para elicitação das necessidades, expectativas, restrições e interfaces externas	X	-	-

Fonte: primária

Figura 38 Produtos de trabalho da RD SP1.1-2

3.3.3 Desenvolver os Requisitos dos clientes: SP 1.2-1

O resultado do *Sprint Planning Meeting* conhecido como *Sprint Backlog* define e identifica o que será desenvolvido durante os 30 dias do *Sprint*.

Faz-se então o questionamento ao Scrum, conforme sugestão do Quicklocus, para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *Existe uma forma de transformação das necessidades dos stakeholders, expectativas, restrições e interfaces em requisitos dos clientes?* Para responder esta pergunta remete-se

ao Scrum, e se percebe que neste método o *Sprint Backlog* transforma todas as necessidades e expectativas dos *stakeholders* em requisitos de cliente.

A Figura 39 apresenta os produtos de trabalho resultantes da prática específica RD SP 1.2-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Requisitos dos clientes	-	-	X
2) Restrições dos clientes na condução da verificação	-	-	X
3) Restrições dos clientes na condução da validação	-	-	X

Fonte: primária

Figura 39 Produtos de trabalho da RD SP1.2-1

Todos os requisitos do cliente, item 1 da Figura 39, são acompanhados durante o desenvolvimento pelo *Product Owner,* pois é sua função determinar se os requisitos estão sendo desenvolvidos de acordo com o solicitado pelo cliente, comparando-os ainda com a lista das funcionalidades presente no *Product Backlog*. Restrições dos clientes na condução de verificação ou validação, conforme itens 2 e 3 da Figura 39, são acompanhadas pelo *Product Owner* e o *Scrum Master* respectivamente.

Desta forma, a prática específica RD SP 1.2-1 do modelo CMMI é atendida no Scrum.

3.3.4 Estabelecer os requisitos dos produtos e seus componentes: SP 2.1-1

O estabelecimento dos requisitos dos produtos e dos componentes é realizado no Scrum através do *Product Backlog*, onde são detalhadas todas as funcionalidades, características, infra-estrutura, arquitetura e tecnologia que o produto deverá possuir.

Diante do exposto, conforme sugestão do método Quicklocus, faz-se então o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática

com a seguinte pergunta: *Estabelece e mantém os requisitos dos produtos e dos componentes dos produtos, os quais serão baseados nos requisitos dos clientes?* Para responder esta pergunta remete-se ao Scrum, e se percebe que neste método o estabelecimento dos requisitos dos produtos e dos componentes são realizados na geração do *Product Backlog*.

A seguir, a Figura 40 apresenta os produtos de trabalho resultantes da prática específica RD SP 2.1-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Requisitos derivados (custos, performance, etc.)	-	-	X
2) Requisitos dos produtos	-	-	X
3) Requisitos dos componentes dos produtos	-	-	X

Fonte: primária

Figura 40 Produtos de trabalho da RD SP2.1-1

Schwaber (2002, p. 63), aponta a existência da definição de custos e orçamentos no projeto que será desenvolvido no SCRUM, indicando com isso, que a prática específica RD SP 2.1-1 do modelo CMMI é atendida no Scrum.

3.3.5 Alocar os requisitos dos componentes dos produtos: SP 2.2-1

Após a definição da lista das funcionalidades do produto que estão no *Sprint Backlog List* e que serão desenvolvidas durante o *Sprint*, a equipe Scrum, o *Scrum Master* e o *Product Owner* atuam na transformação das necessidades dos *stakeholders* em requisitos de produto, decidindo como alocar ou distribuir os requisitos para os componentes de produto. Faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *Os requisitos são alocados para cada componente dos produtos?* Percebe-se que neste método não são detalhadas as alocações dos requisitos para os componentes dos produtos.

A seguir, a Figura 41 apresenta os produtos de trabalho resultantes da prática específica RD SP 2.2-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Planilhas com alocação dos requisitos	X	-	-
2) Alocação dos requisitos temporários	X	-	-
3) Projeto das restrições	X	-	-
4) Requisitos derivados	X	-	-
5) Relacionamentos entre os requisitos derivados	X	-	-

Fonte: primária

Figura 41 Produtos de trabalho da RD SP2.2-1

A alocação dos requisitos aos componentes de produto não é um processo detalhado durante o *Sprint*, pois o SCRUM não define técnicas durante esta fase. Portanto, todos os itens da Figura 41 podem estar presentes, mas não estão especificados no Scrum. Diante disto, a prática específica RD SP 2.2-1 do modelo CMMI não é atendida no Scrum.

3.3.6 Identificar os requisitos de interfaces: SP 2.3-1

No Scrum, os requisitos de interface com outros sistemas não são detalhados, seja durante a elaboração do *Product Backlog* ou na execução do *Sprint*, descrevendo superficialmente a ligação com sistemas.

Diante do exposto, conforme sugestão do método Quicklocus, faz-se então o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São elaborados os requisitos de interface?* Para buscar esta resposta remete-se ao Scrum, e nota-se que neste método os requisitos de interface não são detalhados.

A Figura 42 apresenta o único produto de trabalho da prática específica RD SP 2.3-1, do modelo CMMI, demonstrando que não é atendido no Scrum.

Produto de Trabalho Resultante	Não atendido	Parcialmente atendido	Atendido
1) Elaboração dos Requisitos de Interface	X	-	-

Fonte: primária

Figura 42 Produto de trabalho da RD SP 2.3-1

3.3.7 Estabelecer conceitos operacionais e cenários: SP 3.1-1

Com a criação do *Product Backlog*, que contém todas as necessidades do negócio e os requisitos (funcionais e não funcionais) que servirão de base para o desenvolvimento do software, espera-se descrever toda a seqüência de eventos que poderão ocorrer com o uso deste sistema, gerando, de certa forma um cenário para o desenvolvimento do software. Conforme sugestão do método Quicklocus, faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São utilizados os cenários?* Neste método são utilizados os cenários.

A Figura 43 apresenta os produtos de trabalho resultantes da prática específica RD SP 3.1-1 do modelo CMMI.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Conceitos operacionais	-	-	X
2) Instalação de produto, operação, manutenção e conceitos de suporte	-	-	X
3) Disposição dos conceitos utilizados	-	X	-
4) Construção de Cenários	-	X	-
5) Casos de Uso	X	-	-
6) Novos requisitos	-	-	X

Fonte: primária

Figura 43 Produtos de trabalho da RD SP3.1-1

Com o *Product Backlog*, os itens 1 e 2 da Figura 43, estão Atendidos pelo Scrum, e o *Scrum master* coloca a disposição dos *stakeholders* todos os conceitos utilizados durante o projeto, e, assim, o item 3 da Figura 43 é considerado parcialmente atendido pelo Scrum.

Como este método não detalha a construção de cenários, mas descreve a seqüência de eventos deste sistema, o item 4 é parcialmente atendido pelo Scrum. Pelo fato do Scrum não definir técnicas para eliciar os requisitos, como casos de uso, o item 5 da Figura 43 não

é atendido pelo Scrum. Por outro lado, novos requisitos, podem ser inseridos após a *Sprint Review Meeting*, pois o *Scrum Master* reporta os resultados do trabalho com os participantes para avaliação, e caso surja um novo requisito é neste momento que ele é inserido, assim, o item 6 da Figura 43 é Atendido no Scrum. Desta forma, a prática específica RD SP3.1-1 do modelo CMMI, é atendida no Scrum.

3.3.8 Estabelecer uma definição das funcionalidades requeridas: SP 3.2-1

A criação de uma lista das funcionalidades, denominada *Product Backlog*, é o ponto inicial no Scrum. Além destas funcionalidades, nesta lista, são inseridas as características, padrões, tecnologias e estratégias do produto que será desenvolvido. Schwaber (2002, p.33) considera que o *Product Backlog* e o *Sprint* são considerados as fases mais importantes do Scrum.

Diante do exposto, conforme sugestão do método Quicklocus, faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São desenvolvidos diagramas de atividades e casos de uso?* Para responder esta pergunta remete-se ao Scrum, e se percebe que neste método não são desenvolvidos os diagramas de atividades tampouco os casos de uso.

A seguir, a Figura 44 apresenta os produtos de trabalho resultantes da prática específica RD SP 3.2-1 do modelo CMMI.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Arquitetura funcional	-	-	X
2) Diagrama de Atividades e casos de uso	X	-	-
3) Análise Orientada a Objeto com identificação de serviços	X	-	-

Fonte: primária

Figura 44 Produtos de trabalho da RD SP3.2-1

A arquitetura funcional apontada no item 1 da Figura 44, é indicada na criação do *Product Backlog*. Como o Scrum não determina a técnica que será adotada para especificar

a funcionalidade requerida pelo cliente, fica a critério do desenvolvedor a utilização de casos de uso e diagramas de atividades, bem como a identificação de serviços na análise orientada a objeto. Em resumo, a prática específica RD SP 3.2-1 do modelo CMMI não é atendida no Scrum.

3.3.9 Analisar os requisitos: SP 3.3-1

No Scrum, o processo de análise dos requisitos procura identificar se todas as funcionalidades do sistema descritas no *Sprint Backlog List*, resultado da análise do *Product Backlog*, serão resolvidos com os requisitos descritos e se os mesmos se sobrepõem ou estão em conflito, não definindo explicitamente um documento de requisitos acordados entre os *stakeholders*.

Conforme sugestão do método Quicklocus, faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São gerados relatórios dos defeitos dos requisitos, e se propõe mudança nos requisitos para resolver estes defeitos?* Neste método não são gerados relatórios de defeitos dos requisitos.

A seguir, a Figura 45 apresenta os produtos de trabalho resultantes da prática específica RD SP 3.3-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produtos de Trabalho Resultantes	Não atendido	Parcialmente atendido	Atendido
1) Relatórios dos defeitos dos requisitos	X	-	-
2) Propor mudanças nos requisitos para resolver defeitos	-	-	X
3) Apresentar os requisitos chaves	-	-	X
4) Elaborar medidas técnicas de performance	X	-	-

Fonte: primária

Figura 45 Produtos de trabalho da RD SP3.3-1

Percebe-se que a análise dos requisitos está presente no projeto SCRUM, porém, como não apresenta relatórios de defeitos como sugere o item 1 da Figura 45, este produto de trabalho não é atendido pelo Scrum. Propor mudanças nos requisitos para resolução de

defeitos, item 2 da Figura 45, é uma das principais características do Scrum como todo método ágil.

Apresentar os requisitos chaves, item 3 da Figura 45, é Atendido no Scrum porque neste método existe a preocupação no desenvolvimento dos requisitos de acordo com as necessidades dos *stakeholders*. Para que isto ocorra, é necessário elencar os requisitos chaves, ou fundamentais para execução do projeto.

Os requisitos analisados desde a elaboração das funcionalidades do software até o seu desenvolvimento, não detalham prováveis medidas técnicas de performance que poderão ser utilizadas no projeto.

3.3.10 Analisar os requisitos para avaliação: SP 3.4-3

O Scrum não define modelos, simuladores ou protótipos para analisar o risco das necessidades dos *stakeholders* e suas restrições. Então, conforme sugestão do método Quicklocus, faz-se o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São avaliados os riscos relatados nos requisitos?* No Scrum não são avaliados os riscos relatados nos requisitos.

A seguir, a Figura 46 apresenta o único produto de trabalho resultante da prática específica RD SP 3.4-3 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produto de Trabalho Resultante	Não atendido	Parcialmente atendido	Atendido
1) Avaliar os riscos relatados nos requisitos	X	-	-

Fonte: primária

Figura 46 Produto de trabalho da RD SP3.4-3

Avaliar os riscos relatados nos requisitos é produto de trabalho resultante desta prática, porém, conforme demonstra o único item da Figura 46, o Scrum não avalia os riscos, pois entende que os prováveis riscos são eliminados rapidamente através das

reuniões diárias e na entrega rápida do sistema. Desta forma, a prática específica RD SP 3.4-3 do modelo CMMI não é atendida no Scrum.

3.3.11 Validar os requisitos: SP 3.5-1

A validação dos requisitos ocorre na fase conhecida como *Post Sprint Demonstration and Meeting*, onde os requisitos são verificados se está descrito de forma apropriada, procurando, com isso, eliminar problemas como requisitos incompletos, ambíguos ou inconsistentes. Ressalta-se, que esta prática específica é somente aplicada na Representação Contínuo do modelo CMMI.

Diante do exposto, conforme sugestão do método Quicklocus, faz-se então o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São analisados e validados os requisitos para determinar o risco que poderá resultar caso o produto não esteja de acordo com o esperado pelo ambiente?* O risco que poderá resultar caso o produto não esteja de acordo com o projeto é analisado e validado durante a fase denominada de *Post Sprint Demonstration and Meeting*.

A seguir, a Figura 47 apresenta a lista do produto de trabalho resultante da prática específica RD SP 3.5-1 do modelo CMMI, com a determinação da escala de classificação do atendimento ao Scrum.

Produto de Trabalho Resultante	Não atendido	Parcialmente atendido	Atendido
1) Resultados da validação dos requisitos	-	-	X

Fonte: primária

Figura 47 Produto de trabalho da RD SP3.5-1

Os resultados da validação dos requisitos são expostos para que a equipe Scrum inicie o processo de desenvolvimento. Desta forma, o resultado de trabalho da prática específica RD SP 3.5-1 do modelo CMMI é Atendido no Scrum.

3.3.12 Validar os requisitos com métodos válidos: SP 3.5-2

A reunião conhecida como *Post Sprint Demonstration and Meeting* é um método que o Scrum utiliza para validar os requisitos. Esta prática específica diferencia-se da prática anterior, somente por exigir a definição de um método válido para validação dos requisitos.

Assim, conforme sugestão do método Quicklocus, faz-se então o questionamento ao Scrum para verificar se este método ágil atende a exigência desta prática com a seguinte pergunta: *São validados os requisitos para assegurar que o resultado esperado pelo produto seja de acordo com o esperado?* Como ocorreu na análise da prática específica RD SP 3.5-1 do modelo CMMI, os requisitos são validados garantindo que o produto gerado esteja de acordo com o esperado.

A Figura 48 apresenta o produto de trabalho resultante da prática específica RD SP 3.5-2 do modelo CMMI, com a determinação da escala de classificação do atendimento ao SCRUM, demonstrando que esta prática é atendida pelo Scrum.

Produto de Trabalho Resultante	Não atendido	Parcialmente atendido	Atendido
1) Registro dos métodos de análise e seus resultados	-	-	X

Fonte: primária

Figura 48 Produto de trabalho da RD SP3.5-2

3.4 Considerações finais

A Figura 49 apresenta um resumo da análise realizada individualmente em cada prática específica em relação à determinação da escala de classificação do atendimento ao Scrum. Observa-se, que somente foram analisadas as práticas das áreas de processo Gerenciamento de Requisitos e Desenvolvimento de Requisitos do modelo CMMI.

Código	Descrição da Prática específica	Não atendido	Parcialmente atendido	Atendido
GERENCIAMENTO DE REQUISITOS				
SP 1.1-1	Obter o entendimento dos requisitos			X
SP 1.2-2	Obter o compromisso para os requisitos			X
SP 1.3-1	Gerenciar as mudanças dos Requisitos		X	
SP 1.4-2	Manter a rastreabilidade bidirecional dos requisitos	X		
SP 1.5-1	Identificar inconsistências entre o Projeto de Trabalho e os Requisitos		X	
DESENVOLVIMENTO DE REQUISITOS				
SP 1.1-1	Coletar as necessidades dos *stakeholders*			X
SP 1.1-2	Eliciar as necessidades	X		
SP 1.2-1	Desenvolver os Requisitos dos clientes			X
SP 2.1-1	Estabelecer os requisitos dos produtos e dos componentes dos produtos			X
SP 2.2-1	Alocar os requisitos dos componentes dos produtos	X		
SP 2.3-1	Identificar os requisitos de interface	X		
SP 3.1-1	Estabelecer conceitos operacionais e cenários			X
SP 3.2-1	Estabelecer uma definição das funcionalidades requeridas		X	
SP 3.3-1	Analisar os requisitos			X
SP 3.4-3	Analisar os requisitos para avaliação	X		
SP 3.5-1	Validar os requisitos			X
SP 3.5-2	Validar os requisitos com métodos válidos			X

Fonte: primária

Figura 49 Resultado final da avaliação

Percebe-se que 47% das práticas específicas avaliadas encontram-se na posição de não atendido ou parcialmente atendido, ou seja, quase metade destas práticas necessitam de um estudo mais detalhado para adequação ao modelo CMMI. Dentre as práticas na situação de não atendido ou parcialmente atendido, a maioria, cerca de 62%, encontra-se como não atendido demonstrando que uma atenção especial deverá ser atribuída para estas práticas.

Conclui-se, que o método ágil Scrum requer que sejam agregadas novas diretrizes, ou ações, que quando estabelecidas, satisfaçam as necessidades impostas pelas práticas

específicas das áreas de processo Gerenciamento de Requisitos (REQM) e Desenvolvimento de Requisitos (RD) do modelo CMMI.

4 Proposta de Extensão do método Scrum para adequação ao modelo CMMI nas áreas de processo REQM e RD

Este capítulo apresenta uma proposta de adequação do método Scrum ao modelo CMMI. Denominou-se esta proposta como: xScrum. A letra “x” foi agregada ao nome inicial do método, representando a palavra “extensão”.

As extensões serão realizadas somente onde o método Scrum, após análise realizada no capítulo 3, obteve o conceito de *não atendido* ou *parcialmente atendido* conforme exigências das práticas específicas das áreas de processo Gerenciamento de Requisitos (REQM) e Desenvolvimento de Requisitos (RD) do modelo CMMI. Cabe ressaltar, por delimitação do trabalho, que as demais áreas de processo do modelo CMMI não serão abordadas por esta proposta.

Para cada problema identificado na análise realizada, será proposta aqui uma solução. As soluções propostas utilizaram como base os seguintes fundamentos:

- Atender às práticas – Todas as práticas das áreas de processo REQM e RD do modelo CMMI deverão ser atendidas;
- Estar alinhada à “cultura” Scrum – As soluções deverão estar de acordo com a filosofia Scrum, de forma a não descaracterizá-la e a realizar o menor impacto possível no método;
- Utilizar práticas da engenharia de software que possam solucionar os problemas encontrados, como por exemplo a documentação de requisitos.

A seguir, uma relação das práticas específicas não atendidas ou parcialmente

atendidas pelo método Scrum, separadas pelas áreas de processo REQM e RD.

a) Gerenciamento de Requisitos.
    1. Gerenciar as mudanças dos Requisitos: SP 1.3-1
    2. Identificar inconsistências entre o Projeto de Trabalho e os Requisitos: SP 1.5-1
    3. Manter a rastreabilidade bidirecional dos requisitos: SP 1.4-2
b) Desenvolvimento de Requisitos.
    1. Alocar os requisitos dos componentes dos produtos: SP 2.2-1
    2. Identificar os requisitos de interface: SP 2.3-1
    3. Eliciar as necessidades: SP 1.1-2
    4. Analisar os requisitos para avaliação: SP 3.4-3
    5. Estabelecer uma definição das funcionalidades requeridas: SP 3.2-1

Então, são propostas diretrizes no método Scrum que, quando executadas, satisfaçam as exigências do modelo CMMI nas práticas específicas destacadas anteriormente. Vale lembrar que alguns documentos utilizados na presente proposta são adaptações de Gampert (2003)[27]. A seguir, faz-se um detalhamento da proposta, através da descrição das diretrizes que deverão ser executadas.

4.1 Diretriz 1: Gerenciar as mudanças dos requisitos.

Para gerenciar as mudanças dos requisitos, a proposta é o desenvolvimento de um documento que deverá atender basicamente as seguintes necessidades:

a) registro de novos requisitos ou alterações: as solicitações de novos requisitos ou alterações serão recebidas formalmente.
b) elaboração de relatórios de impacto: documentando a inclusão ou alteração,

[27] GAMPERT, Gilberto. *Gerenciamento de Requisitos na Seção de Informática da UPF*. Monografia (Graduação em Ciência da Computação) – Curso de Ciência da Computação, Universidade de Passo Fundo, 2003

mantendo um histórico de alteração de cada requisito, inclusive de seus atributos.

A seguir, uma descrição de seu conteúdo, integração com o Scrum, responsável pelo seu preenchimento e quem efetivamente poderá consultá-lo:

Responsável pela manutenção: A pessoa responsável por manter este documento será o Scrum Master.

Consultas: A consulta poderá ser executada por todos membros da equipe envolvidos diretamente no desenvolvimento dos requisitos

Integração com o Scrum: Durante o processo do Scrum, este documento será utilizado durante o *Sprint*, logo após a confecção do *Product Backlog*.

Conteúdo: Seu conteúdo tem como objetivo principal gerenciar as mudanças dos requisitos. A principal função é armazenar todas as alterações realizadas em um requisito. O documento é divido em três partes: “A” - Histórico de Revisões, “B” - Descrição dos Requisitos e por fim “C” – Impacto nos Requisitos.

O item “A” representa um histórico de revisões, tendo como objetivo guardar dados como data, nome do autor e tipo de operação realizada, ou seja, uma inclusão, alteração ou exclusão no documento. Registra também a versão da revisão. O item “B” é a principal parte do documento, pois tem como função armazenar os dados específicos de cada requisito, como por exemplo, tipo, origem, versão, etc. A seguir uma descrição dos campos do documento:

a) ID: identificação do projeto;
b) DATA: data da identificação do requisito;
c) PROJETO: código do projeto em que está em desenvolvimento;
d) NOME: descrição sucinta do requisito;
e) ORIGEM: descrição da origem dos requisitos (quem foi que o identificou);

f) TIPO_REQ: descrição se o requisito é do tipo funcional ou não funcional;
g) DESCRIÇÃO: local para esboço de forma objetiva das características do requisito;
h) SOLUÇÃO: descrição de uma provável solução para o requisito em questão;
i) PRIORIDADE: o requisito está dividido em três categorias para a preferência no em seu desenvolvimento: a) alto, o requisito tem alta prioridade para seu desenvolvimento; b) médio, o requisito tem média prioridade para seu desenvolvimento; c) baixo, o requisito tem pouca prioridade para seu desenvolvimento;
j) TIPO: representa a quem se destina o requisito: ao cliente, ao produto ou a interface;
k) RISCO: representa a gravidade que o requisito representa para o bom andamento do projeto, e está dividido em três partes: a) alto, b) médio e c) baixo;
l) VERSAO DESTINO: identifica em que versão do projeto o requisito será desenvolvido;
m) DEPENDÊNCIA: descrição dos requisitos que dependem deste requisito.

Finalmente, o item “C” tem como objetivo, armazenar, através de uma breve descrição, que impactos o requisito poderá exercer no sistema. Além disto, registra qual pessoa será responsável por esta implementação. Vale ressaltar que todo seu conteúdo foi projetado visando atender principalmente as exigências do CMMI. A Figura 50 apresenta o documento, intitulado Descrição de Requisitos.

DESCRIÇÃO DE REQUISITOS

A) HISTÓRICO DE REVISÕES		
Data	**Revisão**	**Autor**
DD/MM/AAAA		
Movimento:		I / A / E

B) DESCRIÇÃO DE REQUISITOS	
ID:	
DATA:	DD/MM/AAAA
PROJETO:	
NOME:	
ORIGEM:	
TIPO_REQ:	( ) Funcional ( ) Não funcional
DESCRIÇÃO:	
SOLUÇÃO:	
PRIORIDADE:	( ) Alto ( ) Médio ( ) Baixo
TIPO:	( ) Cliente ( ) Produto ( ) Interface
RISCO:	( ) Alto ( ) Médio ( ) Baixo
VERSAO DESTINO:	
DEPENDÊNCIA:	

c) IMPACTO nos REQUISITOS	
NOME	ALTERAÇÃO
	I / A / E
RESPONSÁVEL:	

Fonte: Adaptado de Gampert (2003)

Figura 50 Gerenciar as mudanças dos requisitos

Com o desenvolvimento deste documento, espera-se atender as exigências da prática específica REQM SP 1.3-1 do modelo CMMI.

4.2 Diretriz 2 : Identificar inconsistências entre o Projeto de Trabalho e os Requisitos.

A proposta é a criação de um documento, conforme Figura 51, durante o *Sprint* do Scrum, que indique as inconsistências entre o projeto de trabalho e os requisitos alocados. Este documento será avaliado durante o *Sprint Review Meeting*, que é a prática responsável pela apresentação dos incrementos ao cliente. Também auxiliará na detecção de desvios, na visualização de inconsistências e identificação de pontos do projeto fora do escopo de desenvolvimento.

A seguir, uma descrição do documento com relação ao seu responsável pelo seu

preenchimento, consultas, integração com o Scrum e, finalmente, seu conteúdo.

Responsável pela manutenção: A pessoa responsável por manter este documento será o Scrum Master.

Consultas: A consulta deste documento poderá ser executada por todos membros da equipe envolvidos diretamente no desenvolvimento dos requisitos.

Integração com o Scrum: Este documento será preenchido durante o *Sprint*, e seu conteúdo será avaliado durante o *Sprint Review Meeting*.

Conteúdo: Seu conteúdo tem como objetivo principal armazenar as inconsistências entre o projeto de trabalho e os requisitos alocados. É composto basicamente por uma data, uma identificação do requisito e uma breve descrição da inconsistência encontrada em relação aos requisitos para cada componente.

A Figura 51 apresenta o documento, intitulado *Component Product Backlog*.

COMPONENT PRODUCT BACKLOG

INCONSISTENCIA	
DATA:	DD/MM/AAAA
ID REQUISITO	**INCONSISTÊNCIA**

Fonte: primária

Figura 51 Identificar inconsistências

Este documento possui este formato para identificação das inconsistências entre o projeto de trabalho conforme exigência sugerida pela prática específica REQM SP 1.5-1 do modelo CMMI.

4.3 Diretriz 3: Desenvolver uma matriz de rastreabilidade.

O uso de matriz de rastreabilidade representa os dados que permitem rastrear como

os requisitos se relacionam: relacionamento dos requisitos entre si, com *stakeholders* e com os módulos do projeto, conforme apresenta Sommerville (2003, p.120).

A proposta é incluir no método Scrum uma matriz de rastreabilidade, conforme Figura 52, com a utilização da mesma estrutura utilizada para gerenciar as mudanças dos requisitos, conforme exigência da REQM SP 1.3-1.

A seguir, uma descrição do documento com relação ao responsável pelo seu preenchimento, consultas, integração com o Scrum e, finalmente, seu conteúdo.

Responsável pela manutenção: A pessoa responsável por manter este documento será o Scrum Master.

Consultas: Poderá ser executada por todos os membros da equipe envolvidos diretamente no desenvolvimento dos requisitos.

Integração com o Scrum: Este documento será utilizado durante o *Sprint*.

Conteúdo: Seu conteúdo tem como objetivo principal a obtenção da informação de rastreabilidade entre os requisitos e os artefatos do projeto durante um sprint. Este dado é de fundamental importância para poder avaliar o impacto de uma mudança. Também é necessária para agilizar a estimativa de quantidade de esforço e custo para modificar os artefatos do projeto de forma a acomodá-los em relação à mudança requisitada. O documento é divido em duas partes: a primeira contém o “Histórico de Revisões” e a segunda a “Rastreabilidade dos Requisitos”.

A primeira parte representa os dados gerais da matriz, e tem como objetivo armazenar a data, código do projeto em desenvolvimento e código do Sprint que será rastreado os requisitos. A segunda parte é a principal parte do documento, pois tem como função registrar a matriz em si, ou seja, para cada requisito ou componente é atribuído se este tipo é: I (inconsistente), D (dependente) ou A (alocado). A letra “I” representa a inconsistência entre o componente e o requisito. A letra “D” representa a dependência de

um requisito com outro requisito, e, por fim, a letra “A” representa a alocação de componente com componente. A Figura 52, apresenta o documento, intitulado Matriz de Rastreabilidade.

MATRIZ DE RASTREABILIDADE

DATA:	DD/MM/AAAA							
PROJETO:	Código do projeto							
SPRINT:	Código do Sprint							
Tipo: Requisito/ Componente	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1								
2								
3		D						
4	I							
5			A					
6								
7								
8								

Fonte: Adpatado de Sommerville (2003)

Legenda: I: Inconsistente / D: dependente / A: alocado
Onde: I: Componente com Requisito
D: Requisito e Requisito
A: Componente com Componente

Figura 52 Matriz de rastreabilidade

Vale ressaltar que poderá ser utilizada uma tabela diferente para cada *Sprint*, evitando, com isso, que ela tenha um tamanho relativamente grande, prejudicando seu manuseio pela equipe durante o desenvolvimento dos trabalhos. Caso necessário, a tabela pode referenciar requisitos, e também outros artefatos, definidos em outra tabela, ou seja, que foram definidos em outro *Sprint*. Se poucos requisitos estiverem sendo definidos em um *Sprint* e eles forem bastante relacionados com os requisitos definidos em outro *Sprint*, os requisitos deste novo *Sprint* podem ser adicionados à tabela de requisitos do *Sprint* já realizado.

Com isso, ter-se-á uma matriz de rastreabilidade dos requisitos e um sistema de acompanhamento dos requisitos, atendendo também as necessidades da prática específica REQM SP1.4-2 do modelo CMMI.

4.4 Diretriz 4: Alocar os requisitos aos componentes dos produtos.

Um projeto de software é composto por uma descrição da estrutura de software a ser implementada, dos dados que farão parte do sistema, das interfaces e também dos algoritmos a serem utilizados.

A proposta é a utilização de um documento para registrar os componentes do produto e os requisitos alocados a eles. Cada projeto de trabalho terá um documento. Esta implementação será executada através da criação do documento *Alocar Requisitos*, conforme demonstra a Figura 53. Este documento armazenará, também, os requisitos temporários, as derivações dos requisitos e seus relacionamentos.

A seguir, uma descrição do documento com relação ao responsável pelo seu preenchimento, consultas, integração com o Scrum e, finalmente, seu conteúdo.

Responsável pela manutenção: A pessoa responsável por manter este documento será o *Scrum Master*.

Consultas: A consulta poderá ser executada por todos os membros da equipe envolvidos diretamente no desenvolvimento dos requisitos.

Integração com o Scrum: Este documento será utilizado durante o *Sprint Review Meeting*.

Conteúdo: Para cada projeto de trabalho será gerado um documento que armazenará os requisitos temporários, as derivações destes requisitos e seus prováveis relacionamentos. O documento é divido em duas partes: “A” - Dados gerais da alocação e “B” – Requisitos alocados. O item “A” representa os dados gerais do documento, tendo como objetivo principal armazenar a data, o código do projeto ou do componente e uma descrição sucinta do componente. O item “B” tem como função armazenar os requisitos alocados. A seguir uma descrição dos campos do documento do item “B”:

a) ID: identificação do requisito que será alocado;
b) DESCRIÇÃO: breve descrição dos requisitos;
c) DERIVAÇÕES: descreve as prováveis dependências entre os requisitos;
d) TEMPORÁRIO: indica se o requisito é passageiro, ou seja, não é definitivo;
e) RELAC: indica o relacionamento entre os requisitos.

A Figura 53 apresenta o documento, intitulado Alocar Requisitos.

ALOCAR REQUISITOS

ALOCAÇÃO				
DATA:	DD/MM/AAAA			
PROJETO DE TRAB / COMPONENTE				
NOME:				
REQUISITOS ALOCADOS				
ID	DESCRIÇÃO	DERIVAÇÕES	TEMPORÁRIO	RELAC.
ID			S / N	

Fonte: primária

Figura 53 Alocar requisitos

Com o desenvolvimento deste documento, espera-se atender as exigências da prática específica RD SP 2.2-1 do modelo CMMI.

4.5 Diretriz 5: Identificar as interfaces dos requisitos.

A proposta é a elaboração do documento de requisitos de software, descrevendo as restrições quanto às características do produto, a interface com outras aplicações, a descrição sob o domínio e as informações de suporte ao conhecimento do problema. É necessário que este documento seja escrito em linguagem de alto nível, pois ele servirá de uma espécie de “contrato” entre os usuários e os desenvolvedores.

A seguir, uma descrição do documento com relação ao responsável pelo seu preenchimento, consultas, integração com o Scrum e, finalmente, seu conteúdo.

Responsável pela manutenção: A pessoa responsável por manter este documento será o cliente e o *Scrum Master*.

Consultas: A consulta poderá ser executada por todos os membros da equipe envolvidos diretamente no desenvolvimento dos requisitos.

Integração com o Scrum: Este documento será utilizado durante a elaboração do *Product Backlog*.

Conteúdo: Deve identificar o produto, fornecer uma descrição geral do que ele será e do que não será, descrever o uso pretendido, incluindo os benefícios e objetivos. Também devem fazer parte dele as referências a outros documentos. (novamente, não explicou cada campo do documento.) O documento é divido em duas partes: “A” - Dados gerais do documento e “B” – Documento de Visão.

O item “A” representa um histórico de revisões, tendo como objetivo guardar dados como data, nome do autor, além de registrar a versão da revisão. O item “B” é a principal parte do documento, pois tem como função realizar a descrição do documento de visão A seguir uma descrição dos campos do documento do item “B”:

a) DATA: data do documento;
b) PROJETO: código do projeto;
c) INTRODUÇÃO: deve identificar o produto, fornecer uma descrição geral do que ele será e do que não será. Descrever o uso pretendido incluíndo os benefícios e objetivos. Também devem fazer parte as referências a outros documentos;
d) VISÃO DO PRODUTO: (Domínio, restrições e interfaces) deve fornecer uma visão de alto nível das capacidades do produto, das interfaces com outras aplicações e das configurações do sistema. Deve também, descrever restrições quanto às características do produto e seu domínio;
e) RESPONSÁVEL: descrição do nome do responsável pelo preenchimento do documento.

A Figura 54 apresenta o documento, intitulado Documento de Visão.

DOCUMENTO DE VISÃO

HISTÓRICO DE REVISÕES		
Data	**Revisão**	**Autor**
DD/MM/AAAA	Versão Inicial	Nome do autor
DOCUMENTO DE VISÃO		
DATA:	DD/MM/AAAA	
PROJETO:		
INTRODUÇÃO:		
VISÃO DO PRODUTO: (Domínio, restrições e interfaces)		
RESPONSÁVEL:		

Fonte: Adaptado de Gampert (2003)

Figura 54 Documento de Visão

Com o desenvolvimento deste documento, espera-se atender as exigências da prática específica RD SP 2.3-1 do modelo CMMI.

4.6 Diretriz 6: Definir uma técnica para eliciar os requisitos.

A proposta para eliciar os requisitos é a utilização da técnica de dinâmica de grupo JAD *(Joint Application Development).* Justifica-se a escolha desta técnica porque é fortemente baseada em reuniões, como ocorre no Scrum e também porque outro método ágil, o ASD, já a utiliza com sucesso, conforme destaca Highsmith (2003). A idéia básica aqui é "aproveitar" o que os outros métodos ágeis possuem.

O JAD teve origem nos laboratórios de Software da IBM, no Canadá, no final dos anos 60. Segundo Raghavan *et al.* (1994)[28] esta técnica se baseia em quatro princípios: a) dinâmica de grupo; b) utilizar os recursos visuais para facilitar a comunicação. c) manter um processo organizado e racional; e, por fim, d) filosofia de documentação WYSIWYG *(What you see is what you get).* Ainda segundo o autor, é uma técnica baseada em reuniões e está dividida em três fases: Adaptação, Celebração e Conclusão.

A fase da Adaptação tem como objetivo principal definir, em alto nível, o que será projetado, para isto, busca dados referentes à empresa, seus clientes e seus usuários. A fase

[28] Raghavan, S. G Zelesnik *et al.* Lecture notes on Requirements Elicitation. Educational Material. CMU/SEI 94-EM-10. Software Engineering Institute. http://www.sei.cmu.edu.

da Celebração tem como função determinar como será executada a reunião propriamente dita, ou seja, realizar a apresentação, definir quais são os objetivos da reunião, documentar os resultados finais e, por fim, encerrar a reunião. E, finalmente, a fase da Conclusão objetiva transformar o que foi relatado na fase da Celebração em um documento por escrito.

Para realização destas três fases o JAD possui os seguintes papéis a seguir destacados:

a) Chefe JAD: É o responsável por todo o processo durante as reuniões. Deve possuir habilidades como comunicação e liderança.
b) Analista: Responsável pela geração dos documentos que são gerados durante a reunião. Deve ser um bom redator porque necessita colocar tudo o que o foi tratado na reunião num documento final.
c) Patrocinador Executivo: Responsável por decidir se o projeto será ou não desenvolvido. Também deve oferecer todas as condições para o sucesso do projeto.
d) Representantes dos usuários: Pessoas que serão os futuros clientes do projeto que está sendo desenvolvido.
e) Representantes do negócio: São especialistas sobre o tema que está sendo estudado, possuem um alto conhecimento de toda organização.

Então, esta técnica possibilita a todos os *stakeholders* uma visão global do sistema, consolidando interesses dos diversos usuários quanto ao software que será implementado.

Como o Scrum possui papéis distintos do JAD a proposta é que os papéis presentes no Scrum, como *Scrum Master, Product Owner*, Cliente e Equipe, desempenhem as funções sugeridas pelo JAD. Com isso, ter-se-á a seguinte configuração: o *Scrum master* fará o papel do Chefe JAD. O *Product Owner* será o patrocinador Executivo. O Analista, o Representante dos Usuários e Representante dos Negócios serão representados pelo Cliente

e a Equipe Scrum.

O xScrum manterá as três fases propostas pelo JAD, a adaptação, a celebração e a conclusão. Estas fases serão aplicadas desde a fase de levantamento de dados até a implantação do sistema (software). O produto final destas três fases auxiliará na construção do *Product Backlog*, considerado um dos principais produtos do Scrum.

Vale ressaltar que o método Scrum utiliza também a reunião com o usuário para definição do domínio da aplicação e a identificação de todas as partes interessadas do projeto. Com a utilização do JAD espera-se atender a prática específica RD SP 1.1-2 do modelo CMMI.

4.7 Diretriz 7: Analisar os riscos.

Como o método SCRUM não define modelos, simuladores ou protótipos para analisar o risco das necessidades dos *stakeholders* e suas restrições, a proposta é o desenvolvimento de um documento que contemple esta necessidade.

A seguir, uma descrição do documento com relação ao responsável pelo seu preenchimento, consultas, integração com o Scrum e, finalmente, seu conteúdo.

Responsável pela manutenção: A pessoa responsável por manter este documento será o cliente e o *Scrum Master*.

Consultas: A consulta poderá ser executada por todos os membros da equipe envolvidos diretamente no desenvolvimento dos requisitos.

Integração com o Scrum: Este documento será utilizado durante o *Sprint*.

Conteúdo: Descrever quais riscos o requisito provavelmente produzirá caso ocorram mudanças na arquitetura funcional. Deverá relatar também o impacto previsto e as possíveis contramedidas previstas.

O documento é divido em três partes: “A” - Identificação do projeto e do requisito, “B” - Dados referentes ao risco em si como: prioridade, tipo, probabilidade de ocorrência e estabilidade e, por fim, “C” – Uma avaliação de quais riscos serão produzidos caso a arquitetura funcional tenha uma mudança e, ocorrendo isto, que contramedidas deverão ser executadas.

O item “A” representa os dados gerais do documento, tendo como objetivo principal armazenar a data, o código do projeto a identificação e o nome do requisito. O item “B” tem como função armazenar os riscos relacionados a cada requisito. A seguir uma descrição dos campos do documento deste item:

a) PRIORIDADE: o requisito está dividido em três categorias para a preferência no em seu desenvolvimento: a) alto: o requisito tem alta prioridade para seu desenvolvimento; b) médio, o requisito tem média prioridade para seu desenvolvimento; c) baixo: o requisito tem pouca prioridade para seu desenvolvimento;
b) RISCO: representa a gravidade que o requisito representa para o bom andamento do projeto, e está dividido em três partes: a) alto: o requisito tem alto risco em seu desenvolvimento; b) médio, o requisito tem médio risco em seu desenvolvimento; c) baixo: o requisito tem pouco risco em seu desenvolvimento;
c) PROBABILIDADE DE OCORRÊNCIA: o requisito está dividido em três categorias para a probabilidade de ocorrência de algum risco em seu desenvolvimento: a) alto: o requisito tem alta probabilidade de ocorrência em seu desenvolvimento; b) médio, o requisito tem média probabilidade de ocorrência em seu desenvolvimento; c) baixo: o requisito tem pouca probabilidade de ocorrência em seu desenvolvimento;
d) ESTABILIDADE: o requisito está dividido em três categorias para a durabilidade ou permanência: a) alto: o requisito tem alta estabilidade em seu desenvolvimento; b) médio, o requisito tem média estabilidade em seu desenvolvimento; c) baixo: o requisito tem pouca estabilidade em seu desenvolvimento;

O item “C” tem como função através do campo “AVALIAÇÃO NA ARQUITETURA FUNCIONAL” descrever que riscos o requisito provavelmente produzirá caso ocorram mudanças na arquitetura funcional. Deverá relatar, também, o impacto previsto e as possíveis contramedidas previstas.

A Figura 55 apresenta o documento intitulado Análise dos Riscos.

ANÁLISE DOS RISCOS

DESCRIÇÃO DE REQUISITOS	
DATA:	DD/MM/AAAA
PROJETO:	
ID:	
NOME:	
PRIORIDADE:	( ) Alto ( ) Médio ( ) Baixo
RISCO:	( ) Alto ( ) Médio ( ) Baixo
PROBABILIDADE DE OCORRÊNCIA	( ) Alto ( ) Médio ( ) Baixo
ESTABILIDADE:	( ) Alta ( ) Média ( ) Baixa
AVALIAÇÃO NA ARQUITETURA FUNCIONAL	

Fonte: Adaptada de Gampert (2003)

Figura 55 Análise dos Riscos

Este documento contempla, de certa forma, as sugestões propostas pelo PMBOK[29] na área de análise de riscos. O PMBOK sugere que, para realização da análise qualitativa dos riscos, os documentos gerados devem possuir a seguinte estrutura:

a) Entrada: Identificação dos riscos;
b) Ferramenta: Probabilidade de risco e impacto É a probabilidade que um risco venha a ocorrer: alto, médio ou baixo;
c) Saída: Lista de riscos para analise e gerenciamento.

Desta forma, avaliando os riscos relatados nos requisitos, espera-se que a exigência da prática específica RD SP 3.4-3 do modelo CMMI seja atendida pelo método Scrum.

4.8 Diretriz 8: Estabelecer uma definição das funcionalidades requeridas.

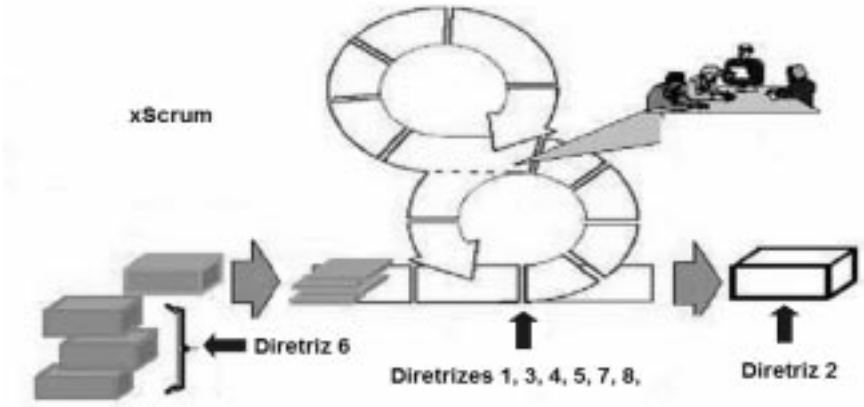
O método ágil Scrum determina quais são as funcionalidades requeridas com a elaboração do *Product Backlog*. Porém, não especifica explicitamente a utilização de casos

[29] Tradução livre do PMBOK 2000, V1, disponibilizada por PMI MG Jan 2002.

de usos, exigência da prática específica RD SP 3.2-1 do modelo CMMI. Desta forma, a proposta é simplesmente agregar a técnica de caso de uso presente na UML. Com isso, espera-se atender as exigências da RD SP 3.2-1.

4.9 Locais de desenvolvimento das Diretrizes propostas

As diretrizes propostas pelo presente trabalho serão aplicadas juntamente com o método Scrum durante a execução do exemplo de aplicação. A diretriz 6 será aplicada logo após a confecção do *Product Backlog*, na fase conhecida como PréGame. As diretrizes 1, 3, 4, 5, 7 e 8 serão aplicadas durante o Sprint, ou na fase chamada Game. Finalmente, a diretriz 2 será aplicada no final do processo, na fase denominada PostGame. A Figura 56 apresenta em quais fases, do método ágil Scrum, serão aplicadas as diretrizes visando a adequação às práticas específicas do modelo CMMI.



Fonte: Adaptado de Schwaber (2002)

Figura 56 Inclusão das atividades ao método Scrum.

5 Exemplo de Aplicação do xScrum

Este capítulo apresenta o exemplo de aplicação para validação da extensão proposta ao método ágil Scrum, denominada **xScrum.** Inicialmente apresenta-se o local da realização do exemplo de aplicação, após a descrição das atividades realizadas. Uma avaliação do processo xScrum, em relação às áreas de processo de Gerenciamento de Requisitos e Desenvolvimento de Requisitos do CMMI, também é realizada com a utilização do método Quicklocus.

5.1 Descrição do local do exemplo de aplicação

O local para a realização do exemplo de aplicação foi o grupo de pesquisa interdisciplinar denominado Simuplan, que o Curso de Ciência da Computação da Universidade de Passo Fundo mantém com a Embrapa Trigo, sob coordenação dos professores Msc Willingthon Pavan e Dr. José Mauricio Cunha Fernandes. O projeto tem como objetivo desenvolver aplicações (softwares) de simulação do crescimento de pragas em algumas culturas, como por exemplo, o trigo. A URL da página do projeto é http://inf.upf.br/simuplan.

O projeto existe há aproximadamente 3 anos, possuindo participações relevantes[30] no cenário científico nacional e internacional. Como resultados já apresentados pelo Simuplan,

[30] Algumas publicações relevantes:
1.FERNANDES, J.M.C. et al. SimTrigo - modelo de simulação do crescimento e desenvolvimento do trigo: predição de risco de Giberela. In: XXIV CONGRESSO NACIONAL DE MILHO E SORGO, 2001, Florianópolis, SC. Anais do... Florianópolis, SC: 2001.
2.FERNANDES, J.M.C.; W. PAVAN. A phenology-based predictive model for Fusarium Head Blight of Wheat. In: 2002 NATIONAL FUSARIUM HEAD BLIGHT FORUM, 2002, Erlanger, KY, USA. 2002. p.154-158.

podem-se citar o desenvolvimento dos seguintes softwares: SimTrigo (Simulação do Crescimento e Desenvolvimento do Trigo e suas Pragas), Sisalert (Sistema de alerta para macieiras, alho, milho, morango e outros), SimuSoja (Simulação do ataque da ferrugem da soja), entre outros.

Para o desenvolvimento dos trabalhos são utilizadas tecnologias como o WAP e linguagens de desenvolvimento próprias para a criação de aplicações voltadas para o celular, como o WML e WMLScript. Também, utiliza-se a linguagem de programação PHP versão 4.0, o banco de dados Postgress versão 7.4.3 e o sistema operacional Conectiva Linux versão 10.0.

Os principais problemas do Simuplan, relacionados ao desenvolvimento de software, são:

a) ausência de um controle dos processos de desenvolvimento de software;
b) inexistência de um processo de gerenciamento e documentação de projeto de desenvolvimento de software;
c) a falta da garantia da qualidade.

Estes problemas são de caráter gerencial, evidenciados principalmente na organização de cada equipe, o que faz com que os projetos funcionem sem acompanhamento padrão e sem controle eficaz das atividades, dificultando sua monitoração e projeção de resultados, gerando com isso, um produto com qualidade questionável.

Com base nestes problemas expostos, optou-se pela escolha do Simuplan como local para exemplo de aplicação. Além disso, no Simuplan os requisitos mudam rapidamente, bem como existe uma grande rotatividade dos integrantes, visto que são alunos da graduação do curso de Ciência da Computação da UPF. Atualmente o projeto conta com aproximadamente 10 pessoas, sendo a maioria formada por desenvolvedores de software. Outro fator preponderante pela escolha do local foi a total receptividade da equipe gestora do Simuplan para realização do exemplo de aplicação proposto neste trabalho.

O objetivo principal do exemplo de aplicação é desenvolver algumas funcionalidades do portal web do Simuplan utilizando o xScrum.

5.2 Desenvolvimento utilizando o xScrum

As atividades de implantação do método ágil Scrum com a extensão proposta (xScrum) no Simuplan começaram no início do mês de agosto de 2004. Nesta data, realizou-se a apresentação dos participantes do projeto e uma explanação do método para todos *stakeholders*. A equipe aceitou a aplicação do xScrum, bem como colocou-se a disposição na colaboração com o novo projeto.

Após os primeiros contatos da equipe com o Scrum, fez-se um levantamento da situação atual do Simuplan. Observaram-se características, comportamentos, hábitos e a forma de trabalho atual do grupo. Constatou-se, inicialmente, a total falta de controle das atividades de gerenciamento nos projetos. Os projetos e suas atividades eram repassados pelo coordenador do grupo para a equipe apenas pela forma verbal, sem nenhum tipo de documentação ou anotações. Exemplo disto é a inexistência de modelagem ou documentação, por exemplo, da base de dados utilizada em cada projeto.

Vale ressaltar que a equipe do Simuplan não adota nenhum método de desenvolvimento ou gerenciamento de software e os conhecimentos técnicos da equipe são limitados e divergentes, ou seja, cada integrante tem seu próprio conhecimento, não sendo especializado em nenhuma técnica ou método. A equipe de desenvolvimento é formada por seis integrantes, todos acadêmicos do curso de graduação em Ciência da Computação da Universidade de Passo Fundo.

Após avaliações preliminares, iniciaram-se os contatos entre o coordenador do grupo, o *Scrum Master* e o cliente, visando esclarecer e definir etapas e atividades para serem realizadas na construção do portal do Simuplan. Vale destacar que para este exemplo de aplicação o autor deste trabalho exerce a função de *Scrum Master*.

A seguir, uma descrição das atividades realizadas no exemplo de aplicação. Inicialmente, na seção 5.2.1, apresentam-se as tarefas executadas na fase *PreGame* do Scrum. Esta fase é responsável basicamente por levantar as necessidades dos clientes e serviu como base para o desenvolvimento dos dois *Sprints* realizados no exemplo de aplicação

Após, detalha-se o primeiro *Sprint*, descrito na seção 5.2.2. O *Sprint* é considerado a etapa mais importante da fase *Game*, pois tem como objetivo principal desenvolver as funcionalidades para entregar ao cliente. Na seção 5.2.3, apresenta-se a fase *PostGame* responsável pela preparação para entrega da funcionalidade final.

Finalmente, após a entrega do produto final da Fase *PostGame* do primeiro *Sprint*, apresenta-se o segundo *Sprint*, descrito nas seções 5.2.4 (Game) e 5.2.5 (PostGame). Nestas seções não são novamente detalhadas as diretrizes, porque estas já foram descritas e comentadas durante o primeiro *Sprint*.

As diretrizes propostas foram distribuídas da seguinte forma durante as fases do Scrum:

a) PreGame: Diretriz 6.
b) Game: Diretrizes 1, 3, 4, 5, 7 e 8.
c) PostGame: Diretriz 2.

Algumas práticas da XP, detalhadas na seção 5.2.6, foram utilizadas durante o desenvolvimento do presente exemplo de aplicação.

5.2.1 Fase *PreGame*

A técnica utilizada para o levantamento das necessidades dos usuários foi a JAD, que teve como principal função a definição dos objetivos do projeto. A equipe Scrum considerou esta técnica muito interessante e aconselhou que ela fosse novamente utilizada, quando necessária. Isto confirmou, de certa forma, que a utilização de reuniões freqüentes e

rápidas, tão presentes nos métodos ágeis é uma boa alternativa para o levantamento das necessidades dos usuários. O resultado desta técnica contribuiu para a confecção do *Product Backlog*, gerando as primeiras funcionalidades do projeto. Com a utilização desta técnica, atendeu-se, de forma plena a solicitação da Diretriz 6 do xScrum que sugere a utilização de alguma técnica para eliciação de requisitos. A Figura 57 mostra a fase do Scrum onde foi aplicada a Diretriz 6 (definir uma técnica de eliciação de requisitos).

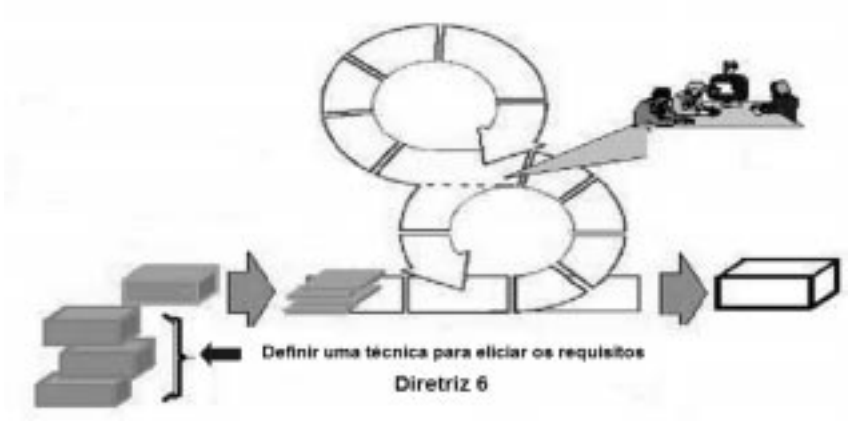


Figura 57 *Aplicação da diretriz 6*

Após a utilização da Diretriz 6, elaborou-se um dos principais elementos do Scrum, o *Product Backlog*. Esta lista foi organizada da seguinte forma: os níveis de prioridade das funcionalidades foram estipulados pelo cliente sem participação efetiva da equipe, também foram indicados os responsáveis pela execução e, finalmente, estabeleceram-se os tempos estimados (em horas) para o desenvolvimento dos trabalhos. Observa-se que a definição do tempo foi baseada, principalmente, na experiência do cliente e do *Scrum Master*.

O *Product Backlog* gerado pode ser observado na 0. Até este momento as atividades restringiram-se ao *Scrum Master*, ao cliente e ao *Product Owner*, não havendo participação de outros integrantes da equipe.

Prioridade	Item	Descrição	Tempo (horas)	Responsável
Muito Alta				
	1	Cadastro de agricultores e produtores	20	Élder e Adriano
	2	Módulo de alimentação de dados	20	Telmo e Samuel
	3	Módulo de simulação	40	Ping e Jonas
	4	Módulo de autorização	8	Élder e Adriano
Alta				
	5	Obtenção de dados pelo web	20	Élder e Samuel
	6	Mapas de informação	40	Jonas e Élder
	7	Módulo de relatórios	20	Samuel e Ping
	8	Mapas de risco	40	Adriano e Ping
	9	Módulo de importação de dados	10	Telmo e Adriano
	10	Geração de gráficos	12	Ping
	11	Importação de outros modelos (CROPSIM);	20	Élder e Telmo
	12	Cadastro de municípios brasileiros	8	Adriano
	13	Previsões de incidência baseados em previsões (tempo) futuras	30	Jonas e Élder
Média				
	14	Notícias (Alimentação)		Adriano e Ping
	15	Fóruns		Telmo e Adriano
	16	Readaptação do Sisalert		Jonas e Élder

Fonte: primária

Figura 58 Product Backlog

5.2.2 Fase *Game* - Primeiro *Sprint*

Após a etapa da definição do *Product Backlog*, a equipe reuniu-se novamente (através da *Sprint Planning Meeting)* para a definição de quais funcionalidades fariam parte do primeiro *Sprint Backlog*. Neste encontro, todas as funcionalidades com prioridade do tipo “muito alta”, com exceção do “Módulo de Simulação”, compuseram o *Sprint Backlog*. A retirada desta funcionalidade foi solicitada pelo *Product Owner* devido à complexidade no seu desenvolvimento. O resultado do *Sprint Backlog* formará a lista das funcionalidades que serão desenvolvidas durante o *Sprint*.

A 0 apresenta o primeiro *Sprint Backlog*, para o desenvolvimento do projeto do Portal do Simuplan.

Sprint Backlog				
Prioridade	**Item**	**Descrição**	**Tempo (horas)**	**Responsável**
Muito Alta	1	Cadastro de agricultores e produtores	20	Élder e Adriano
	2	Módulo de alimentação de dados	20	Telmo e Samuel
	3	Módulo de autorização	8	Élder e Adriano

Fonte: primária

Figura 59 *Funcionalidades definidas no Sprint Backlog 1*

A primeira funcionalidade (Cadastro de agricultores e produtores), item 1 da 0, especifica a manutenção da classe Pessoa que armazenará os dados pessoais dos usuários, como nome, endereço e quais as estações que ele estará conectado e que deseja receber informações.

A segunda funcionalidade, item 2 da 0, é referente às informações climáticas sobre as regiões das estações e são baseadas nos dados obtidos das fontes externas. Estes dados podem ser inseridos tanto pelo usuário como por um sistema de busca na internet. Neste primeiro *Sprint* foi feita a manutenção dos dados climáticos através de uma entrada para os usuários, ou seja, os dados são fornecidos pelos usuários para as estações que estejam ligadas a eles. Basicamente, é um espaço no portal em que o usuário fornece dados como temperatura, umidade e vento, para as suas estações.

A última funcionalidade, item 3 da 0, especificada para o primeiro *Sprint* é o módulo de autorização, onde o usuário com permissão de moderador definirá o nível de acesso dos usuários cadastrados. Quando um usuário se cadastra no portal este deve confirmar sua inscrição por e-mail e, após a confirmação, o moderador define qual a permissão que ele terá no *site*, restringindo seu acesso apenas a determinadas seções.

Definidas as funcionalidades do primeiro *Sprint*, preencheu-se o documento de visão. Este documento é proposto na Diretriz 5. Esta diretriz, foi considerada pela equipe muito válida, pois, de certa forma, contribui para o entendimento do software a ser desenvolvido.

A Figura 60 mostra onde são aplicadas as diretrizes relacionadas com a Fase Game de cada Sprint: diretriz 1 (Gerenciar as mudanças dos requisitos); diretriz 3(Desenvolver uma matriz de rastreabilidade); diretriz 4 (Alocar os requisitos aos componentes dos produtos); diretriz 5: (Identificar as interfaces dos requisitos); diretriz 7 (Analisar os riscos) e diretriz 8 (Estabelecer uma definição das funcionalidades requeridas).

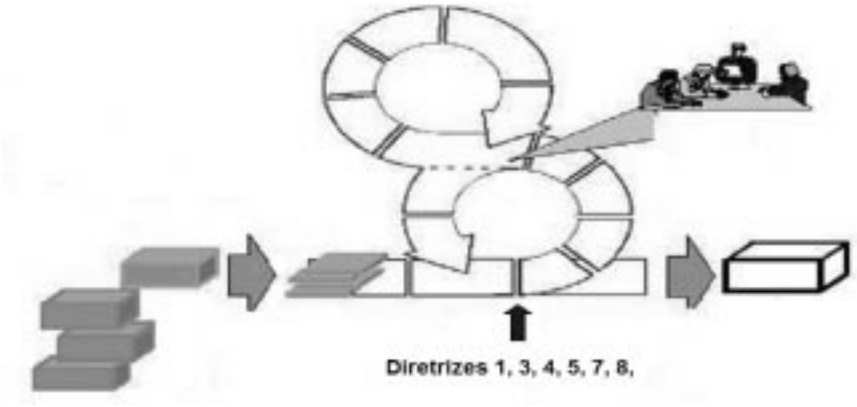


Figura 60 Diretrizes na fase Game

A 0 apresenta o documento elaborado que visa atender as necessidades da Diretriz 5 (identificar as interfaces dos requisitos).

DOCUMENTO DE VISÃO	
DATA:	09/08/2004
PROJETO:	001
INTRODUÇÃO: Agregar funcionalidades ao portal Simuplan.	
VISÃO DO PRODUTO: **(Domínio, restrições e interfaces)** O produto possuirá cadastros dos usuários, simuladores de crescimentos de pragas nas plantas, consultas a previsões do tempo além de elaborar mapas de riscos. O Cadastro de agricultores e produtores especifica o cadastramento de usuários do portal, onde cada interessado em se cadastrar deverá fornecer seus dados como nome, endereço e as estações que ele estará ligado e que deseja receber informações. Entre estas informações estão dados resultantes de simulações realizadas e alertas. Outra funcionalidade do produto é referente às informações climáticas sobre as regiões das estações baseadas nos dados obtidas das fontes externas. Estes dados podem ser inseridos tanto pelo usuário como por um sistema de busca na internet. Somente pessoas autorizadas poderão ter acesso aos dados cadastrados.	
RESPONSÁVEL:	Elias

Fonte: primária

Figura 61 Identificar as interfaces dos requisitos

Com o documento de visão elaborado, contendo um resumo dos principais tópicos que o software deverá possuir, elaboraram-se os casos de uso das três funcionalidades requeridas. A 0 demonstra o caso de uso apresentado à equipe Scrum para o desenvolvimento da primeira funcionalidade.

Caso de uso: Realizar cadastro
Atores: Cliente
Cenários:
1. Identifica-se pelo seu nome, endereço, complemento, bairro - Cliente
2. Escolhe determinado país selecionando sua descrição - Cliente
3. Escolhe determinado estado selecionando sua descrição - Cliente
4. Escolhe determinado cidade selecionando sua descrição – Cliente
5. Identifica-se pelo seu cep, telefone e telefone celular – Cliente
6. Escolhe determinada operadora selecionando sua descrição – Cliente
7. Identifica-se pelo seu email – Cliente
8. Escolhe determinado sexo – Cliente
9. Informa o texto relativo ao login, com até 8 caracteres e senha - Cliente
10. Informa o texto relativo à senha, com até 6 caracteres - Cliente
11. Informa o texto relativo a repetição da senha - Cliente
12 O site valida a senha e o login digitados pelo cliente

Requisitos Especiais:
1. O cliente pode abandonar o cadastro a qualquer momento.

Entrevistado: Willingthon Pavan
Data: Agosto de 2004
Versão: 1.0

Fonte: primária

Figura 62 *Caso de Uso: Realizar cadastro*

A utilização do caso de uso atendeu a exigência da Diretriz 8 do xScrum. A equipe Scrum considerou a utilização desta técnica válida. Isto, de certa forma, confirmou a importância da utilização dos casos de uso.

Após a definição dos requisitos a serem desenvolvidos foram designados os componentes dos produtos e seus requisitos foram alocados. Desta forma, a Diretriz 4 do xScrum foi atendida. A equipe mostrou-se indiferente à apresentação desta documentação e percebeu-se que o documento proposto foi preenchido com uma certa “inquietação”.

A 0 apresenta a documento elaborado que visa atender as necessidades da Diretriz 4.

ALOCAÇÃO				
DATA:	10/08/2004			
PROJETO DE TRAB / COMPONENTE:	001			
NOME:	Cadastro de Usuários			
REQUISITOS ALOCADOS				
ID	DESCRIÇÃO	DERIVAÇÕES	TEMPORÁRIO	RELAC.
010	Pessoa	Não	Não	Sim
011	Estação	Não	Não	Sim

ID REQUISITO	INCONSISTÊNCIA
001	Não foram encontradas inconsistências neste requisito.

Fonte: primária

Figura 63 Alocar os requisitos aos componentes do produto

O desenvolvimento do primeiro *Sprint* teve início na terceira semana de agosto de 2004, tendo como objetivo o desenvolvimento do "cadastro de agricultores e produtores" (item 1 da Figura 2 definido no *Sprint Backlog*). Nesta semana, realizaram-se as primeiras reuniões diárias (*daily meeting*) propostas pelo Scrum.

Após a primeira *daily meeting*, entregou-se à equipe um documento contendo detalhes dos requisitos. Este documento atende a Diretriz 1 da proposta de extensão do Scrum. O documento entregue também atendeu uma necessidade da equipe, que solicitou ao *Scrum Master* um detalhamento dos requisitos. Cabe destacar que o Scrum não define detalhamento das atividades durante o *Sprint*.

A Figura 64 apresenta a documento elaborado que visa atender as necessidades da Diretriz 1.

DESCRIÇÃO DE REQUISITOS			
ID:	001		
DATA:	9/08/2004		
PROJETO:	001		
NOME:	Cadastro Usuários		
ORIGEM:	Willingthon		
TIPO_REQ:	(X) Funcional	( ) Não funcional	
DESCRIÇÃO:	Deverá armazenar os dados pessoais dos usuários, como nome, endereço e quais as estações que ele estará conectado e que deseja receber informações.		
SOLUÇÃO:	Desenvolver uma classe para os usuários		
PRIORIDADE:	(X) Alto	( ) Médio	( ) Baixo
TIPO:	(X) Cliente	( ) Produto	( ) Interface
RISCO:	( ) Alto	(X) Médio	( ) Baixo
VERSAO DESTINO:	V1.0		
DEPENDÊNCIA:			

IMPACTO NOS REQUISITOS	
NOME	ALTERAÇÃO
	I / A / E
RESPONSÁVEL:	Elias

Figura 64 Gerenciar as mudanças dos requisitos

Os primeiros dias de trabalho decorrentes ao início do *Sprint* não foram relacionados a nenhuma das funcionalidades presentes no *Sprint Backlog*, ou seja, durante uma semana a equipe envolveu-se com o estudo da uma ferramenta denominada *Struts*. O objetivo deste estudo foi avaliar esta ferramenta para provável utilização no Simuplan, o que não ocorreu devido a sua complexidade. Desta forma, o trabalho não foi produtivo, além de comprometer o prazo final estipulado para o primeiro *Sprint*, fato este que aconteceu. Por fim, adotou-se a linguagem PHP como padrão para o desenvolvimento do projeto.

O primeiro *Sprint* durou aproximadamente 20 dias e a equipe Scrum iniciou o desenvolvimento dos produtos para criar uma versão executável das funcionalidades requeridas no *Product Backlog*. Não foram adicionados novos itens ao *Product Backlog*. Neste momento, a equipe Scrum analisou os prováveis riscos dos requisitos, com a utilização da Diretriz 7.

A 0 apresenta a documento elaborado que visa atender as necessidades da Diretriz 7.

DESCRIÇÃO DE REQUISITOS			
DATA:	19/08/2004		
PROJETO:	001		
ID:	001		
NOME:	Cadastro de Usuários		
PRIORIDADE:	(X) Alto	( ) Médio	( ) Baixo
RISCO:	( ) Alto	(X) Médio	( ) Baixo
PROBABILIDADE DE OCORRÊNCIA	(X) Alto	( ) Médio	( ) Baixo
ESTABILIDADE:	(X) Alta	( ) Média	( ) Baixa
AVALIAÇÃO NA ARQUITETURA FUNCIONAL	Este requisito não possui um grande risco caso ocorram mudanças na arquitetura funcional. Os impactos resultantes deste requisito são considerados irrelevantes.		

Fonte: primária

Figura 65 Análise dos riscos

Por exigência da Diretriz 3, a matriz de rastreabilidade foi preenchida pela equipe Scrum durante o desenvolvimento do *Sprint.* Porém, devido à simplicidade das funcionalidades com pouca quantidade de requisitos a equipe utilizou somente um documento de rastreabilidade de requisitos. Percebeu-se claramente, nesta diretriz, uma insatisfação no seu preenchimento, alegando que “isto não será utilizado”.

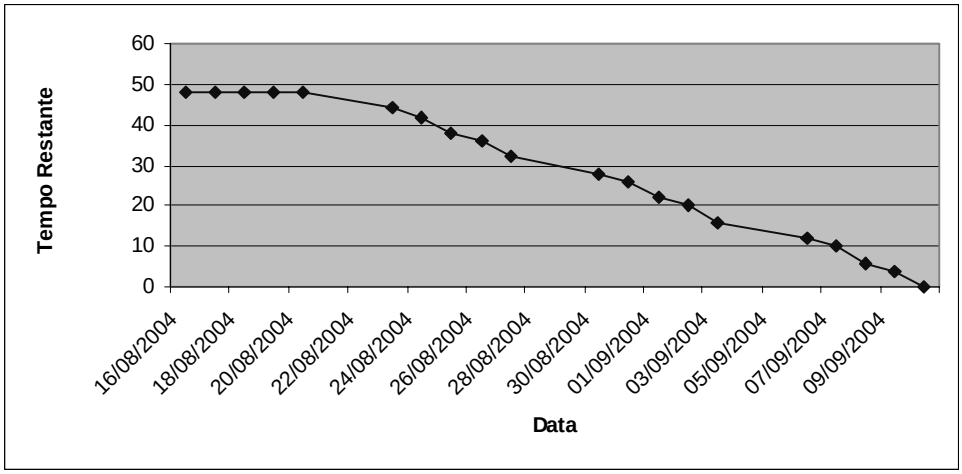
A 0 apresenta o documento elaborado que visa atender as necessidades da Diretriz 3.

DATA:	30/08/2004					
PROJETO:	001					
Tipo: Requisito/ Componente	**Agricultores**	**Autorização**	**Alimentação**			
Pessoa	AD	AD				
Estação	AD	AD				
Cidade	AD					
Estado	AD					
País	AD		AD			
Clima			AD			

Fonte: primária

Figura 66 Desenvolver uma matriz de rastreabilidade

A 0 mostra o gráfico gerado relatando como foi o andamento do primeiro *Sprint*, auxiliando, de certa forma, o gerenciamento para o novo *Sprint*. A geração de gráficos de andamento é uma exigência do *Scrum*.



Fonte: primária

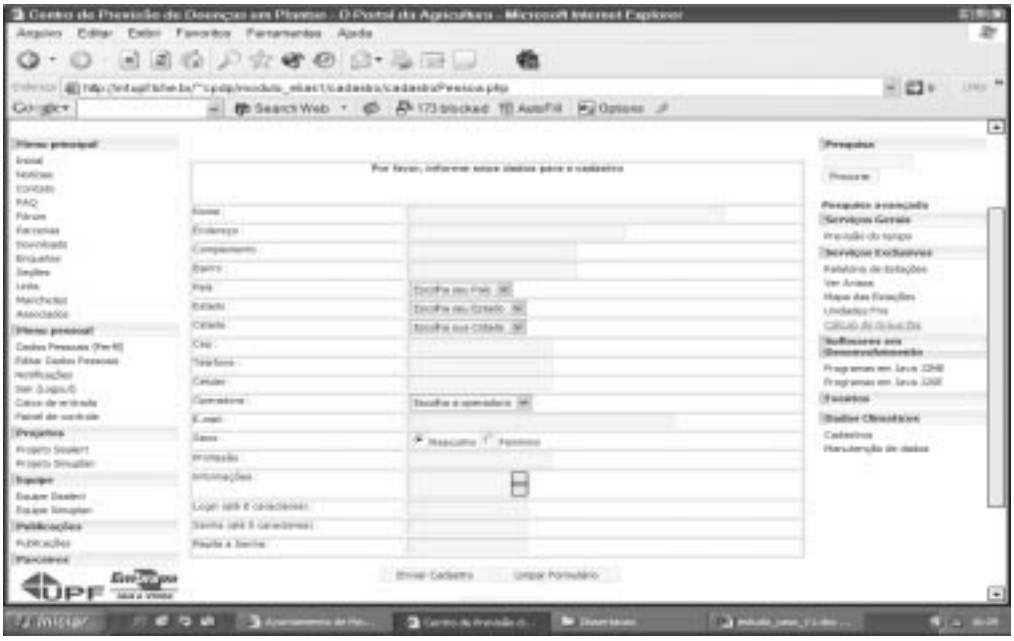
Figura 67 Primeiro *Sprint*

O primeiro *Sprint*, que durou apenas 22 dias úteis, com tempo estimado no *Product Backlog* em 48 horas passaram para 60 horas, excedendo com isso, cerca de 25% das atividades previstas. Isto ocorreu porque na primeira semana não houve o desenvolvimento

das aplicações devido ao estudo da ferramenta *Struts*. Outro fator a se destacar para aumento no tempo das atividades previstas foi o período gasto pelos programadores para documentação e análise do material recebido. Vale ressaltar que a 0 apresenta somente o tempo restante do desenvolvimento e não quais foram as atividades realizadas.

5.2.3 Fase *PostGame* - Primeiro *Sprint*

A 0 mostra a tela capturada do software produzido que representa a primeira funcionalidade do *Product Backlog* (cadastro de agricultores e produtores), resultado do primeiro *Sprint*. Uma melhor visualização poderá ser obtida em http://inf.upf.tche.br/~cpdp/modulo_elias1/cadastro/cadastroPessoa.php.



Fonte: primária

Figura 68 Primeiro incremento entregue

Ao final do primeiro incremento, entregou-se à equipe o documento *Component Product Backlog*, com detalhes da alocação dos requisitos e prováveis inconsistências encontradas. Este documento atende a Diretriz 2 da proposta de extensão do Scrum (xScrum). A equipe, durante o *Sprint Review Meeting*, analisou o documento não sugerindo

alterações. A Figura 69 mostra a fase do Scrum onde foi aplicada a Diretriz 2 (verificar as inconsistências).

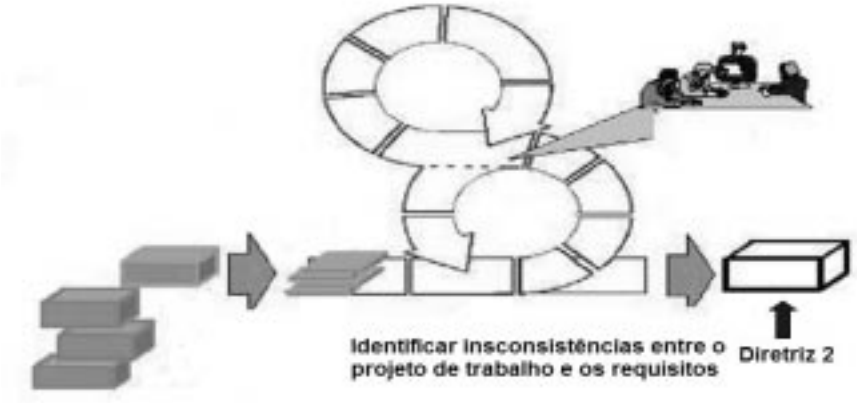


Figura 69 Identificar inconsistências entre o projeto de trabalho e os requisitos

Ao final do *Sprint*, a revisão foi realizada com a presença de toda a equipe e o *Product Owner*. Realizou-se então, o planejamento para o segundo *Sprint,* com re-priorização de funcionalidades, definição de entrega e duração do próximo *Sprint*. Após a entrega do *Product Increment,* resultado do primeiro Sprint, iniciou-se o segundo *Sprint* descrito a seguir:

5.2.4 Fase *Game* - Segundo *Sprint*

Com a entrega do *Product Increment*, a equipe reuniu-se novamente (*Sprint Planning Meeting)* para definição de quais funcionalidades fariam parte do segundo *Sprint Backlog*. Neste momento, realizou-se também uma breve explanação sobre os requisitos dessas funcionalidades. Nesta explanação foi apresentada uma visão geral de cada funcionalidade, bem como uma descrição do produto final desejado.

A 0 apresenta o segundo *Sprint Backlog*, resultado do *Sprint Planning Meeting*, para o desenvolvimento do projeto do Portal do Simuplan, com estimativa de duração em horas das atividades e seus respectivos responsáveis para o desenvolvimento.

Prioridade	Item	Descrição	Tempo(horas)	Responsável
	4	Obtenção de dados pela web	20	Telmo e Adriano
	5	Módulo de relatórios	20	Samuel

Fonte: primária

Figura 70 Funcionalidades para o Segundo *Sprint Backlog*

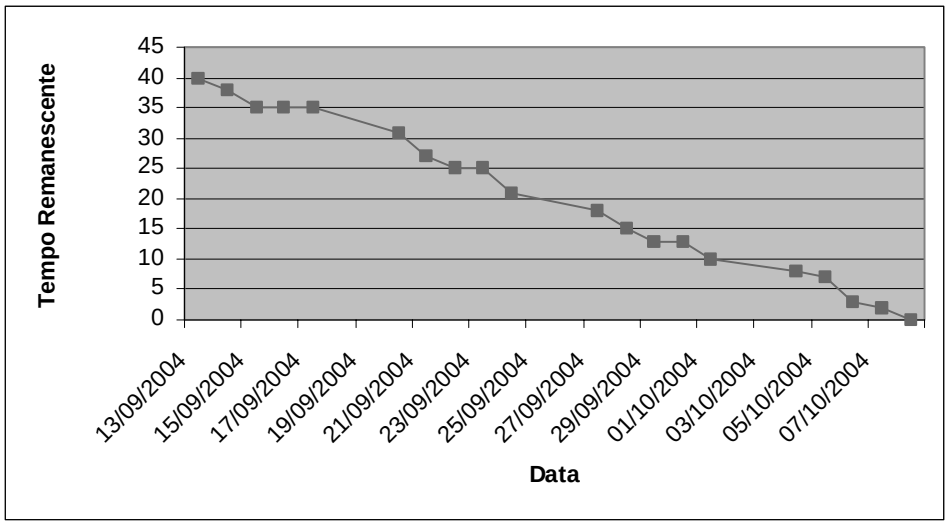
A primeira funcionalidade (obtenção de dados pela web), item 4 da 0, consiste basicamente na busca de informações climáticas no *site* canaldotempo.com, onde os dados colhidos neste site serão inseridos em uma base de dados local. O *script* deverá ser executado de hora em hora colhendo os dados de cada região cadastrada. As regiões serão definidas pelas estações cadastradas, ou seja, o sistema buscará as informações para a região de cada estação. O sistema será uma adaptação do script URLReader.java que se encontra no servidor da máquina inf[31], o qual será executada uma vez ao dia para buscar os dados de previsão do tempo para os próximos dez dias. Os dados capturados são: temperatura atual; vento; direção do vento; ponto de orvalho;umidade; visibilidade; barômetro e índice UV;

A segunda funcionalidade deste *Sprint* (módulos de relatórios), item 5 da 0, especifica a geração de relatórios através de figuras de mapas coloridos. Os dados que servirão de base para a geração dos mapas serão obtidos através de simulações feitas para cada tipo de infecção (módulo de simulação). Estas simulações são baseadas em informações climáticas advindas de outra base de dados. Esta base é alimentada tanto pelo próprio usuário como por um sistema de busca automática na web. No entanto, as informações hoje são apenas de previsão da temperatura para os próximos dez dias, sendo insuficientes para gerar as simulações. Para suprir esta falta de dados, o *script* será adaptado para obter os dados restantes.

[31] Nome da máquina servidora do curso de Ciência da Computação da UPF.

Observa-se que o usuário somente receberá informações, tanto dos mapas de risco como dos relatórios, sem interagir com o módulo de busca de informações na web, que será executado a cada hora e irá alimentar a base de dados que será usada por outros módulos.

Para a construção das atividades estipuladas neste sprint foram estimadas 40 horas de trabalho, visto que cada atividade estava programada para levar 20 horas cada e a equipe foi reduzida. O desenvolvimento das duas funcionalidades no segundo *Sprint* começou dia 13 de setembro e o encerrou dia 11 de outubro. Ao final da segunda semana de atividades restavam apenas para o desenvolvimento, algumas listagens e a realização da integração com o portal e o módulo de segurança para finalização das funcionalidades. A 0 mostra o gráfico gerado relatando o andamento do segundo *Sprint*.



Fonte: primária

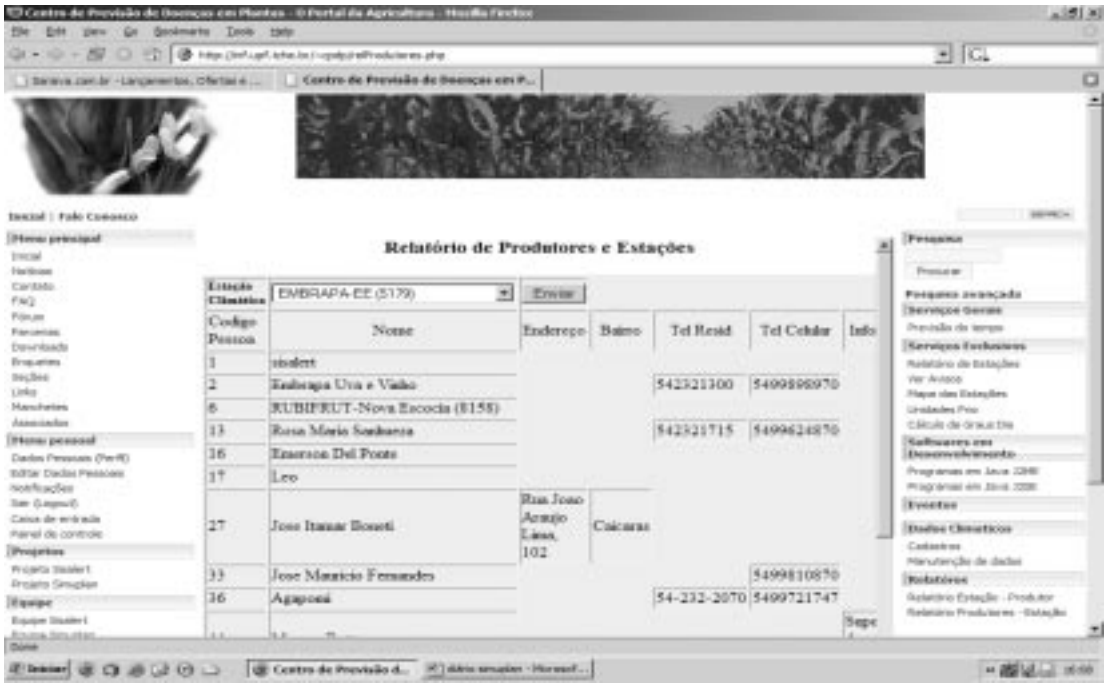
Figura 71 Segundo Sprint

Analisando-se a 0, percebe-se que ao final do segundo *Sprint*, que durou apenas 19 dias úteis (o máximo permitido pelo Scrum é de 30 dias), o tempo designado pelo Scrum *master* para o desenvolvimento das atividades foi suficiente, não excedendo o tempo estimado. Isto ocorreu, de certa forma, pela experiência adquirida pelo Scrum *master*, durante o primeiro *Sprint*.

Também pode-se observar que, diferentemente do primeiro *Sprint*, o tempo utilizado para o desenvolvimento das atividades diárias se manteve em equilíbrio, formando “quase” uma linha reta entre o início e o final dos trabalhos. Isto demonstra, conforme Schwaber (2002, p.84), “um controle excelente sobre o trabalho” desenvolvido.

5.2.5 Fase *PostGame* - Segundo *Sprint*

A 0 mostra a tela capturada do software produzido que representa o relatório de Produtores e Estações. Com isso, entregou-se o segundo incremento e toda documentação gerada pelo xScrum.



Fonte: primária

Figura 72 Segundo incremento entregue

Vale ressaltar que durante a realização do segundo *Sprint* também foram seguidas novamente todas as diretrizes propostas no xScrum.

Algumas alterações na proposta do presente trabalho foram realizadas durante a execução do exemplo de aplicação. As principais alterações ocorreram nos locais do Scrum

em que as diretrizes foram incorporadas, por exemplo, a diretriz 3 que inicialmente estava na fase PostGame passou para a Fase Game.

5.2.6 Práticas utilizadas da XP no exemplo de aplicação

Esta seção descreverá algumas práticas utilizadas da XP durante a realização dos *Sprints* do exemplo de aplicação. Foram aplicadas as seguintes práticas:

a) Propriedade Coletiva do Código Gerado: todos os desenvolvedores conhecem o código gerado independente de terem sido os autores.
b) *Releases* Pequenas: a entrega do software foi organizada para ocorrer incrementalmente, sendo entregue em um prazo médio de 20 dias;
c) Cliente Sempre Disponível: o cliente sempre esteve disponível durante o projeto;
d) Excesso de Tempo: esta prática foi respeitada durante o projeto, principalmente porque todos integrantes da equipe Simuplan são alunos do curso e, com isso, não trabalharam 8h diárias no projeto, algo que teoricamente seria normal em outra empresa qualquer;
e) Programação em Pares: utilizou-se também a prática da programação em pares proposta pela XP. Inicialmente, os programadores tiveram muita resistência pela nova forma, porém percebeu-se que após uma semana estavam mais habituados e relataram que gostaram da experiência.
f) Teste primeiro: foram executados pequenos testes antes do desenvolvimento dos requisitos no *Sprint.*
g) Planejamento da Versão: a elaboração do *Product Backlog* pode ser considerada como um planejamento, pois definem prioridades, estimativas de tempo, etc.
h) Estórias de usuários: utilizaram-se os casos de uso.

Cabe ressaltar que os valores da XP, como Simplicidade, Comunicação, Coragem e Realimentação, foram também utilizados durante todo o processo de desenvolvimento do

software. A equipe considerou válida a utilização destes valores, porém, como são considerados subjetivos, não foram avaliados.

Além das diretrizes propostas, serão incluídas no projeto práticas de desenvolvimento de software advindas da XP. Vale lembrar que Schwaber (2002, p. 20), defende a união de práticas da XP ao Scrum, relatando que existe uma “perfeita harmonia” entre estes dois métodos. Esta inclusão tem por objetivo fornecer ao método Scrum práticas de engenharia de software inexistentes neste método. Outras características na XP, que serão incluídas no Scrum, são os valores como simplicidade, comunicação, coragem e realimentação.

5.3 Avaliação do xScrum no exemplo de aplicação

Esta seção tem como objetivo apresentar a avaliação do xScrum no exemplo de aplicação através do método Quicklocus. Este método é utilizado para avaliação de desenvolvimento de software em pequenas organizações.

5.3.1 Quicklocus

O Quicklocus (Kohan, 2003) tem como objetivo, com baixo custo de implantação, servir como base para um plano de melhoria de processo de desenvolvimento de software. A seguir são apresentadas algumas considerações no uso deste método para aplicação no presente trabalho:

a) A organização deve ser pequena, com, no máximo, 50 pessoas envolvidas no processo de desenvolvimento de software. O Simuplan possui apenas 10 pessoas envolvidas diretamente no processo de desenvolvimento de software.
b) A seleção em até 3 áreas de processo é uma condição/restrição básica do método Quicklocus para a avaliação ser realizada em um dia de trabalho. O escopo do presente trabalho é avaliar somente duas áreas de processo do modelo CMMI, REQM e RD.

c) É baseado e compatível com os requisitos propostos pelo ARC V1.1, como acontece no SCAMPI, que é o método que a SEI utiliza para avaliações (método classe A que fornece o nível de maturidade/capacidade do processo).
d) É considerado como Classe “C”. Os métodos desta classe devem satisfazer apenas um subconjunto de requisitos, e tem como objetivo “dar uma rápida olhada” em todo processo de desenvolvimento de software, conforme destaca Kohan (2003, p. 49). Isto pode ser considerado como uma limitação do trabalho, porém simplifica a avaliação da aplicação do método proposto.

Segundo Kohan (2003, p. 112), a aplicação do Quicklocus ocorre em três fases: Preparação, Avaliação e Pós-avaliação. A fase da Preparação contempla as atividades que permitem uma compreensão do contexto da avaliação e a preparação dos trabalhos. A Avaliação compreende basicamente a coleta dos dados. Finalmente, a fase da Pós-avaliação contempla as atividades que encerram os trabalhos da avaliação com o armazenamento e emissão do relatório final.

5.4 Aplicação do método

Nesta seção é realizada a descrição das três fases desenvolvidas para avaliação deste exemplo de aplicação, a partir da utilização do Quicklocus.

5.4.1 Preparação

A primeira exigência da preparação é a definição do escopo organizacional, modelo de referência e escopo no modelo de referência. Neste caso, o escopo organizacional é o Simuplan. Com relação ao modelo de referência, optou-se pelo CMMI, pela necessidade que o Simuplan possui na melhoria do processo de desenvolvimento de software. A definição do escopo no modelo de referência, ocorreu pela abrangência do presente trabalho, que são as áreas de processo REQM e RD do modelo CMMI.

Após a definição do escopo e o modelo de referência, elaborou-se o planejamento da avaliação. Este plano pode ser observado no Anexo 2, que destaca os objetivos da

avaliação, a identificação do patrocinador, a definição da equipe, quais projetos serão avaliados e a definição das diretrizes e orientações dos participantes. Ao final, fez-se um planejamento da logística de avaliação. Este planejamento previu a infra-estrutura mínima para execução das atividades na organização.

5.4.2 Avaliação

A segunda fase compreendeu a coleta preliminar de dados bem como as atividades do dia de trabalho na organização. Esta fase envolveu cinco passos: coleta de dados fonte 1; orientação dos participantes; coleta de dados fonte 2; graduação final dos dados e, finalmente, a emissão do relatório preliminar.

A coleta de dados fonte 1 tem como objetivo recolher informações sobre o processo de desenvolvimento de software, através de um questionário que é analisado preliminarmente. O questionário pode ser observado no Anexo 3.

A orientação aos participantes é bem simples e consiste basicamente numa reunião, onde foram expostos à alta administração e à equipe de avaliação todos os objetivos da avaliação. Este etapa foi realizada durante a reunião para avaliação.

A coleta de dados fonte 2 é formada por entrevistas com todos integrantes do grupo de desenvolvimento. Foi precedida por uma reunião da equipe de avaliação tendo como objetivo básico buscar um entendimento do processo ou “alinhamento” de atitudes e ações que serão tomadas ao longo do dia de trabalho, conforme destaca Kohan (2003, p.119). Esta etapa foi realizada, porém não se gerou documentação devido a simplicidade da tarefa executada.

Finalmente, a etapa da emissão do relatório preliminar constitui basicamente elaboração de um documento que contém a graduação dos itens de maior grau de detalhe do modelo de referência. Este documento pode ser observado através do Anexo 4. Cada item de maior grau de detalhe do modelo, no caso as práticas das áreas de processo REQM

e RD, são graduadas com uma das seguintes opções, segundo afirma Kohan (2003, p.12): “E” o item existe, “M” o item existe e deve ser melhorado e “N” o item não existe.

5.4.3 Pós-avaliação

A etapa da Pós-avaliação contempla as atividades que encerram os trabalhos da avaliação, contendo três passos: emissão do relatório final; apresentação dos resultados finais e o armazenamento do resultado da avaliação.

A emissão do relatório final é constituída basicamente da elaboração e da entrega de um documento, que deve conter inicialmente a identificação da organização; identificação do método de avaliação utilizado, no caso o Quicklocus; data e local; escopo da organização e projetos; pessoas envolvidas; objetivos e relação de todos os itens em maior grau de detalhes do escopo do modelo com a graduação respectiva. Este documento pode ser observado no ANEXO 5. A apresentação dos resultados é uma reunião na qual são divulgados os resultados da avaliação e que pode ser assistida por todos os membros da organização. O armazenamento dos resultados deve cuidar para que as informações coletadas durante a avaliação, que são confidenciais, não sejam acessadas por pessoas externas à organização.

5.5 Resultados da avaliação

Nesta seção são apresentados os resultados finais do exemplo de aplicação realizado no Simuplan com a utilização do xScrum. São descritas as avaliações das duas áreas de processo: REQM e RD

5.5.1 Avaliação Final da área de processo REQM

A área de processo do Gerenciamento de Requisitos descreveu as atividades para obter e controlar as mudanças nos requisitos assegurando que as mudanças relacionadas aos planos do projeto permanecessem atualizadas. Isso aconteceu devido ao desenvolvimento da rastreabilidade dos requisitos do cliente.

Esta área assegurou as alterações dos requisitos relacionados aos planos do projeto, atividades e produtos. Os requisitos foram gerenciados e não se verificaram inconsistências do projeto com os artefatos. O objetivo desta área de processo foi gerenciar os requisitos e para o desenvolvimento desta área de processo foram realizadas as seguintes atividades

a) Gerenciaram-se todas as mudanças dos requisitos;

b) Mantiveram-se o relacionamento entre os requisitos, o plano do projeto e os produtos de trabalho;

c) Identificaram-se as inconsistências entre os requisitos, o plano do projeto e os produtos de trabalho;

d) Realizaram-se ações corretivas.

Desta forma, a área de processo Gerenciamento de Requisitos do modelo CMMI está totalmente presente no Simuplan.

5.5.2 Avaliação Final da área de processo RD

A área de processo do desenvolvimento de requisitos identificou as necessidades do cliente e traduziu estas necessidades em requisitos do produto. Estes requisitos foram analisados para produzirem uma solução conceitual, descrevendo a apresentação do produto, as características, etc.

O objetivo desta área de processo foi produzir e analisar os requisitos dos clientes, dos produtos e os componentes do produto. Para o desenvolvimento desta área, foram realizadas as seguintes atividades:

a) Eliciou-se, analisou-se e validou-se as necessidades e expectativas dos clientes visando um bom entendimento entre todos stakeholders do projeto;

b) Colecionou-se e coordenou-se as necessidades dos stakeholders;

c) Estabeleceu-se os requisitos do cliente;

d) Estabeleceu-se um produto inicial e os componentes do produto em concordância com os requisitos do cliente.

Desta forma, a área de processo Desenvolvimento de Requisitos do modelo CMMI está totalmente presente no Simuplan.

Pelo fato do Quicklocus ser considerado um método de Classe C, isto, de certa forma, pode ser analisado como uma limitação da avaliação realizada, pois o método de avaliação desta classe, como citado anteriormente, tem como objetivo satisfazer apenas um subconjunto de requisitos, analisando de forma superficial todo processo de desenvolvimento de software. Porém, vale destacar não foi utilizado um método do tipo Classe A ou B devido a sua rigidez e complexidade na sua avaliação. Com isso, pode-se concluir que o xScrum é uma alternativa para pequenas organizações que desejam atender as práticas REQM e RD do modelo CMMI.

A seção 5.5 apresentou os resultados finais da avaliação realizada. Os dados apresentados no relatório final mostram que todas as práticas específicas das áreas de processo REQM e RD do modelo CMMI foram atendidas.

5.6 Análise do xScrum conforme os princípios ágeis

Nesta seção é realizada uma análise do xScrum baseada na verificação do grau de satisfação dos princípios ágeis pelos métodos ágeis. Este análise tem como principal objetivo verificar se a proposta continua atendendo aos princípios ágeis em função das alterações/inclusões sugeridas ao método ágil Scrum. A seguir, a análise realizada.

a) No princípio 1 (a prioridade é satisfazer o cliente através da entrega contínua e rápida de uma versão do software com valor agregado), o xScrum manteve a entrega rápida do software através dos Sprints.
b) No princípio 2 (as alterações sobre os requisitos são bem vindas), o xScrum agregou documentos de controles para estas alterações.

c) No princípio 3 (entrega de software freqüente a cada semana ou a cada mês, sempre visando o menor tempo possível), o xScrum mantém o ciclo de 30 dias para a entrega de novas funcionalidades.
d) No princípio 4 (os especialistas no negócio e os desenvolvedores devem trabalhar juntos diariamente no desenvolvimento do projeto), o xScrum mantém a união destes especialistas e desenvolvedores, contribuindo, assim, com a integração contínua e diária.
e) No princípio 5 (manter uma equipe motivada é o principal objetivo deste princípio), o xScrum mantém a premissa básica do Scrum de que a equipe de trabalho deve estar unida e motivada.
f) No princípio 6 (a maneira mais eficiente de comunicação da equipe é através de uma conversa face-a-face), o xScrum pode ter perdido um pouco de agilidade pois foram desenvolvidos novos documentos, porém, isto não evitou que a equipe continuasse trabalhando no mesmo ambiente com reuniões diárias.
g) No princípio 7 (a simplicidade é essencial), o xScrum pode ter perdido um pouco da agilidade, como aconteceu no princípio 6, porque foram agregados novos documentos ao Scrum. Porém, vale destacar que os documentos agregados são simples, justamente para que este princípio continue sendo atendido.
h) No princípio 8 (o software funcionando é a medida primária de progresso), o xScrum continuou com o objetivo de entregar uma versão do software ao final de cada incremento.
i) No princípio 9 (a atenção contínua à excelência técnica e ao bom projeto melhoram a agilidade), não foi possível detectar especificações diretas para este princípio, fato este que ocorreu também quando da análise realizada com o Scrum.
j) No princípio 10 (as melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas), o xScrum manteve a idéia central do Scrum em que as equipes devem ser auto gerenciadas através de práticas e papéis responsáveis.

k) No princípio 11 (a equipe reflete, a intervalos regulares, sobre como se tornar mais efetiva), ajustando-se e otimizando-se em seu comportamento, o xScrum manteve este princípio através das reuniões diárias.
l) No princípio 12 (os stakeholders devem promover um desenvolvimento sustentável), como aconteceu no principio 9, não foi possível detectar especificações diretas para este princípio, fato este que ocorreu também quando da análise realizada com o Scrum.

Desta forma, percebe-se que apenas os princípios 6 e 7 foram os mais afetados pelas mudanças propostas pelo xScrum. No princípio 6, o desenvolvimento de novos documentos não diminuiu a comunicação face a face conforme constatado durante a execução do exemplo de aplicação do presente trabalho. Porém, no princípio 7, onde a simplicidade é fundamental, talvez as alterações propostas possam ter tornado o processo teoricamente mais complicado, fato este não constatado durante a execução do exemplo de aplicação.

Pode-se concluir então, que a agilidade presente no Scrum, conforme sugestão dos princípios ágeis, não foi afetada pelas mudanças propostas no xScrum.

6 Conclusões

O presente trabalho foi motivado pela busca de informações sobre os métodos ágeis quando relacionados a modelos de qualidade de software, ou seja, se eles poderiam conviver harmonicamente tendo princípios e idéias bem diferentes. Para isto, foram amplamente estudados os métodos ágeis, em especial o Scrum, e o modelo de qualidade de software CMMI. Uma análise do Scrum e do modelo CMMI foi realizada para encontrar conformidades, ou aderências entre eles, resultando, com isso, numa proposta de extensão deste método para total adequação ao modelo CMMI. Esta adequação foi denominada de "xScrum".

O exemplo de aplicação realizado para utilização do Scrum e sua extensão mostrou que os métodos ágeis e os modelos de qualidade podem conviver de forma harmoniosa em um mesmo ambiente, desde que se façam pequenos ajustes. Vale ressaltar que a durante a execução do exemplo de aplicação foram realizadas várias alterações à proposta do presente trabalho, sendo que a maioria destas alterações foi incluída por sugestão dos participantes envolvidos no trabalho.

As conclusões deste trabalho estão descritas neste capítulo, que está organizado desta forma: principais contribuições, problemas encontrados e trabalhos futuros.

6.1 Principais contribuições

Este trabalho relata vários métodos ágeis que atendem ao desenvolvimento de software: XP, Scrum, DSDM e FDD. Estes métodos são voltados para pequenas equipes que desenvolvem software em um ambiente onde os requisitos são instáveis. A análise

comparativa realizada entre estes métodos demonstrou que a XP, é o método “mais” ágil pois ficou em melhor concordância com os princípios ágeis propostos no manifesto ágil. O DSDM é um dos métodos mais bem definidos e organizados, apresentando características muito similares ao RUP. O FDD obteve pouca pontuação na análise realizada, sendo considerado, de certa forma, como um método intermediário, ou seja, entre os ágeis e os “pesados”. Finalmente, o Scrum, objeto principal do presente estudo, foi considerado o método ágil mais indicado para o processo de gerenciamento no desenvolvimento de software, além de possuir uma grande possibilidade de escalabilidade. Outro fator a se destacar é que o Scrum carece de mecanismos para controles na área do Gerenciamento de Requisitos.

Outra contribuição se deu na avaliação do Scrum segundo as perspectivas do modelo CMMI, nas áreas de processo Gerenciamento de Requisitos (REQM) e Desenvolvimento de Requisitos (RD). Percebeu-se com as avaliações realizadas, que o Scrum não está plenamente de acordo com as exigências do modelo CMMI, contrariamente ao que é relatado em Paulk (2001) (seção 2.4.2), que mostra que a área de processo Gerenciamento de Requisitos do SW-CMM é atendida pelo Scrum. Acredita-se que este antagonismo tenha ocorrido porque, naquele trabalho, realizou-se uma análise superficial de todas áreas de processo do SW-CMM. Outro fato interessante é que carecem estudos aprofundados de como os métodos ágeis se comportam quando comparados a modelos de qualidade de software como o CMMI. O estudo elaborado por Turner e Jain (2002), descrito na seção 1.2, não especifica que método ágil é comparado com o modelo CMMI.

Uma importante contribuição deste trabalho foi a proposta da extensão do Scrum denominada de “xScrum”. O xScrum define diretrizes para utilização do Scrum de forma a estar de acordo com as áreas de processo REQM e RD do modelo CMMI. Com isso, organizações que almejam uma certificação internacional de padrão de qualidade em software podem utilizar o xScrum como uma referência.

A aplicação da extensão proposta “xScrum” em ambientes reais de produção de software oportunizou a possibilidade de avaliar os resultados no emprego das abordagens

de desenvolvimento ágil e de modelos de qualidade em um mesmo ambiente. Este exemplo de aplicação pode confirmar que é viável a união dos métodos ágeis com modelos de qualidade de software. Esta união gera, de certa forma, um “modelo híbrido” capaz de atender as diversas necessidades de cada organização, na área de desenvolvimento de software. Nesse sentido, o xScrum pode ser uma solução viável.

Percebeu-se, pelo exemplo de aplicação realizado, que as práticas da XP podem ser utilizadas juntamente com o Scrum, principalmente pelo fato do Scrum não definir técnica para testes e codificação. Com isso, confirma-se a hipótese sugerida por Schwaber (2003, p. 34) que a XP e o Scrum podem trabalhar juntos.

6.2 Dificuldades Encontradas

Algumas dificuldades foram encontradas durante o desenvolvimento do presente trabalho.

A falta de bibliografia foi uma das maiores dificuldades encontradas para realização do presente do trabalho. A principal obra sobre Scrum relata superficialmente a aplicação deste método no gerenciamento do processo de desenvolvimento de softwares. A maioria dos artigos consultados relata experimentos da utilização da XP com o modelo CMM e não com o CMMI. Além disto, é recente a utilização do modelo CMMI, em especial na representação contínuo, nas organizações brasileiras.

Também não foram encontradas na literatura extensões propostas ao Scrum visando uma adequação ao modelo CMMI. O trabalho realizado por Paulk (2001, p.19), apresentado na seção 2.4.2, compara o Scrum com o SW-CMM e não o CMMI. Este trabalho afirma que apenas 4 das 18 áreas de processo do SW-CMM são atendidas pelo Scrum, não sugerindo extensões que poderiam ser agregadas ao método ágil Scrum visando um maior atendimento das práticas específicas presentes do modelo SW-CMM.

Outra dificuldade encontrada foi a elaboração da análise do Scrum com relação ao modelo CMMI e as soluções propostas pelas diretrizes (xScrum), pois são baseadas

fundamentalmente em interpretação do modelo CMMI. Este modelo é uma guia para melhoria de processos das organizações, portanto, foi elaborado visando ser abrangente e genérico.

6.3 Trabalhos Futuros

O presente trabalho pode ser continuado de várias formas visando aumentar a contribuição para as áreas de engenharia de software que pretendem utilizar os métodos ágeis com apoio de modelos de qualidade. Alguns trabalhos sugeridos são:

1) Desenvolver um protótipo de ferramenta computacional baseada no xScrum para atender, inicialmente, as áreas de processo REQM e RD do modelo CMMI;
2) Estender o Scrum para outras áreas de processo do modelo CMMI, em especial, para as demais áreas de processos da categoria Engenharia: Verificação, Validação, Solução Técnica e Integração de Produto;
3) A validação sistemática do xScrum em outras organizações produtoras de software utilizando também as outras práticas da XP não adotadas neste trabalho como as metáforas, *refactoring*, cartões CRC, etc.

7 Referências Bibliográficas

ABRAHAMSSON, Pekka, SALO Outi. *Agile Software Development Methods – Review and Analysis*. Espoo 2002, VTT Publications 478 107.

AHERN, Dennis. CLOUSE, Aaron. TURNER, Richard. *CMMI Distilled: a practical introduction to integrated process improvement*. Boston: Addison Wesley, 2004.

AMBLER, Scott. *Modelagem ágil*: práticas eficazes para a Programação Extrema e o Processo Unificado. Trad. Acauan Fernandes. Porto Alegre: Bookman, 2004.

ASTELS, David. *Extreme Programming: Guia Prático*. Rio de Janeiro: Campus. 2002.

BARTIÉ, Alexandre. *Garantia da Qualidade de Software*. São Paulo: Campus. 2003.

BECK, Kent. *Extreme Programming: Embrace Change*. Addison Wesley. 2000.

BEEDLE, Mike *et al*. SCRUM: *An extension pattern language for hyperprodutive software development*. Disponível em: <http://www.controlchaos.com>. Acesso em: out 2003.

BOEHM, Barry. DeMARCO, Tom. *The Agile Methods Fray*. IEEE *Computer Science*, June, p. 90-91. 2002.

_____, Barry. TURNER, Richard. *Balancing agility and discipline: a guide for the perplexed*. Boston: Addison Wesley, 2004.

CHRISSIS, Mary B; KONRAD M.; SHRUM S.; *CMMI: Guidelines for Process Integration and Product Improvement*. SEI, Addison Wesley, 2003.

COAD, Peter. *Java Modeling in Color with UML*. Prentice Hall, 1999.

COCKBURN, Alistair. *Agile Software Development*. Addison-Wesley. 2001.

COLEMAN, Gerry. *Um Processo de Software para Rapid Application Development*. Disponível em: http://www.async.com.br. Acesso em: jan 2004.

CORDEIRO, Marco Aurélio. *Uma Ferramenta Automatizada de suporte ao processo de* Gerenciamento de Requisitos. Dissertação (Mestrado) - PUCPR. Curitiba, 2002. 182p.

CÔRTES, Mario Lúcio; CHIOSSI, Thelma C. dos Santos. *Modelos de Qualidade de Software*. Campinas: Unicamp, Instituto de Computação, 2001.

De LUCA Jeff. *Feature-Driven Development (FDD) Overview Presentation*. Disponível em <http://www.nebulon.com/articles/fdd/download/fddoverview.pdf>. Acesso em Jan 2004.

DEMING, W. Eduards. *Qualidade: A Revolução da Administração*. Rio de Janeiro: Saraiva, 1990.

DSDM. Disponível em: <http://www.dsdm.org >. Acesso em: jan 2004.

FIORINI, Soeli. STAA, Arndt, BAPTISTA R. *Engenharia de software com CMM*. Rio de Janeiro: Brasport, 1998.

FOWLER, Martin. *The New Methodology*. Disponível em: < http://www.martinfowler.com/articles/newMethodology.html>. Acesso em : set 2003.

HARTMAN, Julio. *Estudo sobre a aplicação de métodos ágeis no desenvolvimento e gerenciamento de projetos de software*. UFRGS. Trabalho individual. Porto Alegre: 2003.

HEINZ, Lauren; *CMMI Myths and Realities;* Disponível em: <http://interactive.sei. cmu.edu/news@sei/features/2002/4q02/feature-4-4q02.htm>; Acesso em: jul 2003.

HIGHSMITH, Jim. *Agile Software Development Ecosystems*. Addison Wesley, 2002.

HOFMANN, Hubert F.; LEHNER, Franz. *Requirements Engineering as a Success Factor in Software Projects*. 2001.

ISO/IEC 9126. *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their use*, 1991.

LINDA Rising, NORMAN Janoff, *The SCRUM Software Development Process for Small Teams*, IEEE Software, July-August 2000.

_____, Rising. *Gerenciamento e Equipes: Reuniões Ágeis*. Disponível em: <http://members.cox.net/risingl1/articles/AgilePortuguese.pdf>. Acesso em: set 2003.

MANIFESTO. Manifesto Ágil. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: dez 2003.

MESQUITA, Renato. *Métodos ágeis: Notas de Aula*. Disponível em: <http://www.ead.eee.ufmg.br/~renato/engsoft/MetodologiasAgeis.pdf>. Acesso em: dez 2003.

NAWROCKI, Jerzy. WALTER, Bartosz. WOJCIECHOWSKI, Adam. *Comparasion of CMM level 2 and extreme programming*. Quality Connection – 7th European Conference on Software Quality, Helsinki, June 9-13, 2002. p. 288-297.

OPPERTHAUSER, Don. *Defect Management in an Agile Development Environment*. Publicado por: Crosstalk. 01 set. 2003. Disponível em:< http://www.stsc.hill.af.mil/crosstalk/2003/09/0309Opperthauser.html>.

ORNBURN, Steve. Disponível em: <http://www.stsc.hill.af.mil/crosstalk/2002/10/kane.html>. Acesso em: dez 2003.

ORR, Ken. *CMM versus agile development: Religious Wars and Software Development*. Cutter Consortium. Executive Report. Vol.3 Nº 7. 2002.

PAETSCH, F. EBERLEIN, A. MAURER, F. *Requirements Engineering and Agile Software Development*, WETICE 2003, *IEEE Computer Science*. 2003. p.308.

PALMER, Steven. *Feature Driven Development*. Disponível em <http://www.step-10.com> Acesso em Jan 2004.

PAULK, Mark. C., *Extreme Programming from a CMM Perspective*, IEEE Software, vol. 18, no. 6, 2001. p. 19-26.

_____, Mark. WEBBER, C.V., CURTIS, B., CHRISSIS, M.B.; *The Capability maturity model: guidelines for improving the software process*. SEI, 1995.

PRESSMAN, Roger S., *Engenharia de software*. São Paulo: Makron Books, 1995.

REIFER, Donald J. *XP and the CMM*. Publicado por: IEEE Computer Society. 01 jan. 2003.

ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. *Qualidade de Software*: Teoria e Prática. São Paulo: Prentice Hall, 2001.

SCHWABER, Ken, BEEDLE, Mike. *Agile Software Development with SCRUM*. Prentice Hall, 2002.

SCRUM. Disponível em: <http://www.mountaingoatsoftware.com/SCRUM>. Acesso em: set 2003.

SEI TR-028 – SOFTWARE ENGINEERING INSTITUTE, *Capability Maturity Model Integration for Software Engineering*, 2002; Disponível em: < http://www.sei.cmu.edu/publications/documents/02.reports/02tr028.html>; Acesso em: jul 2003.

SEI TR-029 - SOFTWARE ENGINEERING INSTITUTE, Technical Report 029, *Capability Maturity Model Integration for Software Engineering*, 2002; Disponível em: <http://www.sei.cmu.edu/publications/documents/02.reports/02tr029.html>; Acesso em: jul 2003.

SOMMERVILLE, Ian, *Engenharia de Software*, São Paulo: Addison-Wesley, 2003.

TURNER, Richard. JAIN, Apurva. *Agile Meets CMMI: Culture clash or common cause.* XP/Agile Universe. 2002, p.153-165.

ANEXO 1 – Práticas Específicas das áreas de processo REQM e RD

1. REQM – GERENCIAMENTO DE REQUISITOS

1.1 Gerenciamento dos Requisitos – SG1

1.1.1 Obter o entendimento dos requisitos - SP 1.1-1

Desenvolver um entendimento comum com os geradores (fornecedores) dos requisitos. São típicos produtos resultantes desta prática:

1 Lista de critérios para distinguir requisitos fornecidos;

2 Critério para avaliação e aceitação dos requisitos;

3 Análise dos resultados conforme lista de critérios estabelecidos;

4 Elaboração de um conjunto de requisitos aceitos.São produtos dessa prática as seguintes sub práticas:

6 Estabelecer critérios para distinguir requisitos fornecidos;

7 Estabelecer critérios objetivos para aceitação dos requisitos;

8 Analisar os requisitos para assegurar que os critérios estabelecidos sejam encontrados;

9 Chegar a um acordo dos requisitos para que os *stakeholders* do projeto possam executá-lo.

1.1.2 Obter o compromisso para os requisitos: SP 1.2-2

Obter o compromisso para os requisitos dos *stakeholders* do projeto. São típicos produtos de trabalhos resultantes desta prática a validação dos impactos dos requisitos e o compromisso na documentação dos requisitos e suas mudanças. Além disso, são produtos dessa prática as seguintes sub práticas:

1 Avaliação do impacto dos requisitos nos compromissos existentes;

2 Negociação e registro dos compromissos.

1.1.3 Gerenciar as mudanças dos Requisitos: SP 1.3-1

Gerenciar as mudanças dos requisitos que acontecem durante o projeto. São típicos produtos de trabalhos resultantes desta prática:

1 Status dos requisitos;

2 Banco de Dados dos requisitos;

3 Banco de Dados para Tomada de decisão dos requisitos.

Além disso, são produtos dessa prática as seguintes sub práticas:

1 Capturar todos requisitos e as mudanças geradas pelo projeto;

2 Manter a história das mudanças dos requisitos;

3 Avaliar o impacto das mudanças dos requisitos do ponto de vista dos *stakeholders* importantes;

4 Realizar os requisitos e tornar disponíveis suas mudanças no projeto.

1.1.4 Manter a rastreabilidade bidirecional dos requisitos: SP 1.4-2

Manter a rastreabilidade bidirecional entre o plano do projeto e produto de trabalho dos requisitos. São típicos produtos de trabalhos resultantes desta prática:

5 Matriz da rastreabilidade dos requisitos;
6 Sistema de acompanhamento dos requisitos.

Além disso, são produtos dessa prática as seguintes sub práticas:

1 Manter a rastreabilidade dos requisitos para assegurar que a origem dos requisitos seja documentada;
2 Manter a rastreabilidade dos requisitos para derivar requisitos como alocação de recursos, objetos, pessoas, processos e produtos de trabalho;
3 Manter a rastreabilidade horizontal função por função e através das interfaces;
4 Gerar uma matriz da rastreabilidade dos requisitos.

1.1.5 Identificar inconsistências entre o Projeto de Trabalho e os requisitos :SP 1.5-1

Esta prática deve resultar em:

1 Documentação das inconsistências incluindo fontes, condições e lógica;
2 Ações Corretivas.

Além disso, são produtos dessa prática as seguintes sub práticas:

1 Revisar os planos dos projetos, atividades, e produtos de trabalho para verificar (consistir) com os requisitos e as mudanças feitas nele.
2 Identificar a fonte das inconsistências;
3 Identificar mudanças que são necessárias para realizar o plano e o produto de trabalho resultante das mudanças dos requisitos.
4 Iniciar ações corretivas.

A seguir, a descrição da área de processo Desenvolvimento de Requisitos, conforme o SEI TR-028 (2002) e Chrissis et al. (2003), em tradução livre:

2 Desenvolvimento dos Requisitos (RD)

2.1 Desenvolver os Requisitos dos Clientes - SG1

2.1.1 Coletar as necessidades dos *stakeholders*: SP 1.1-1

Identificar e coletar as necessidades dos *stakeholders*, expectativas, restrições e interfaces para todas as fases do ciclo de vida do produto. Esta prática específica, não resulta em produtos de trabalho, e ocorre somente, quando utilizada na Representação Contínuo.

2.1.2 Eliciar as necessidades: SP 1.1-2

Eliciar as necessidades dos *stakeholders*, expectativas, restrições e interfaces para todas as fases do ciclo de vida do produto. Não existem típicos produtos de trabalho para esta prática. A sub prática é, envolver os *stakeholders* mais importantes utilizando métodos para elicitação das necessidades, expectativas, restrições e interfaces externas.

2.1.3 Desenvolver os Requisitos dos clientes: SP 1.2-1

Transformar as necessidades dos *stakeholders*, expectativas, restrições e interfaces para os requisitos dos clientes. São típicos produtos de trabalhos resultantes desta prática:
1 Requisitos dos clientes;
2 Restrições dos clientes na condução da verificação;
3 Restrições dos clientes na condução da validação.
4 Além disso, são produtos dessa prática as seguintes sub práticas:
5 Traduzir as necessidades dos *stakeholders*, expectativas, restrições e interfaces num documento com os requisitos dos clientes;
6 Definir as restrições para verificação e validação.

2.2 Desenvolver os Requisitos dos Produtos: SG2

2.2.1 Estabelecer os requisitos dos produtos e dos componentes dos produtos: SP 2.1-1

Estabelecer e manter os requisitos dos produtos e dos componentes dos produtos, os quais serão baseados nos requisitos dos clientes. São típicos produtos de trabalhos resultantes desta prática:

1 Requisitos derivados;
2 Requisitos dos produtos;
3 Requisitos dos componentes dos produtos.

Além disso, são produtos dessa prática as seguintes sub práticas:

1 Desenvolver requisitos em termos técnicos necessários para o projeto do produto e seus componentes;
2 Derivar requisitos resultantes de decisão do projeto;
3 Estabelecer e manter relacionamentos entre requisitos para verificação durante o gerenciamento das mudanças e alocação dos requisitos;

2.2.2 Alocar os requisitos dos componentes dos produtos: SP 2.2-1

Alocar os requisitos para cada componente dos produtos. Típicos produtos de trabalhos resultantes desta prática:

1 Planilhas com alocação dos requisitos;
2 Alocação dos requisitos temporários;
3 Projeto das restrições;
4 Requisitos derivados;
5 Relacionamentos entre os requisitos derivados.

Além disso, são produtos dessa prática as seguintes sub práticas:
1 Alocar requisitos funcionais;
2 Alocar requisitos para componentes dos produtos;
3 Alocar restrições de projeto para componentes dos produtos;
4 Documentar relacionamentos entre os requisitos alocados.

2.2.3 Identificar as interfaces dos requisitos: SP 2.3-1

É um produto de trabalho resultante desta prática, a elaboração dos requisitos de interface, e as sub práticas são:

1 Identificar interfaces internas e externas do produto;
2 Desenvolver os requisitos para identificação das interfaces.

2.3 Analisar e Validar os Requisitos: SG3

2.3.1 Estabelecer conceitos operacionais e cenários: SP 3.1-1

Um cenário é uma seqüência de eventos que podem acontecer no uso do produto. São saídas desta prática:

1 Conceitos operacionais;
2 Instalação de produto, operação, manutenção e conceitos de suporte;
3 Disposição dos conceitos utilizados;
4 Construção de Cenários;
5 Use cases;
6 Novos requisitos;

São sub-práticas resultantes desta prática específica:

1 Desenvolver conceitos operacionais e cenários que incluem funcionalidade, performance, manutenção e suporte;
2 Definir o ambiente em que o produto irá operar, incluindo restrições e limites.
3 Rever conceitos operacionais e cenários para refinar e descobrir novos requisitos;

4 Desenvolver conceitos operacionais em detalhes, como produtos e componentes de produtos que são selecionados, qual definição de interação de produtos, usuário final, ambiente, e o que satisfaz as operações, manutenções e suporte.

2.3.2 Estabelecer uma definição das funcionalidades requeridas: SP 3.2-1

São típicos produtos de trabalhos resultantes desta prática a Arquitetura funcional; Diagrama de Atividades e Uses Cases e Análise Orientada a Objeto com identificação de serviços. E, esta prática específica tem como resultado as seguintes sub práticas:

1 Analisar e quantificar requisitos funcionais solicitados pelos usuários finais;
2 Analisar requisitos para identificar a lógica funcional;
3 Separar os requisitos em grupos, baseado em critérios, como por exemplo, performance. Isto servirá para facilitar a análise dos requisitos.
4 Considerar a seqüência da função de tempo-crítico durante o desenvolvimento de um componente do produto;
5 Alocar as funcionalidades dos requisitos.
2.3.3 Analisar os requisitos: SP 3.3-1

Esta prática deverá resultar em:

1 Relatórios dos defeitos dos requisitos;
2 Propor mudanças nos requisitos para resolver defeitos;
3 Apresentar requisitos chaves;
4 Elaborar medidas técnicas de performance.
5 Esta prática específica tem como resultados, as seguintes sub práticas:
6 Analisar as necessidades dos *stakeholders*, expectativas, restrições e interfaces para remover conflitos e para organizar assuntos relacionados;
7 Analisar requisitos para assegurar que estão completos, praticáveis, realizáveis e verificáveis;
8 Identificar requisitos chaves que possuem forte influência no custo, funcionalidade, riscos ou performance;
9 Analisar conceitos operacionais e cenários para refinar as necessidades dos clientes, restrições e interfaces a fim de descobrir novos requisitos.
2.3.1 Analisar os requisitos para avaliação: SP 3.4-3

Avaliar os riscos relatados nos requisitos é produto de trabalho resultante desta prática. Para isso deve-se:

1 Utilizar modelos, simuladores, protótipos para analisar o risco das necessidades dos *stakeholders* e suas restrições;
2 Executar uma avaliação do risco nos requisitos e na arquitetura funcional;
3 Examinar o impacto que os riscos dos requisitos produzem no ciclo de vida do produto;
2.3.2 Validar os requisitos: SP 3.5-1

Deve-se analisar e validar os requisitos para determinar o risco que poderá resultar caso o produto não esteja de acordo com o esperado pelo ambiente. Cabe ressaltar que esta prática, bem como a RD SP1.1-1, ocorre somente quando utilizada na Representação Contínuo.

2.3.3 Validar os requisitos com métodos válidos: SP 3.5-2

Validar os requisitos para assegurar que o resultado esperado pelo produto seja de acordo com o esperado. Porém, esta prática necessita da utilização de algum método reconhecido para validação. A saída desta prática é registrar os métodos de análise e os resultados, para futura consulta. Esta prática específica tem como resultados as seguintes sub práticas:

1 Analisar os requisitos para determinar o risco que resultará caso o produto não esteja de acordo com o esperado pelo ambiente;
2 Explorar a adequação e completude dos requisitos no desenvolvimento dos produtos, obtendo retorno sobre eles dos *stakeholders* mais importantes;
3 Avaliar a maturidade do projeto no contexto da validação dos requisitos no ambiente para identificar casos já validados anteriormente.

ANEXO 2 – PLANEJAMENTO DA AVALIAÇÃO

Avaliação de processo de desenvolvimento de software para pequenas organizações

QuickLocus

Plano para a avaliação

Do
Simuplan

Outubro de 2004

Objetivo da avaliação

O Simuplan, na busca pela melhoria de processo está iniciando a implantação de modelo de qualidade de software baseada no CMMI. Com o objetivo de fornecer subsídios para o plano de implantação do modelo, será realizada uma avaliação para verificar a aderência e lacunas no processo da organização em relação às práticas do modelo.
Parâmetros da avaliação

Modelo de Referência

O modelo definido como referência para esta avaliação é o CMMI de autoria de CHRISSIS, Mary B; KONRAD M.; SHRUM S.; *CMMI: Guidelines for Process Integration and Product Improvement*. SEI, Addison Wesley, 2003.

Escopo no modelo

O objetivo é a avaliação das seguintes das áreas de processo do modelo CMMI:
Gerenciamento de Requisitos - REQM (Requirements Management)
Desenvolvimento de Requisitos - RD (Requirements Development)

Patrocinador da avaliação

Os patrocinadores da avaliação são o Sr. Willingthon Pavan, coordenador do grupo e o Dr. Maurício pesquisador da Embrapa Trigo de Passo Fundo RS

Escopo organizacional

O escopo organizacional foi definido como a área de desenvolvimento conforme o processo da organização.

Seleção de projetos para avaliação

Portal Simuplan.
Projetos selecionados para a avaliação

Total de projetos selecionados: 1		
Projeto: **Portal Simuplan**	Coordenador: Elias Müller	Processo: Cadastro de agricultores, Alimentação de dados, Autorização. Obtenção de dados pela web, Módulo de relatórios.
Papel		**Quantidade de Pessoas**
Analista de sistemas		1

Programador	6
Gerente de projeto	1
Total de *stakeholders* no projeto	8

Cronograma geral da avaliação

O cronograma geral das atividades da avaliação é fornecido na tabela a seguir.

Atividade	Data	Horário
Entrega dos formulários para levantamento de dados da organização	08/10/04	
Entrega dos formulários para levantamento de dados do processo	08/10/04	
Devolução dos formulários preenchidos sobre a organização	11/10/04	
Devolução dos formulários preenchidos sobre o processo	11/10/04	
Envio do plano preliminar da avaliação	13/10/04	
Acordo da organização sobre o plano	14/10/04	9:30-11:30
Treinamento da equipe de avaliação	14/10/04	7:30-9:30
Atividades de avaliação na organização	15/10/04	8:00–18:00
Envio do relatório final para aprovação da organização	20/10/04	
Reunião final de apresentação dos resultados com todos *stakeholders*	21/10/04	8:30-9:30

Participantes da avaliação

Nome	**Projeto**	**Papel no projeto**	**Cargo organizacional**	**Chefia**
Mauricio	Portal Simuplan		Responsável	
Willingthon Pavan	Portal Simuplan	Líder de Projeto	Coordenador	Mauricio
Elias Muller	Portal Simuplan	Gerente de Projeto	Coordenador	Mauricio
Telmo De Cesaro Jr	Portal Simuplan	Programador	Consultor	Mauricio
Samuel Zandonai	Portal Simuplan	Programador	Consultor	Mauricio
Jonas Augusto Prediger	Portal Simuplan	Programador	Consultor	Mauricio
Élder Francisco Fontana	Portal Simuplan	Programador	Consultor	Mauricio
Ricardo Ogliari e	Portal Simuplan	Programador	Consultor	Mauricio

Adriano Filippi	Portal Simuplan	Programador	Consultor	Mauricio
Juliano Tonezer da Silva	Portal Simuplan	Analista de Sistemas	Consultor	Mauricio
Cristiano Roberto Cervi	Portal Simuplan	Analista de Sistemas	Consultor	Mauricio
Total de participantes:	11			

Avaliação

A avaliação é formada pelo avaliado externo Alexandre Lazaretti Zanatta.

Estratégia da coleta de dados

Os dados serão coletados através das respostas aos questionários enviados para a organização e dos depoimentos fornecidos pelos participantes durante as entrevistas.
Responsabilidades dos membros da equipe

Nome	Papel na equipe	Responsabilidade
Willingthon Pavan	Líder de Projeto	REQM e RM

Cronograma das atividades na organização

Plano para as atividades de avaliação do Processo de Desenvolvimento de Software do Simuplan, a serem realizadas por profissionais da equipe do método QuickLocus no dia 13/10/2004 .

Horário	**Assunto**	**Atividade**	**Pessoas**
8h00	Planejamento 1	Planejamento da entrevista com líder do projeto	Equipe avaliadora
8h15	Abertura	Reunião de abertura	Todos os *stakeholders*
8h30	Entrevista 1	Entrevista com líder de projeto	Equipe avaliadora Entrevistados Willingthon Pavan e Elias Muller
9h00	Fechamento 1	Fechamento da Entrevista 1	Equipe avaliadora
9h15	Planejamento 2	Planejamento da entrevista com gerência do projeto Portal	Equipe avaliadora
9h30	Entrevista 2	Entrevista com gerência do projeto Portal	Equipe avaliadora Entrevistados Willingthon Pavan e Elias Muller
11h00	Fechamento 2	Fechamento da Entrevista 2	Equipe avaliadora
13h00	*Almoço*		
14:00	Reunião de trabalho	Consolidação e elaboração do relatório preliminar	Equipe avaliadora
15h00		Encerramento do dia de trabalho	

Responsabilidades da organização

Providenciar a infra-estrutura necessária para realização da avaliação.
Revisar e aprovar o plano.
Garantir a presença dos participantes nos horários previstos
Infra-estrutura necessária

Para realização dos trabalhos são necessários:

Sala de reuniões com mesa e cadeira para no mínimo 9 pessoas para a realização das entrevistas e trabalhos da equipe
Pontos de energia elétrica disponível para ligação de computadores
Dois computadores
Projetor de slides ou canhão para ser ligado a um computador
Local para projeção de slides
Mesa de apoio para material de trabalho
Premissas para elaboração do plano
A equipe de avaliação deve conhecer a estrutura do modelo e os detalhes das áreas chave do escopo (REQM, RD).
A equipe de avaliação deve participar do treinamento sobre o método QuickLocus.

Compromisso do patrocinador para apoio e realização dos trabalhos de avaliação.

Papel	Nome	Assinatura	Data
Patrocinador	Maurício Fernandes		
Líder de projeto	Willingthon Pavan		
Coordenador (organização)	Elias Muller		

ANEXO 3 – COLETA DE DADOS FONTE 1

Informações iniciais sobre o processo de desenvolvimento de software

Empresa – Simuplan **Data: Set/04**

Existe uma estrutura organizacional definida para cada projeto?	Sim	
Existe uma área da qualidade?		Não
Se sim, a área da qualidade examina produto e/ou processo?	Produto	Processo
A organização possui sistemática para controle de documentos e versões de software?		Não
A sistemática é documentada?		Não
Existe sistemática para planejamento e acompanhamento de projeto?		Não
A sistemática é documentada?		Não
Existe sistemática para definição e controle de requisitos?		Não
A sistemática é documentada?		Não
A organização faz uso de serviços de terceiros para desenvolvimento e manutenção de software?		Não

Comentários adicionais:
O desenvolvimento de Projetos é baseado no Método ágil Scrum. O Gerenciamento e o Controle das Atividades são exercidos pelo Scrum Master, que é o responsável pelo desenvolvimento como um todo e pelo posicionamento perante o cliente, além da realização das reuniões diárias estabelecidas no cronograma.

As equipes envolvidas nos trabalhos são contempladas basicamente por Analistas e Programadores de Sistemas, perfeitamente compatíveis com as plataformas, ferramentas e linguagens de acordo com as necessidades definidas no escopo do projeto.

O Simuplan não possui uma metodologia própria, podendo com isso, desenvolver projetos com baixa qualidade e pouca previsibilidade.

A equipe de gerenciamento é composta por profissionais capacitados e com excelente conhecimento técnico. Porém , os programadores são alunos de graduação com pouca experiência profissional mas com conhecimento em várias plataformas, ferramentas e linguagens de programação.

ANEXO 4 – GRADUAÇÃO DOS ITENS DAS ÁREAS DE PROCESSO

CMMI – *Capability Maturity Model Integration*

Área de Processo: REQM – Gerenciamento de Requisitos

Legenda: E - prática existe **M** - prática existe e deve ser melhorada **N -** prática não existe

Prática	Seq	Descrição	Sit
SP1.1-1	1	Os requisitos são esclarecidos com quem define os requisitos?	E
SP1.2-2	2	Os participantes do projeto se comprometem com os requisitos?	E
SP1.3-1	3	As mudanças nos requisitos durante o projeto, são gerenciadas?	E
SP1.4-2	4	Existe rastreabilidade bidirecional entre os requisitos e os planos de projeto e os produtos de trabalho?	E
SP1.5-1	5	As inconsistências entre os planos de projeto, produtos de trabalho e requisitos são identificados?	E

Área de Processo: RD – Desenvolvimento de Requisitos

Prática	Seq	Descrição	Sit
SP 1.1-1	1	São identificadas e coletadas as necessidades dos *stakeholders?*	E
SP 1.1-2	2	Utiliza-se de alguma técnica para elicitação das necessidades dos *stakeholders*?	E
SP 1.2-1	3	Existe uma forma de transformação das necessidades dos *stakeholders*, expectativas, restrições e interfaces em requisitos dos clientes?	E
SP 2.1-1	4	São estabelecidos e mantidos os requisitos dos produtos e dos componentes dos produtos que serão baseados nos requisitos dos clientes	E
SP 2.2-1	5	São alocados os requisitos para cada componente dos produtos?	E
SP 2.3-1	6	São elaborados os requisitos de interface?	E

SP 3.1-1	7	São utilizados cenários ?	E
SP 3.2-1	8	São desenvolvidos diagramas de atividades e casos de usos?	E
SP 3.3-1	9	São gerados relatórios dos defeitos dos requisitos, e se propõe mudança nos requisitos para resolver estes defeitos?	E
SP 3.4-3	10	São avaliados os riscos relatados nos requisitos?	E
SP 3.5-1	11	São analisados e validados os requisitos para determinar o risco que poderá resultar caso o produto não esteja de acordo com o esperado pelo ambiente?	E
SP 3.5-2	12	São validados os requisitos para assegurar que o resultado esperado pelo produto seja de acordo com o esperado?	E

ANEXO 5 – RELATÓRIO FINAL AVALIAÇÃO

Avaliação de processo de desenvolvimento de software para pequenas organizações

QuickLocus

Relatório Final
Do
Simuplan

Outubro de 2004

A empresa e a organização

Simuplan é um grupo de pesquisa interdisciplinar do Curso de Ciência da Computação da Universidade de Passo Fundo, que mantém com a Embrapa Trigo, sob coordenação dos professores Ms.c Willingthon Pavan e Dr. José Mauricio Cunha Fernandes. O projeto tem como objetivo desenvolver aplicações (softwares) de simulação do crescimento de pragas em algumas culturas, como por exemplo, o trigo. Existe há aproximadamente 3 anos, possuindo participações relevantes no cenário científico nacional e internacional. Como resultados já apresentados pelo Simuplan, podem-se citar o desenvolvimento dos seguintes softwares: SimTrigo (Simulação do Crescimento e Desenvolvimento do Trigo e suas Pragas), Sisalert (Sistema de alerta para macieiras, alho, milho, morango e outros), SimuSoja (Simulação do ataque da ferrugem da soja), entre outros.

O Simuplan, na busca pela melhoria de processo está iniciando a implantação de modelo de qualidade de software. Com o objetivo de fornecer subsídios para o plano de implantação do modelo, será realizada uma avaliação para verificar a aderência e lacunas no processo da organização em relação às práticas do modelo.

Objetivos da Avaliação

O objetivo é a avaliação das seguintes das áreas de processo do modelo CMMI: Gerenciamento de Requisitos - REQM (*Requirements Management*) e o Desenvolvimento de Requisitos - RD (*Requirements Development*)

Os patrocinadores da avaliação são o Sr. Willingthon Pavan e o coordenador do grupo o Dr. Maurício pesquisador da Embrapa Trigo de Passo Fundo RS.

Projetos Avaliados

Processos de desenvolvimento de projetos

Portal Simuplan

Projetos selecionados para a avaliação

Total de projetos selecionados: 1		
Projeto: **Portal Simuplan**	Coordenador: Willingthon	Processo: 1.Cadastro de agricultores e produtores 2.Alimentação dos dados 3.Módulo de Autorização

		Processo: 1.Obtenção de dados pela web 2. Módulos de Relatórios

Participantes da avaliação

Equipe de Avaliação

A equipe de avaliação foi composta de duas pessoas. O líder Quicklocus é Alexandre Lazaretti Zanatta e o membro da equipe de avaliação é o Elias Muller.

Entrevistados

O número total de entrevistados foi de 9 pessoas, que fornecem dados para avaliação do processo. Seus papéis no projeto estão apresentados a seguir:

Tabela 1. Número de entrevistados por Papel.

Papel na equipe	Número de entrevistados
Gerencia Sênior	1
Gerência de Projetos	1
Desenvolvedores	6
Analista de Sistemas	1
Total	**9**

Resultado por Graduação

Os resultados foram graduados conforme a seguinte classificação:

E: o item existe

M: o item existe e deve ser melhorado

N: o item não existe

Item em maior grau de detalhes existentes

Todas as áreas de processo foram consideradas existentes.

Item em maior grau de detalhes existentes e que devem ser melhorados

Inexistem itens com esta graduação.

Item em maior grau de detalhes não existentes

Inexistem itens com esta graduação.

Resultados por Área de Processo

Gerenciamento de Requisitos

A área de processo do Gerenciamento de Requisitos descreveu as atividades para obter e controlar as mudanças nos requisitos assegurando que as mudanças relacionadas aos planos do projeto permanecessem atualizadas. Isso aconteceu devido ao desenvolvimento da rastreabilidade dos requisitos do cliente.

Esta área assegurou as alterações dos requisitos relacionados aos planos do projeto, atividades e produtos. Os requisitos foram gerenciados e verificam-se inconsistências do plano do projeto com os produtos de trabalho finais. O projeto manteve-se em um conjunto de requisitos aprovados do ciclo de vida do projeto através da execução das seguintes atividades:

e) Gerenciou-se todas as mudanças dos requisitos;

f) Manteve-se o relacionamento entre os requisitos, o plano do projeto e os produtos de trabalho;

g) Identificou-se as inconsistências entre os requisitos, o plano do projeto e os produtos de trabalho;

h) Realizou-se ações corretivas.

Desta forma, a área de processo Gerenciamento de Requisitos do modelo CMMI está totalmente presente no Simuplan.

Desenvolvimento de Requisitos

A área de processo do Desenvolvimento de Requisitos identificou as necessidades do cliente e traduziu estas necessidades em requisitos do produto. Estes requisitos foram analisados para produzirem uma solução conceitual, descrevendo a apresentação do produto, as características, a verificação dos requisitos, etc;

O objetivo desta área de processo foi produzir e analisar os requisitos dos clientes, dos produtos e os componentes do produto. Para o desenvolvimento desta área, foram realizadas as seguintes atividades:

e) Eliciou-se, analisou-se e validou-se as necessidades e expectativas dos clientes visando um bom entendimento entre todos *stakeholders* do projeto;

f) Colecionou-se e coordenou-se as necessidades dos *stakeholders*;

g) Desenvolveu-se o ciclo de vida dos requisitos do produto;

h) Estabeleceu-se os requisitos do cliente;

i) Estabeleceu-se um produto inicial e os componentes do produto em concordância com os requisitos do cliente.

Desta forma, a área de processo Desenvolvimento de Requisitos do modelo CMMI está totalmente presente no Simuplan.

Acordos

Nesta seção é documentado o acordo quanto ao conteúdo do relatório final por parte do Simuplan e da equipe avaliadora

Simuplan	**Nome**	**Assinatura**	**Data**
Patrocinador	Maurício		
Coordenador	Willingthon		
Equipe da Avaliação			
Quicklocus	Alexandre		
Participação especial	Sarah Kohan*		

* a autora do método, auxiliou remotamente no processo de avaliação do Simuplan.